

Business Transaction Protocol

An OASIS Committee Specification

CURRENT STATUS : internal committee draft

Version 1.0 [0.9.1.2]

DD Mmm 2002²¹ [8 February 2002 19:26]

<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

[Change marks relative to 0.9.1](#)

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

196	ENROLLED	40
197	RESIGN	41
198	RESIGNED	42
199	PREPARE	43
200	PREPARED	43
201	CONFIRM	45
202	CONFIRMED	46
203	CANCEL	47
204	CANCELLED	48
205	CONFIRM_ONE_PHASE	49
206	HAZARD	50
207	CONTRADICTION	51
208	SUPERIOR_STATE	51
209	INFERIOR_STATE	53
210	REDIRECT	55
211	Messages used in control relationships	56
212	BEGIN	56
213	BEGUN	57
214	PREPARE_INFERIORS	58
215	CONFIRM_TRANSACTION	60
216	TRANSACTION_CONFIRMED	62
217	CANCEL_TRANSACTION	62
218	CANCEL_INFERIORS	63
219	TRANSACTION_CANCELLED	64
220	REQUEST_INFERIOR_STATUSES	65
221	INFERIOR_STATUSES	66
222	Groups – combinations of related messages	69
223	CONTEXT & application message	69
224	CONTEXT_REPLY & ENROL	69
225	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED	70
226	CONTEXT_REPLY & ENROL & application message (& PREPARED)	71
227	BEGUN & CONTEXT	72
228	BEGIN & CONTEXT	72
229	Standard qualifiers	72
230	Transaction timelimit	72
231	Inferior timeout	73
232	Minimum inferior timeout	74
233	Inferior name	75
234	State Tables	76
235	Explanation of the state tables	76
236	Status queries	76
237	Decision events	76
238	Disruptions – failure events	77
239	Invalid cells and assumptions of the communication mechanism	77
240	Meaning of state table events	78
241	Persistent information	82
242	Failure Recovery	95

243	Types of failure	95
244	Persistent information	96
245	Redirection.....	97
246	Terminator:Decider failures.....	98
247	XML representation of Message Set.....	98
248	Addresses	99
249	Qualifiers	99
250	Identifiers	100
251	Message References.....	100
252	Messages.....	100
253	CONTEXT.....	100
254	CONTEXT_REPLY	100
255	REQUEST_STATUS	101
256	STATUS	101
257	FAULT.....	101
258	ENROL	103
259	ENROLLED	104
260	RESIGN	104
261	RESIGNED.....	104
262	PREPARE	105
263	PREPARED	105
264	CONFIRM	105
265	CONFIRMED	106
266	CANCEL	106
267	CANCELLED.....	106
268	CONFIRM_ONE_PHASE	107
269	HAZARD.....	107
270	CONTRADICTION.....	107
271	SUPERIOR_STATE.....	108
272	INFERIOR_STATE.....	108
273	REDIRECT.....	108
274	BEGIN	109
275	BEGUN.....	109
276	PREPARE_INFERIORS	110
277	CONFIRM_TRANSACTION	110
278	TRANSACTION_CONFIRMED.....	110
279	CANCEL_TRANSACTION	111
280	CANCEL_INFERIORS	111
281	TRANSACTION_CANCELLED.....	112
282	REQUEST_INFERIOR_STATUSES	112
283	INFERIOR_STATUSES	112
284	Standard qualifiers	115
285	Transaction timelimit.....	115
286	Inferior timeout	115
287	Minimum inferior timeout	115
288	Inferior name.....	115
289	Compounding of Messages.....	115

290	XML Schemas	117
291	XML schema for BTP messages.....	117
292	XML schema for standard qualifiers	130
293	Carrier Protocol Bindings	132
294	Carrier Protocol Binding Proforma.....	132
295	Bindings for request/response carrier protocols	133
296	Request/response exploitation rules.....	134
297	SOAP Binding	135
298	Example scenario using SOAP binding	137
299	SOAP + Attachments Binding.....	139
300	Conformance	141
301	Part 3. Appendices.....	144
302	A. Glossary.....	144
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

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

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

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

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

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

bt-spec@lists.oasis-open.org

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

The main list of the committee is:

business-transaction@lists.oasis-open.org

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 An Inferior which is also an Atomic Superior.
755

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

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

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

768 **Sub-composer**

769
770 An Inferior which is also a Cohesive Superior.

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

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

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

782
783

784 **Roles involved in the control relationships**

785

786 **Decider**

787

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

794

795 A Decider is instructed to cancel by receiving CANCEL_TRANSACTION.

796

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

800

801 All Deciders receive

802 CONFIRM_TRANSACTION

803 CANCEL_TRANSACTION

804 REQUEST_INFERIOR_STATUSES

805

806 All Deciders send
807 CONFIRM_COMPLETE
808 CANCEL_COMPLETE
809 INFERIOR_STATUSES

810
811

812 Coordinator

813

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

816

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

818

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

823

824 Composer

825

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

828

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

831

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

836

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

841

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

844

845

846 Terminator

847

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

850

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

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

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

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

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

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

880
881 A Terminator receives
882 CONFIRM_COMPLETE
883 CANCEL_COMPLETE
884 INFERIOR_STATUSES

885 886 Initiator

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

891 An Initiator sends

892 BEGIN
893 BEGIN & CONTEXT

894 to a Factory, and receives in reply

895 BEGUN & CONTEXT
896

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

Factory

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

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

A Factory receives

BEGIN
BEGIN & CONTEXT

and replies with

BEGUN & CONTEXT

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

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

Other roles

Redirector

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

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

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

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

948

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

951

952 **Status Requestor**

953

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

959

960 A Status Requestor sends

961

962 REQUEST_STATUS

963 REQUEST_INFERIOR_STATUSES

964

965 and receives

966

967 STATUS

968 INFERIOR_STATUSES

969

970 in response.

971

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

975

976 **Abstract Messages and Associated Contracts**

977

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

981

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

983

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

986

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

989

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

991

992 and

993

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

995

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

998

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

1005

1006

1007 **Addresses**

1008

1009 All of the messages except CONTEXT and CONTEXT_REPLY have a “target address”
1010 parameter and many also have other address parameters. These latter identify the desired
1011 target of other messages in the set. In all cases, the exact value will invariably have been
1012 originally determined by the 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 ~~compound~~ target address. This allows full
1040 flexibility for 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). ~~In cases b) and c), the identifier is to
1052 some extent redundant, although interoperability requires that it always be present.~~

1053
1054 BTP recovery requires that the state information for a Superior or Inferior is accessible after
1055 failure and that the peer can distinguish between temporary inaccessibility and the permanent
1056 non-existence of the state information. As is explained in “Redirection” below, BTP provides
1057 mechanisms – having a set of BTP addresses for some parameters, and the REDIRECT
1058 message – that make this possible, even if the recovered state information is on a different
1059 address to the original one (as may be the case if case c) above is used).
1060
1061

1062 **Request/response pairs**

1063
1064 Many of the messages combine in pairs as a request and its response. However, in some cases
1065 the response message is sent without a triggering request, or as a possible response to more
1066 than one type of request. To allow for this, the abstract message set treats each message as
1067 standalone; but where a request does expect a reply, a “reply-address” parameter will be
1068 present. For any message with a reply address parameter, in the case of certain errors, a
1069 FAULT message will be sent to the reply address instead of the expected reply.
1070

1071 For messages which are specified as sent between Superior and Inferior, a FAULT message is
1072 sent to the peer.
1073

1074 **Compounding messages**

1075
1076 BTP messages may be sent in combination with each other, or with other (application)
1077 messages. There are two cases:
1078
1079 a) Sending the messages together where the combination has semantic
1080 significance. One message is said to be “related to” the other – the combination
1081 is termed a “group”.
1082 b) Sending of the messages where the combination has no semantic significance,
1083 but is merely a convenience or optimisation. This is termed “bundling” – the
1084 combination is termed a “bundle”.
1085

1086 The form A&B is used to refer to a combination (group) where message B is sent in relation
1087 to A (“relation” is asymmetric). The form A+B is used to refer to A and B bundled together-

1088 the transmission of the bundle "A+B" is semantically identical to the transmission of A
1089 followed by the transmission of B.

1090
1091 Only certain combinations of messages are possible in a group, and the meaning of the
1092 relation is specifically defined for each such combination in the next section. A particular
1093 group is treated as a unit for transmission – it has a single target address. This is usually that
1094 of one of the messages in the group – the specification for the group defines which.

1095
1096 A “bundle” of messages may contain both unrelated messages and groups of related
1097 messages. The only constraint on which messages and groups can be bundled is that all have
1098 the same binding address, but may have different “additional information” values. (Messages
1099 within a related group may have different addresses, where the rules of their relatedness
1100 permit this). Unless constrained by the binding, any messages or groups that are to be sent to
1101 the same binding address may be bundled – the fact that the binding addresses are the same is
1102 a necessary and sufficient condition for the sender to determine that the messages can be
1103 bundled.

1104
1105 A particular and important case of related messages is where a BTP CONTEXT message is
1106 sent related to an application message. In this case, the target of the application message
1107 defines the destination of the CONTEXT message. The receiving implementation may in fact
1108 remove the CONTEXT before delivering the application message to the application (Service)
1109 proper, but from the perspective of the sender, the two are sent to the same place.
1110 The compounding mechanisms, and the multi-part address structures, support the “one-wire”
1111 and “one-shot” communication patterns.

1112
1113 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,
1114 including the associated application messages, pass via the same “endpoints”. These
1115 “endpoints” may in fact be relays, routing messages on to particular actors within their
1116 domain. The onward routing will require some further addressing, but this has to be opaque to
1117 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors
1118 in its domain have the relay’s address as their binding address, and any routing information it
1119 will need in its own domain is placed in the additional information. (This may involve the
1120 relay changing addresses in messages as they pass through it on the way out). On receiving a
1121 message, it determines the within-domain destination from the received additional
1122 information (which is thus rewritten) and forwards the message appropriately. The sender is
1123 unaware of this, and merely sees addresses with the same binding address, which it is
1124 permitted to bundle. The content of the “additional information” is a matter only for the relay
1125 – it could put an entire BTP address in there, or other implementation-defined information.
1126 Note that a quite different one-wire implementation can be constructed where there is no
1127 relaying, but the receiving entity effectively performs all roles, using the received identifiers
1128 to locate the appropriate state.

1129
1130 “One-shot” communication makes it possible to send an application message, receive the
1131 application reply, enrol an Inferior to be responsible for the confirm/cancel of the operations
1132 of those message and inform the Superior that the Inferior is prepared, all in one two-way
1133 exchange across the network (e.g. one request/reply of a carrier protocol).. The application
1134 request is sent with a related CONTEXT message. The application response is sent with a

1135 relation group of CONTEXT_REPLY/related, ENROL/no-rsp-req message and a
1136 PREPARED message. This is possible even if the Superior address is different from the
1137 address of the application element that sends the original message (if the application
1138 exchange is request/reply, there may not even be an identifiable address for the application
1139 element). The target addresses of the ENROL and PREPARED (the Superior address) are not
1140 transmitted; the actor that was originally responsible for adding the CONTEXT to the
1141 outbound application message remembers the Superior address and forwards the ENROL and
1142 PREPARED appropriately.

1143
1144 With “one-shot”, if there are multiple Inferiors created as a result of a single application
1145 message, there is an ENROL and PREPARED message for each sent related to the
1146 CONTEXT_REPLY. If an operation fails, a CANCELLED message is sent instead of a
1147 PREPARED.

1148
1149 If the CONTEXT has “superior-type” of “atom”, then subsequent messages to the same
1150 Service, with the same related CONTEXT/atom, can have their associated operations put
1151 under the control of the same Inferior, and only a CONTEXT_REPLY/completed is sent back
1152 with the response (if the new operations fail, it will be necessary to send back
1153 CONTEXT_REPLY/repudiated, or send CANCELLED). If the “superior type” on the
1154 CONTEXT is “cohesive”, each operation will require separate enrolment.

1155
1156 Whether the “one-shot” mechanism is used is determined by the implementation on the
1157 responding (Inferior) side. This may be subject to configuration and may also be constrained
1158 by the application or by the binding in use.

1159

1160 **Extensibility**

1161

1162 To simplify interoperation between implementations of this edition of BTP with
1163 implementations of future editions, the “must-be-understood” sub-parameter as specified for
1164 Qualifiers may be defined for use with any parameter added to an existing message in a future
1165 revision of this specification. The default for “must-be-understood” shall be “true”, so an
1166 implementation receiving an unrecognised parameter without a “false” value for “must-be-
1167 understood” shall not accept it (the FAULT value “UnrecognisedParameter” is available, but
1168 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If
1169 “must-be-understood” with the value “false” is present as a sub-parameter of a parameter in
1170 any message, a receiving implementation **should** ignore the parameter.

1171

1172 How the sub-parameter is associated with the new parameter is determined by the particular
1173 binding.

1174

1175 No special mechanism is provided to allow for the introduction of completely new messages.

1176

1177 **Inferior handle**

1178

1179 ~~Some of the messages exchanged between a Terminator and a Decider are concerned with the~~
1180 ~~individual Inferiors enrolled with the Decider, and not with the business transaction as a~~

1181 whole. These messages distinguish the Inferiors of Decider using an “inferior handle”. This is
1182 created by the Decider and is unambiguous within the scope of the Decider.

1183
1184 The “inferior handle” is distinct from the “inferior identifier” passed on an ENROL message
1185 (among other places). The latter is created by the Inferior (or its enroller) and is required to be
1186 unambiguous within the scope of the address as inferior on the ENROL (and unambiguous
1187 within any of the individual addresses in that set of BTP addresses—the identifier must
1188 identify the Inferior across all the places it might migrate to or that have recovery
1189 responsibility for it).

1190
1191 The “inferior handle” is only used by the Terminator to refer to the inferiors of the Decider.
1192 In messages between the Decider and its Inferiors, the address as inferior and inferior
1193 identifier are used.

1195 Messages

1196

1197 Qualifiers

1198

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

1201

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

1202

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

1209

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

1212

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

1217

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

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

1229
 1230

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

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

1239
 1240

1240 **CONTEXT**

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

1248

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

1249
 1250

1251 **address-as-superior** the address to which ENROL and other messages from an
 1252 enrolled Inferior are to be sent. This can be a set of alternative addresses.

1253

1254 **superior identifier** identifies the Superior. This shall be globally unambiguous.
 1255 within the scope of the address-as-superior

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

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

1264
1265
1266 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
1267 timelimit” is carried by CONTEXT.

1268
1269 There is no target address parameter for CONTEXT as it is only transmitted in relation to the
1270 application messages, BEGIN and BEGUN.

1271
1272 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the
1273 superior type with the appropriate value.

1274
1275

CONTEXT_REPLY

1276
1277
1278 CONTEXT_REPLY is sent after receipt of CONTEXT (related to application message(s)) to
1279 indicate whether all necessary enrolments have already completed (ENROLLED has been
1280 received) or will be completed by ENROL messages sent in relation to the
1281 CONTEXT_REPLY or if an enrolment attempt has failed. CONTEXT_REPLY may be sent
1282 related to an application message (typically the response to the application message related to
1283 the CONTEXT). In some bindings the CONTEXT_REPLY may be implicit in the application
1284 message.

1285

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

1286
1287 **target-address** the address to which the CONTEXT_REPLY is sent. This shall
1288 be the “reply-address” from the CONTEXT.

1289
1290 ~~superior-address~~ one of the addresses from the address as superior from the
1291 CONTEXT. (The parameter is present in CONTEXT_REPLY to disambiguate
1292 the superior identifier.)

1293

1294 **superior identifier** the superior identifier from the CONTEXT

1295

1296 **completion_status:** reports whether all enrol operations made necessary by the
1297 receipt of the earlier CONTEXT message have completed. Values are

1298

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.

1299

1300 **qualifiers** standardised or other qualifiers.

1301

1302 The form CONTEXT_REPLY/completed, CONTEXT_REPLY/related and
1303 CONTEXT_REPLY/repudiated refer to CONTEXT_REPLY messages with status having the
1304 appropriate value. The form CONTEXT_REPLY/ok refers to either of
1305 CONTEXT_REPLY/completed or CONTEXT_REPLY/related.

1306

1307 If there are no necessary enrolments (e.g. the application messages related to the received
1308 CONTEXT did not require the enrolment of any Inferiors), then
1309 CONTEXT_REPLY/completed is used.

1310

1311 If a CONTEXT_REPLY/repudiated is received, the receiving implementation **must** ensure
1312 that the business transaction will not be confirmed.

1313

1314

1315 REQUEST_STATUS

1316

1317 Sent to an Inferior, Superior or to a Decider to ask it to reply with STATUS. The receiver
1318 may reject the request with a FAULT(StatusRefused).

1319

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

1320

1321 **target address** the address to which the REQUEST_STATUS message is sent.
1322 This can be any of address-as-decider, address-as-inferior or address-as-superior.

1323

1324 **reply address** the address to which the replying STATUS should be sent.

1325
1326 **target identifier** The identifier for the business transaction, or part of business
1327 transaction whose status is sought. If the target-address is an address-as-decider,
1328 this parameter shall be the “transaction-identifier” on the BEGUN message. If the
1329 target-address is an address-as-inferior, this parameter shall be the “inferior-
1330 identifier” on the ENROL message. If the target-address is a an address-as-
1331 superior, this parameter shall be the “superior-identifier” on the CONTEXT.

1332
1333 **qualifiers** standardised or other qualifiers.

1334
1335 Types of FAULT possible (sent to reply address)

1336
1337 *General*
1338 *StatusRefused* – if the receiver is not prepared to report its status to the
1339 sender of this message
1340 *UnknownTransaction* – if the target-identifier is unknown

1341
1342
1343 **STATUS**

1344
1345 Sent by a Inferior, Superior or Decider in reply to a REQUEST_STATUS, reporting the
1346 overall state of the transaction tree node represented by the sender.
1347

Parameter	Type
target address	BTP address
respondersaddress	BTP address
responders-identifier	Identifier
status	See below
qualifiers	List of qualifiers

1348
1349 **target address** the address to which the STATUS is sent. This will be the reply
1350 address on the REQUEST_STATUS message

1351
1352 ~~responders address~~ the address of the sender of the STATUS message—one of
1353 ~~address as inferior, address as decider, address as superior (with the responders-~~
1354 ~~identifier, this determines who the message is from).. If the sender has different~~
1355 ~~addresses as multiple roles (as Decider, Inferior or Superior), this shall be the~~
1356 ~~address on which the REQUEST_STATUS was received.~~

1357
1358 **responders-identifier** the identifier of the state, ~~identical to the “target-~~
1359 ~~identifier” on the REQUEST_STATUS, aligned with the responders address. If~~
1360 ~~the sender has multiple roles in the transaction (as Decider, Inferior or Superior),~~
1361 ~~this shall be the target identifier on the REQUEST_STATUS~~

1362
 1363
 1364
 1365
 1366
 1367

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 from all Inferiors	CANCELLED has been sent
<i>cancel-contradiction</i>	Not applicable	Autonomous cancel decision was made, CONFIRM received; CONTRADICTION has not been received
<i>confirm-contradiction</i>	Not applicable	Autonomous confirm decision was made, CANCEL received; CONTRADICTION has not been received

status value	Meaning from Superior	Meaning from Inferior
<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

1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378

qualifiers standardised or other qualifiers.

Types of FAULT possible

General

FAULT

Sent in reply to various messages to report an error condition

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

1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389

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

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

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

1390
1391
1392
1393
1394
1395

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

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

1396

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

1397

1398	<i>UnknownParameter</i>	A BTP message has been	Free text explanation
1399		received with an unrecognised	
1400	q	parameter	
1401	u		
1402	Qualifiers	standardised or other qualifiers.	
1403			

1404 Note – If the carrier mechanism used for the transmission of BTP messages
1405 is capable of delivering messages in a different order than they were sent in,
1406 the “WrongState” FAULT is not sent and should be ignored if received.

1407
1408 **REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES**

1409
1410 REQUEST_INFERIOR_STATUSES may be sent to and INFERIOR_STATUSES sent from
1411 any Decider, Superior or Inferior, asking it to report on the status of its relationships with
1412 Inferiors (if any). Since Deciders are required to respond to
1413 REQUEST_INFERIOR_STATUSES with INFERIOR_STATUSES but non-Deciders may
1414 just issue FAULT(StatusRefused), and INFERIOR_STATUSES is also used as a reply to
1415 other messages from Terminator to Decider, these messages are described below under the
1416 messages used in the control relationships.

1418 **Messages used in the outcome relationships**

1419
1420 **ENROL**

1421
1422 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a
1423 CONTEXT message in relation to an application request.
1424 The actor issuing ENROL plays the role of Enroller.

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

1426
1427 **target address** the address to which the ENROL is sent. This will be the
1428 address-as-superior from the CONTEXT message.
1429

1430 **superior identifier.** The superior identifier as on the CONTEXT message
 1431
 1432 **reply requested** true if an ENROLLED response is required, false otherwise.
 1433 Default is false.
 1434
 1435 **reply address** the address to which a replying ENROLLED is to be sent, if
 1436 “reply requested” is true. If this field is absent and “reply requested” is true, the
 1437 ENROLLED should be sent to the “address-as-inferior” (or one of them, at
 1438 sender’s option)
 1439
 1440 **address-as-inferior** the address to which PREPARE, CONFIRM, CANCEL and
 1441 SUPERIOR_STATE messages for this Inferior are to be sent.
 1442
 1443 **inferior identifier** an identifier that ~~unambiguously~~ identifies this Inferior. ~~This~~
 1444 ~~shall be globally unambiguous, within the scope of any of the address-as-inferior~~
 1445 ~~set of BTP addresses.~~
 1446
 1447 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
 1448 name” may be present.
 1449

1450 Types of FAULT possible (sent to Reply address)

1451 **General**

1452 **InvalidSuperior** – if superior identifier is unknown

1453 **DuplicateInferior** – if inferior with at least one of the set address-as-
 1454 inferior the same and the same inferior identifier is already enrolled

1455 **WrongState** – if it is too late to enrol new Inferiors (generally if the
 1456 Superior has already sent a PREPARED message to its superior or
 1457 terminator, or if it has already issued CONFIRM to other Inferiors).
 1458

1459
 1460 The form ENROL/rsp-req refers to an ENROL message with “reply requested” having the
 1461 value “true”; ENROL/no-rsp-req refers to an ENROL message with “reply requested” having
 1462 the value “false”
 1463

1464 ENROL/no-rsp-req is typically sent in relation to CONTEXT_REPLY/related. ENROL/rsp-
 1465 req is typically when CONTEXT_REPLY/completed will be used (after the ENROLLED
 1466 message has been received.)
 1467

1468 **ENROLLED**

1469
 1470 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been
 1471 successfully enrolled (and will therefore be included in the termination exchanges)
 1472

Parameter	Type
target address	BTP address

Parameter	Type
inferior identifier	Identifier
inferior handle	Handle
Qualifiers	List of qualifiers

1473

1474

1475

1476

1477

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)

1478

inferior identifier The inferior identifier as on the ENROL message

1479

1480

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

1481

1482

1483

1484

qualifiers standardised or other qualifiers.

1485

1486

No FAULT messages are issued on receiving ENROLLED.

1487

1488

1489

RESIGN

1490

1491

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.

1492

1493

1494

1495

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.

1496

1497

1498

1499

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

1500

1501

1502

1503

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

1504 **superior-identifier** The superior identifier as on the ENROL message
 1505
 1506 ~~**address-as-inferior** The address as inferior as on the earlier ENROL message~~
 1507 ~~(with the inferior identifier, this determines who the message is from)~~
 1508
 1509 **inferior-identifier** The inferior identifier as on the earlier ENROL message
 1510
 1511 **response-requested** is set to “true” if a RESIGNED response is required.
 1512
 1513 **qualifiers** standardised or other qualifiers.
 1514

1515 Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued
 1516 early.

1517
 1518 Types of FAULT possible (sent to address-as-inferior)

1519
 1520 *General*

1521 *InvalidSuperior* – if superior identifier is unknown

1522 *InvalidInferior* – if no ENROL had been received for this address-as-
 1523 inferior and identifier (Inferior Identity)

1524 *WrongState* – if a PREPARED or CANCELLED has already been
 1525 received by the Superior from this Inferior
 1526

1527 The form RESIGN/rsp-req refers to an RESIGN message with “reply requested” having the
 1528 value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “reply requested”
 1529 having the value “false”

1530
 1531
 1532 **RESIGNED**

1533
 1534 Sent in reply to a RESIGN/rsp-req message.
 1535

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

1536
 1537 **target address** the address to which the RESIGNED is sent. This will be the
 1538 address-as-inferior from the ENROL message.
 1539
 1540 **inferior identifier** The inferior identifier as on the earlier ENROL message for
 1541 this Inferior.
 1542
 1543 **qualifiers** standardised or other qualifiers.

1544
 1545 After receiving this message the Inferior will not receive any more messages with this
 1546 address-as-inferior and identifier.
 1547
 1548 No FAULT messages are issued on receiving RESIGNED.
 1549
 1550 **PREPARE**
 1551
 1552 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor
 1553 RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after
 1554 receiving a PREPARED message.
 1555
 1556

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

1557
 1558 **target address** the address to which the PREPARE message is sent. When sent
 1559 from Superior to Inferior, this will be the address-as-inferior from the ENROL
 1560 message.
 1561

1562 **inferior identifier** When sent from Superior to Inferior, the inferior identifier as
 1563 on the earlier ENROL message.
 1564
 1565

1566 **qualifiers** standardised or other qualifiers. The standard qualifier “Minimal
 1567 inferior timeout” is carried by PREPARE.
 1568
 1569

1570 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or
 1571 RESIGN.
 1572

1573 Types of FAULT possible (sent to Superior address)
 1574

General

1575 **InvalidInferior** – if inferior identifier is unknown, or an inferior-handle
 1576 on the inferiors-list is unknown
 1577

1578 **WrongState** – if a CONFIRM or CANCEL has already been received by
 1579 this Inferior.
 1580
 1581

1582 **PREPARED**
 1583

1584 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when
 1585 the Inferior has determined the operations associated with the Inferior can be confirmed and
 1586 can be cancelled, as may be instructed by the Superior. The level of isolation is a local matter
 1587 (i.e. it is the Inferiors choice, as constrained by the shared understanding of the application
 1588 exchanges) – other access may be blocked, may see applied results of operations or may see
 1589 the original state.
 1590

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

1591
 1592 **target address** the address to which the PREPARED is sent. This will be the
 1593 Superior address as on the ENROL message.
 1594

1595 ~~**superior identifier** When the message is sent from an Inferior to the Superior,~~
 1596 ~~the superior identifier as on the ENROL message~~

1597
 1598 ~~**address as inferior** When the message is sent from an Inferior to the Superior,~~
 1599 ~~the address as inferior as on the earlier ENROL message (with the inferior~~
 1600 ~~identifier, this determines who the message is from)~~

1601
 1602 **inferior identifier** The inferior identifier as on the ENROL message
 1603

1604 **default is cancel** if “true”, the Inferior states that if the outcome at the Superior
 1605 is to cancel the operations associated with this Inferior, no further messages need
 1606 be sent to the Inferior. If the Inferior does not receive a CONFIRM message, it
 1607 will cancel the associated operations. The value “true” will invariably be used
 1608 with a qualifier indicating under what circumstances (usually a timeout) an
 1609 autonomous decision to cancel will be made. If “false”, the Inferior will expect
 1610 a CONFIRM or CANCEL message as appropriate, even if qualifiers indicate that
 1611 an autonomous decision will be made.
 1612

1613 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
 1614 timeout” may be carried by PREPARED.
 1615

1616 On sending a PREPARED, the Inferior undertakes to maintain its ability to confirm or cancel
 1617 the effects of the associated operations until it receives a CONFIRM or CANCEL message.
 1618 Qualifiers may define a time limit or other constraints on this promise. The “default is

1619 cancel” parameter affects only the subsequent message exchanges and does not of itself state
1620 that cancellation will occur.

1621
1622 Types of FAULT possible (sent to address-as-inferior)

1623
1624 *General*
1625 *InvalidSuperior* – if Superior identifier is unknown
1626 *InvalidInferior* – if no ENROL has been received for this address-as-
1627 inferior and identifier, or if RESIGN has been received from this Inferior

1628
1629 The form PREPARED/cancel refers to a PREPARED message with “default is cancel” =
1630 “true”. The unqualified form PREPARED refers to a PREPARED message with “default is
1631 cancel” = “false”.

1632
1633

1634 CONFIRM

1635
1636 Sent by the Superior to an Inferior from whom PREPARED has been received.
1637

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

1638
1639 **target address** the address to which the CONFIRM message is sent. This will
1640 be the address-as-inferior from the ENROL message.

1641
1642 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1643 this Inferior.

1644
1645 **qualifiers** standardised or other qualifiers.

1646
1647 On receiving CONFIRM, the Inferior is released from its promise to be able to undo the
1648 operations of associated with the Inferior. The effects of the operations can be made available
1649 to everyone (if they weren’t already).

1650
1651 Types of FAULT possible (sent to Superior address)

1652
1653 *General*
1654 *InvalidInferior* – if inferior identifier is unknown
1655 *WrongState* – if no PREPARED has been sent by, or if CANCEL has
1656 been received by this Inferior.

1657
1658

1659 **CONFIRMED**

1660
1661
1662
1663
1664
1665

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
address-as-inferior	Set of BTP addresses
inferior identifier	Identifier
confirm received	Boolean
qualifiers	List of qualifiers

1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682

target address the address to which the CONFIRMED is sent. ~~When sent by an Inferior to a Superior, t~~**I**his will be the Superior address as on the CONTEXT message.

superior identifier ~~When the message is sent from an Inferior to the Superior, this shall be~~ the superior identifier as on the CONTEXT message.

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

inferior identifier ~~When the message is sent from an Inferior to the Superior, this shall be~~ the inferior identifier as on the earlier ENROL message.

1683
1684
1685
1686
1687
1688

confirm received “true” if CONFIRMED is sent after receiving a CONFIRM message; “false” if an autonomous confirm decision has been made and either if no CONFIRM message has been received or the implementation cannot determine if CONFIRM has been received (due to loss of state information in a failure).

1689

qualifiers standardised or other qualifiers.

1690
1691
1692
1693
1694

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

General

InvalidSuperior – if Superior identifier is unknown

1695 *InvalidInferior* – if no ENROL has been received for this address-as-
1696 inferior and identifier, or if RESIGN has been received from this Inferior.
1697

1698 Note – A CONFIRMED message arriving before a CONFIRM message is
1699 sent, or after a CANCEL has been sent will occur when the Inferior has
1700 taken an autonomous decision and is not regarded as occurring in the wrong
1701 state. (The latter will cause a CONTRADICTION message to be sent.)

1702 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm
1703 received” = “false”; CONFIRMED/response refers to a CONFIRMED message
1704 with “confirm received” = ”true”.
1705
1706

1707 CANCEL

1708 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.
1709
1710
1711

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

1712 **target address** the address to which the CANCEL message is sent. ~~When sent~~
1713 ~~from Superior to Inferior,~~ ~~t~~This will be the address-as-inferior from the ENROL
1714 message.
1715

1716 **inferior identifier** ~~When sent from Superior to Inferior,~~ the inferior identifier as
1717 on the earlier ENROL message.
1718
1719

1720 **qualifiers** standardised or other qualifiers.
1721

1722 When ~~sent to an~~received by an Inferior, the effects of any operations associated with the
1723 Inferior should be undone. If the Inferior had sent PREPARED, the Inferior is released from
1724 its promise to be able to confirm the operations.
1725

1726 Types of FAULT possible (sent to Superior address)
1727

1728 *General*

1729 *InvalidInferior* – if inferior identifier is unknown, or an inferior-handle
1730 on the inferiors-list is unknown

1731 *WrongState* – if a CONFIRM has been received by this Inferior.
1732
1733

1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753

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 address	BTP address
superior identifier	Identifier
address as inferior	Set of BTP address
inferior identifier	Identifier
qualifiers	List of qualifiers

1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771

target address the address to which the CANCELLED is sent. ~~When sent by an Inferior to a Superior, this will be the Superior address as on the CONTEXT message.~~

superior identifier ~~When the message is sent from an Inferior to the Superior, this shall be~~ the superior identifier as on the CONTEXT message.

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

inferior identifier ~~When the message is sent from an Inferior to the Superior, this shall be~~ the inferior identifier as on the earlier ENROL message.

qualifiers standardised or other qualifiers.

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

1772
1773
1774
1775
1776
1777
1778

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

1779
1780
1781
1782

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.)

1783
1784
1785
1786
1787
1788
1789
1790

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.

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

1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806

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

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

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

qualifiers standardised or other qualifiers.

1807 CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom
1808 PREPARED has been received (subject to the requirement that there is only one enrolled
1809 Inferior).

1810
1811 Types of FAULT possible (sent to Superior address)

1812
1813 *General*
1814 *InvalidInferior* – if inferior identifier is unknown
1815 *WrongState* – if a PREPARE has already been ~~received from~~ sent to this
1816 Inferior

1817
1818 **HAZARD**

1819
1820 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly
1821 and consistently cancel or confirm the operations in accord with the decision (either the
1822 received decision of the superior or its own autonomous decision), or when the Inferior is
1823 unable to determine that a “mixed” condition has not occurred.

1824
1825 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there
1826 is a mixed condition within its associated operations or is unable to determine that there is not
1827 a mixed condition.
1828

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

1829
1830 **target address** the address to which the HAZARD is sent. This will be the
1831 superior address from the ENROL message.

1832
1833 **superior identifier** The superior identifier as ~~used~~ on the ENROL message

1834
1835 ~~address as inferior~~ ~~The address as inferior as on the earlier ENROL message~~
1836 ~~(with the inferior identifier, this determines who the message is from)~~

1837
1838 **inferior identifier** The inferior identifier as on the earlier ENROL message

1839
1840 **level** indicates, with value “mixed” that a mixed condition has definitely
1841 occurred; or, with value “possible” that it is unable to determine whether a mixed
1842 condition has occurred or not.

1843
1844 **qualifiers** standardised or other qualifiers.

1845
1846 Types of FAULT possible (sent to address-as-inferior)

1847
1848 **General**
1849 **InvalidSuperior** – if Superior identifier is unknown
1850 **InvalidInferior** – if no ENROL has been received for this address-as-
1851 inferior and identifier, or if RESIGN has been received from this Inferior

1852
1853
1854 The form HAZARD/mixed refers to a HAZARD message with “level” = “mixed”, the form
1855 HAZARD/possible refers to a HAZARD message with “level” = “possible”.

1856
1857 **CONTRADICTION**

1858
1859 Sent by the Superior to an Inferior that has taken an autonomous decision contrary to the
1860 decision for the atom. This is detected by the Superior when the ‘wrong’ one of
1861 CONFIRMED or CANCELLED is received. CONTRADICTION is also sent in response to a
1862 HAZARD message.
1863

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

1864
1865 **target address** the address to which the CONTRADICTION message is sent.
1866 This will be the address-as-inferior from the ENROL message.

1867
1868 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1869 this Inferior.

1870
1871 **qualifiers** standardised or other qualifiers.

1872
1873 Types of FAULT possible (sent to Superior address)

1874
1875 **General**
1876 **InvalidInferior** – if inferior identifier is unknown
1877 **WrongState** – if neither CONFIRMED or CANCELLED has been sent
1878 by this Inferior

1879
1880 **SUPERIOR_STATE**

1881
1882 Sent by a Superior as a query to an Inferior when
1883

1884
1885
1886
1887
1888
1889
1890
1891

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

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

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

1892
1893
1894
1895
1896
1897
1898
1899
1900
1901

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

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

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

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

1902
1903
1904
1905
1906

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

1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939

qualifiers standardised or other qualifiers.

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

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

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

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

INFERIOR_STATE

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

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

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

1940

1941 **target address** the address to which the INFERIOR_STATE is sent. This will
1942 be the target address as used the original ENROL message.

1943
1944 **superior identifier** The superior identifier as used on the ENROL message
1945

1946 ~~address as inferior~~ ~~The address as inferior as on the ENROL message (with the~~
1947 ~~inferior identifier, this determines who the message is from)~~

1948
1949 **inferior identifier** The inferior identifier as on the ENROL message
1950

1951 **status** states the current state of the Inferior for the atomic business transaction,
1952 which corresponds to the last message sent to the Superior by (or in the case of
1953 ENROL for) the Inferior
1954

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

1955
1956 **reply requested** “true” if INFERIOR_STATE is sent as a query at the
1957 Superior’s initiative; “false” if INFERIOR_STATE is sent in reply to a received
1958 SUPERIOR_STATE or other message. Can only be “true” if “status” is “active”
1959 or “prepared-received”. Can only be “true” if “status” is “active”.
1960

1961 **qualifiers** standardised or other qualifiers.
1962

1963 The Superior, on receiving INFERIOR_STATE with “reply requested” = “true”, should reply
1964 in a timely manner by (depending on its state) repeating the previous message it sent or by
1965 sending SUPERIOR_STATE with the appropriate status value.
1966

1967 A status of “unknown” shall only be sent if it has been determined for certain that the Inferior
1968 has no knowledge of a relationship with the Superior. If there could be persistent information
1969 corresponding to the Superior, but it is not accessible from the entity receiving an
1970 SUPERIOR_STATE/*y (or other) message targetted on the Inferior or the entity cannot
1971 determine whether any such persistent information exists, the response shall be
1972 “inaccessible”.
1973

1974 INFERIOR_STATE/unknown is also used as a response to messages, other than
1975 SUPERIOR_STATE/*y that are received when the Inferior is not known (and it is known
1976 there is no state information for it).

1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999

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

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

REDIRECT

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

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

2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013

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

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

inferior identifier The inferior identifier as on the ENROL message

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

2014 **new address** The (set of alternatives) new addresses to be used for messages
 2015 sent to this entity.
 2016
 2017 **qualifiers** standardised or other qualifiers.
 2018
 2019 If the actor whose address is changed is an Inferior, the new address value
 2020 replaces the address-as-inferior as present in the ENROL.
 2021
 2022 If the actor whose address is changed is a Superior, the new address value
 2023 replaces the Superior address as present in the CONTEXT message (or as present
 2024 in any other mechanism used to establish the Superior:Inferior relationship).
 2025
 2026
 2027

2028 Messages used in control relationships

2029 BEGIN

2030 A request to a Factory to create a new Business Transaction. This may either be a new top-
 2031 level transaction, in which case the Composer or Coordinator will be the Decider, or the new
 2032 Business Transaction may be immediately made the Inferior within an existing Business
 2033 Transaction (thus creating a sub-Composer or sub-Coordinator).
 2034
 2035
 2036

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

2037
 2038 **target address** the address of the entity to which the BEGIN is sent. How this
 2039 address is acquired and the nature of the entity are outside the scope of this
 2040 specification.
 2041
 2042 **reply address** the address to which the replying BEGUN and related
 2043 CONTEXT message should be sent.
 2044
 2045 **transaction type** identifies whether a new Cohesion or new Atom is to be
 2046 created; this value will be the “superior type” in the new CONTEXT
 2047
 2048 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
 2049 timelimit” may be present on BEGIN, to set the timelimit for the new business
 2050 transaction and will be copied to the new CONTEXT. The standard qualifier
 2051 “Inferior name” may be present if there is a CONTEXT related to the BEGIN.
 2052

2053 A new top-level Business Transaction is created if there is no CONTEXT related to the
 2054 BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is
 2055 created if the CONTEXT message for the existing Business Transaction is related to the
 2056 BEGIN. In this case, the Factory is responsible for enrolling the new Composer or
 2057 Coordinator as an Inferior of the Superior identified in that CONTEXT.
 2058

2059 Note – This specification does not provide a standardised means to
 2060 determine which of the Inferiors of a sub-Composer are in its confirm set.
 2061 This is considered part of the application:inferior relationship.

2062 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction type” having
 2063 the corresponding value.
 2064

2065 Types of FAULT possible (sent to Reply address)
 2066

2067 General

2068 **BEGUN**

2069
 2070
 2071
 2072 BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT
 2073 for the new business transaction.
 2074

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

2075
 2076 **target address** the address to which the BEGUN is sent. This will be the reply
 2077 address from the BEGIN.
 2078

2079 **address-as-decider** for a top-~~most~~level transaction (no CONTEXT related to
 2080 the BEGIN), this is the address to which PREPARE_INFERIORS,
 2081 CONFIRM_TRANSACTION, CANCEL_TRANSACTION,
 2082 CANCEL_INFERIORS and REQUEST_INFERIOR_STATUSES messages are
 2083 to be sent; if a CONTEXT was related to the BEGIN this parameter is absent
 2084

2085 address-as-inferior for a non-top-most transaction (a CONTEXT was related to
 2086 the BEGIN), this is the address-as-inferior used in the enrolment with the
 2087 Superior identified by the CONTEXT related to the BEGIN. The parameter is

2088 optional (implementor's choice) if this is not a top-most transaction; it shall be
2089 absent if this is a top-most transaction this parameter.

2090
2091 **transaction-identifier** if this is a top-most transaction, this is an globally-
2092 unambiguous identifier for identifies the new Decider (Composer or Coordinator)
2093 within the scope of the address-as-decider. If this is not a top-level-most
2094 transaction, the transaction-identifier is optional, but if present shall be the
2095 inferior-identifier used in the enrolment with the Superior identified by the
2096 CONTEXT related to the BEGIN.

2098 Note – The “transaction-identifier” may be identical to the “superior-
2099 identifier” in the CONTEXT that is related to the BEGUN

2100
2101 ~~**inferior handle** Shall be absent if this is a top level transaction and may or may~~
2102 ~~not be present otherwise. (Presence or absence will be determined by the nature~~
2103 ~~of the Superior identified in the CONTEXT related to the BEGIN). If present, the~~
2104 ~~inferior handle will identify this new business transaction as in the inferiors-list~~
2105 ~~parameters in messages between the Superior identified in the CONTEXT related~~
2106 ~~to the BEGIN (acting as a Decider) and its Terminator. The value shall be~~
2107 ~~different for each enrolled Inferior of that Superior.~~

2108
2109 ~~**address-as-inferior** This parameter shall be absent if this is a top level~~
2110 ~~transaction and may be present, at implementation option otherwise. If present, it~~
2111 ~~shall be the address-as-inferior used in the enrolment with the Superior identified~~
2112 ~~by the CONTEXT related to the BEGIN. If this is a top level transaction~~

2113
2114 **qualifiers** standardised or other qualifiers.

2115
2116 At implementation option, the “address-as-decider” and/or “address-as-inferior” and the
2117 “address-as-superior” in the related CONTEXT may be the same or may be different. There
2118 is no general requirement that they even use the same bindings. Any may also be the same as
2119 the target address of the BEGIN message (the ~~inferior~~ identifier on messages will ensure they
2120 are applied to the appropriate Composer or Coordinator).

2121
2122 No FAULT messages are issued on receiving BEGUN.

2123 **PREPARE_INFERIORS**

2124
2125
2126 Sent from a Terminator to a Decider, but only if it is a Cohesion Composer, to tell it to
2127 prepare all or some of its inferiors, by sending PREPARE to any that have not already sent
2128 PREPARED, RESIGN or CANCELLED to the Decider (Composer) on its relationships as
2129 Superior. If the inferiors-list parameter is absent, the request applies to all the inferiors; if the
2130 parameter is present, it applies only to the identified inferiors of the Decider (Composer).

2131

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

2132

2133

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

2134

2135

2136

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

2137

2138

2139

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

2140

2141

2142

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

2143

2144

2145

2146

2147

qualifiers standardised or other qualifiers.

2148

2149

2150

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

2151

2152

2153

2154

2155

2156

2157

Types of FAULT possible (sent to Superior address)

2158

2159

General

2160

InvalidDecider – if Decider address is unknown

2161

UnknownTransaction – if the transaction-identifier is unknown

2162

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

2163

WrongState – if a CONFIRM_TRANSACTION or

2164

CANCEL_TRANSACTION has already been received by this

2165

Composer.

2166

2167

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.

2168

2169

2170
2171
2172
2173
2174
2175
2176
2177
2178

CONFIRM_TRANSACTION

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

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

2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205

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.

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

2206 “confirm-set” shall be all remaining Inferiors. If the business transaction is an Atom, the
2207 “confirm-set” is automatically all the Inferiors.

2208
2209 Any Inferiors from which RESIGN is received are not counted in the confirm-set.

2210
2211 If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED
2212 has not been received, PREPARE shall be issued to that Inferior.

2213

2214 NOTE -- If PREPARE has been sent but PREPARED not yet received from
2215 an Inferior in the confirm-set, it is an implementation option whether and
2216 when to re-send PREPARE. The Superior implementation may choose to re-
2217 send PREPARE if there are indications that the earlier PREPARE was not
2218 delivered.

2219

2220

2221 A confirm decision may be made only if PREPARED has been received from all Inferiors in
2222 the “confirm-set”. The making of the decision shall be persistent (and if it is not possible to
2223 persist the decision, it is not made). If there is only one remaining Inferior in the “confirm
2224 set” and PREPARE has not been sent to it, CONFIRM_ONE_PHASE may be sent to it.

2225

2226 All remaining Inferiors that are not in the confirm set shall be cancelled.

2227

2228 If a confirm decision is made and “report-hazard” was “false”, a CONFIRM_COMPLETE
2229 message shall be sent to the “reply-address”.

2230

2231 If a cancel decision is made and “report-hazard” was “false”, a CANCEL_COMPLETE
2232 message shall be sent to the “reply-address”.

2233

2234 If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.
2235 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in
2236 the confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent
2237 to the “reply-address”.

2238

2239 Types of FAULT possible (sent to reply address)

2240

2241

General

2242

InvalidDecider – if Decider address is unknown

2243

UnknownTransaction – if the transaction-identifier is unknown

2244

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

2245

WrongState – if a CANCEL_TRANSACTION has already been
received .

2246

2247

2248

2249

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

2250 CONFIRM_TRANSACTION/specific refers to a CONFIRM_TRANSACTION message
2251 where the “inferiors-list” parameter is present.

2252

TRANSACTION_CONFIRMED

2253

2254

2255

A Decider sends TRANSACTION_CONFIRMED to a Terminator in reply to
2256 CONFIRM_TRANSACTION if all of the confirm-set confirms (and, for a Cohesion, all other
2257 Inferiors cancel) without reporting hazards, or if the Decider made a confirm decision and the
2258 CONFIRM_TRANSACTION had a “report-hazards” value of “false”.

2259

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

2260

2261

target address the address to which the TRANSACTION_CONFIRMED is
2262 sent., this will be the reply address from the CONFIRM_TRANSACTION
2263 message.

2264

2265

~~**address-as-decider** the address as decider of the Decider as on the BEGUN
2266 message (with the transaction identifier, this determines who the message is
2267 from).~~

2268

2269

transaction identifier the transaction identifier as on the BEGUN message (i.e.
2270 the identifier of the Decider as a whole).

2271

2272

qualifiers standardised or other qualifiers.

2273

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

2274

2275

General

2276

InvalidTerminator – if Terminator address is unknown

2277

UnknownTransaction – if the transaction-identifier is unknown

2278

2279

CANCEL_TRANSACTION

2280

2281

Sent by a Terminator to a Decider at any time before CONFIRM_TRANSACTION has been
2282 sent.

2283

2284

Parameter	Type
target address	BTP address
reply address	BTP address

transaction identifier	Identifier
report-hazard	Boolean
qualifiers	List of qualifiers

2285
2286
2287
2288
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
2315
2316
2317
2318
2319
2320

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

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

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

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

qualifiers standardised or other qualifiers.

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

Types of FAULT possible (sent to Superior address)

General

InvalidDecider – if Decider address is unknown

UnknownTransaction – if the transaction-identifier is unknown

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

CANCEL_INFERIORS

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

Parameter	Type
target address	BTP address
reply address	BTP address
transaction identifier	Identifier

inferiors-list List of **inferior handles** **identifiers**
qualifiers List of qualifiers

2321

2322

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

2324

2325

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

2326

2327

2328

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

2329

2330

2331

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

2332

2333

2334

2335

qualifiers standardised or other qualifiers.

2336

2337

2338

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

2339

2340

2341

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

2342

2343

2344

Types of FAULT possible (sent to Superior address)

2345

2346

2347

General

2348

InvalidDecider – if Decider address is unknown

2349

UnknownTransaction – if the transaction-identifier is unknown

2350

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

2351

WrongState – if a CONFIRM_TRANSACTION or

2352

CANCEL_TRANSACTION has been received by this Composer.

2353

2354

2355

2356

TRANSACTION_CANCELLED

2357

2358

A Decider sends TRANSACTION_CANCELLED to a Terminator in reply to **REQUEST_CANCEL_TRANSACTION** or in reply to CONFIRM_TRANSACTION if the Decider decided to cancel. In both cases, TRANSACTION_CANCELLED is used only if all

2359

2360

2361 Inferiors cancelled without reporting hazards or the CANCEL_TRANSACTION or
2362 CONFIRM_TRANSACTION had a “report-hazard” value of “false.”
2363

Parameter

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

2364
2365 **target address** the address to which the TRANSACTION_CANCELLED is
2366 sent. This will be the reply address from the CANCEL_TRANSACTION or
2367 CONFIRM_TRANSACTION message.
2368

2369 ~~address-as-decider~~ ~~the address-as-decider of the Decider as on the BEGUN~~
2370 ~~message (with the transaction-identifier, this determines who the message is~~
2371 ~~from).~~
2372

2373 **transaction identifier** the transaction identifier as on the BEGUN message (i.e.
2374 the identifier of the Decider as a whole).
2375

2376 **qualifiers** standardised or other qualifiers.
2377

2378 Types of FAULT possible (sent to address-as-decider)
2379

General

2381 *InvalidTerminator* – if Terminator address is unknown

2382 *UnknownTransaction* – if the transaction-identifier is unknown
2383

2384
2385

2386 **REQUEST_INFERIOR_STATUSES**
2387

2388 Sent to a Decider to ask it to report the status of its Inferiors with an INFERIOR_STATUSES
2389 message. It can also be sent to any actor with an address-as-superior or address-as-inferior,
2390 asking it about the status of that transaction tree nodes Inferiors, if there are any. In this latter
2391 case, the receiver may reject the request with a FAULT(StatusRefused). If it is prepared to
2392 reply, but has no Inferiors, it replies with an INFERIOR_STATUSES with an empty “status-
2393 list” parameter.
2394

Parameter

Type

target address	BTP address
reply address	BTP address
target-identifier	Identifier

inferiors-list List of [inferior handles](#) [Identifiers](#)
Qualifiers List of qualifiers

2395

2396

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.

2397

2398

2399

2400

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

2401

2402

2403

2404

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

2405

2406

2407

2408

2409

inferiors-list defines which inferiors enrolled with the target are to be included in the INFERIOR_STATUSES, [using the “inferior-identifiers” as on the ENROL received by the Decider \(in its role as Superior\)](#). If the list is absent, the status of all enrolled [inferiors](#) will be reported.

2410

2411

2412

2413

2414

qualifiers standardised or other qualifiers.

2415

2416

Types of FAULT possible (sent to reply-address)

2417

2418

2419

General

2420

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

2421

2422

UnknownTransaction – if the transaction-identifier is unknown

2423

2424

2425

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

2426

2427

2428

2429

INFERIOR_STATUSES

2430

2431

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

2432

2433

2434

2435

2436

2437

2438

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

2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454

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

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

~~**responders-identifier** If the sender is a Decider, the transaction identifier as on the BEGUN message. Otherwise, the target-identifier used on the REQUEST_INFERIOR_STATUSES.~~

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

Field	Type
Inferior- handle <u>identifier</u>	Inferior- handle <u>identifier</u> , 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).

2455
2456
2457
2458

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

status value	Meaning
<i>prepared</i>	PREPARED has been received
<i>autonomously confirmed</i>	CONFIRMED/auto has been received, no completion message has been sent
<i>autonomously cancelled</i>	PREPARED had been received, and since then CANCELLED has been received but no completion message has been sent
<i>confirming</i>	CONFIRM has been sent, no outcome reply has been received
<i>confirmed</i>	CONFIRMED/response has been received
<i>cancelling</i>	CANCEL has been sent, no outcome reply has been received
<i>cancelled</i>	CANCELLED has been received, and PREPARED was not received previously
<i>cancel-contradiction</i>	Confirm had been ordered (and may have been sent), but CANCELLED was received
<i>confirm-contradiction</i>	Cancel had been ordered (and may have been sent) but CONFIRM/auto was received
<i>hazard</i>	A HAZARD message has been received
<i>invalid</i>	No such inferior is enrolled (used only in reply to a REQUEST_INFERIOR_STATUSES/specific)

2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478

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

2479 **Groups – combinations of related messages**

2480

2481

2482

2483

2484

2485

2486

2487

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.

2488

CONTEXT & application message

2489

2490

2491

2492

2493

2494

2495

2496

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.

2497

2498

2499

2500

2501

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

2502

2503

2504

2505

2506

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

2507

2508

2509

2510

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

2511

2512

2513

2514

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

2515

2516

2517

2518

2519

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

2520

2521

2522

CONTEXT_REPLY & ENROL

2523 **Meaning:** the enrolment of the Inferior identified in the ENROL is to be performed with
2524 the Superior identified in the CONTEXT message this CONTEXT_REPLY is replying
2525 to. If the “completion-status” of CONTEXT_REPLY is “related”, failure of this
2526 enrolment shall prevent the confirmation of the business transaction.

2527

2528 **Target address:** the target address is that of the CONTEXT_REPLY. This will be the
2529 reply address of the CONTEXT message (in many cases, including request/reply
2530 application exchanges, this address will usually be implicit).

2531

2532 The target address of the ENROL message is omitted.

2533

2534 The actor receiving the related group will use the retained Superior address from the
2535 CONTEXT sent earlier to forward the ENROL. When doing so, it changes the ENROL to
2536 ask for a response (if it was an ENROL/no-rsp-req) and supplies its own address as the
2537 “reply-address”, remembering the original “reply-address” if there was one.

2538

2539 If ENROLLED is received and the original received ENROL was ENROL/rsp-req, the
2540 ENROLLED is forwarded back to the original “reply-address”.

2541

2542 If this attempt fails (i.e. ENROLLED is not received), and the “completion-status” of the
2543 CONTEXT_REPLY was “related”, the actor is required to ensure that the Superior does
2544 not proceed to confirmation. How this is achieved is an implementation option, but must
2545 take account of the possibility that direct communication with the Superior may fail. (One
2546 method is to prevent CONFIRM_TRANSACTION being sent to the Superior (in its role
2547 as Decider); another is to enrol as another Inferior before sending the original CONTEXT
2548 out with an application message). If the Superior is a sub-coordinator or sub-composer,
2549 an enrolment failure must ensure the sub-coordinator does not send PREPARED to its
2550 own Superior.

2551

2552 If the actor receiving the related group is also the Superior (i.e. it has the same binding
2553 address), the explicit forwarding of the ENROL is not required, but the resultant effect –
2554 that if enrolment fails the Superior does not confirm or issue PREPARED – shall be the
2555 same.

2556

2557 A CONTEXT_REPLY & ENROL group may contain multiple ENROL messages, for
2558 several Inferiors. Each ENROL shall be forwarded and an ENROLLED reply received
2559 before the Superior is allowed to confirm if the “completion-status” in the
2560 CONTEXT_REPLY was “related”.

2561

2562 When the group is constructed, if the CONTEXT had “superior-type” value of “atom”,
2563 the “completion-status” of the CONTEXT_REPLY shall be “related”. If the “superior-
2564 type” was “cohesive”, the “completion-status” shall be “completed” or “related” (as
2565 required by the application). If the value is “completed”, the actor receiving the group
2566 shall forward the ENROLs, but is not required to (though it may) prevent confirmation.

2567

2568 **CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED**

2569

2570 This combination is characterised by a related CONTEXT_REPLY and either or both of
2571 PREPARED and CANCELLED, with or without ENROL.

2572
2573 **Meaning:** If ENROL is present, the meaning and required processing is the same as for
2574 CONTEXT_REPLY & ENROL. The PREPARED or CANCELLED message(s) are
2575 forwarded to the Superior identified in the CONTEXT message this CONTEXT_REPLY
2576 is replying to.
2577

2578 Note – the combination of CONTEXT_REPLY & ENROL & CANCELLED
2579 may be used to force cancellation of an atom

2580
2581 **Target address:** the target address is that of the CONTEXT_REPLY. This will be the
2582 reply address of the CONTEXT message (in many cases, including request/reply
2583 application exchanges, this address will usually be implicit).
2584

2585 The target address of the PREPARED and CANCELLED message is omitted – they will
2586 be sent to the Superior identified in the earlier CONTEXT message.
2587

2588 The actor receiving the group forwards the PREPARED or CANCELLED message to the
2589 Superior in as for an ENROL, using the retained Superior address from the CONTEXT
2590 sent earlier, except there is no reply required from the Superior.
2591

2592 If (as is usual) an ENROL and PREPARED or CANCELLED message are for the same
2593 Inferior, the ENROL shall be sent first, but the actor need not wait for the ENROLLED to
2594 come back before sending the PREPARED or CANCELLED (so an
2595 ENROL+PREPARED bundle from this actor to the Superior could be used).
2596

2597 The group can contain multiple ENROL, PREPARED and CANCELLED messages.
2598 Each PREPARED and CANCELLED message will be for a different Inferior.. There is
2599 no constraint on the order of their forwarding, except that ENROL and PREPARED or
2600 CANCELLED for the same Inferior shall be delivered to the Superior in the order
2601 ENROL first, followed by the other message for that Inferior.
2602

2603
2604

2605 **CONTEXT_REPLY & ENROL & application message (& PREPARED)**

2606
2607 This combination is characterised by a related CONTEXT_REPLY, ENROL and an
2608 application message. PREPARED may or may not be present in the related group.
2609

2610 **Meaning:** the relation between the BTP messages is as for the preceding groups, The
2611 transmission of the application message (and application effects implied by its
2612 transmission) has been associated with the Inferior identified by the ENROL and will be
2613 subject to the outcome delivered to that Inferior.
2614

2615 **Target address:** the target address of the group is the target address of the
2616 CONTEXT_REPLY which shall also be the target address of the application message.
2617 The ENROL and PREPARED messages do not contain their target addresses.
2618
2619 The processing of ENROL and PREPARED messages is the same as for the previous
2620 groups.
2621
2622 This group can be used when participation in business transaction (normally a cohesion),
2623 is initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with
2624 some associated application semantic, performs some work for the transaction and sends
2625 an application message with a related ENROL. The CONTEXT_REPLY allows the
2626 addressing of the application (and the CONTEXT_REPLY) to be distinct from that of the
2627 Superior.
2628
2629 The actor receiving the group may associate the “inferior-~~handle~~identifier” received on
2630 the ENROL-~~LED~~ with the application message in a manner that is visible to the
2631 application receiving the message (e.g. for subsequent use in Terminator:Decider
2632 exchanges).

2634 **BEGUN & CONTEXT**

2635
2636 **Meaning:** the CONTEXT is that for the new business transaction, containing the
2637 Superior address.
2638
2639 **Target address:** the target address is that of the BEGUN message – this will be the reply
2640 address of the earlier BEGIN message.

2641 **BEGIN & CONTEXT**

2642
2643
2644 **Meaning:** the new business transaction is to be an Inferior (sub-coordinator or sub-
2645 composer) of the Superior identified by the CONTEXT. The Factory (receiver of the
2646 BEGIN) will perform the enrolment.
2647
2648 **Target address:** the target address is that of the BEGIN – this will be the address of the
2649 Factory.
2650

2651 **Standard qualifiers**

2652
2653 The following qualifiers are expected to be of general use to many applications and
2654 environments. The URI “urn:oasis:names:tc:BTP:qualifiers” is used in the
2655 Qualifier group value for the qualifiers defined here.
2656

2657 **Transaction timelimit**

2658
2659
2660 The transaction timelimit allows the Superior (or an application element initiating the
2661 business transaction) to indicate the expected length of the active phase, and thus give an

2662 indication to the Inferior of when it would be appropriate to initiate cancellation if the active
2663 phase appears to continue too long. The time limit ends (the clock stops) when the Inferior
2664 decides to be prepared and issues PREPARED to the Superior.

2665
2666 It should be noted that the expiry of the time limit does not change the permissible actions of
2667 the Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is
2668 **permitted** to initiate cancellation for internal reasons. The timelimit gives an indication to the
2669 entity of when it will be useful to exercise this right.

2670
2671 The qualifier is propagated on a CONTEXT message.

2672
2673 The “Qualifier name” shall be “transaction-timelimit”.

2674
2675 The “Content” shall contain the following field:

Content field	Type
Timelimit	Integer

2677
2678 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
2679 time of transmission of the containing CONTEXT, of the active phase of the business
2680 transaction.

2681 2682 **Inferior timeout**

2683
2684 This qualifier allows an Inferior to limit the duration of its “promise”, when sending
2685 PREPARED, that it will maintain the ability to confirm or cancel the effects of all associated
2686 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or
2687 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and
2688 can apply the decision indicated in the qualifier.

2689
2690 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply
2691 a confirm or cancel decision before the CONFIRM or CANCEL is received and before this
2692 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,
2693 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
2694 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
2695 expiry of this timeout, is liable to cause contradictory decisions across the business
2696 transaction. BTP ensures that at least the occurrence of such a contradiction will be
2697 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic
2698 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
2699 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
2700 change in the probability that it will.

2701
2702 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
2703 intended decision, only that is at liberty to do so. An implementation may choose to only
2704 apply the decision if there is contention for the underlying resource, for example.
2705 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for

2706 the business transaction are made before these timeouts expire (and allow a margin of error
2707 for network latency etc.).

2708
2709 The qualifier may be present on a PREPARED message. If the PREPARED message has the
2710 “default is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall
2711 have the value “cancel”.

2712
2713 The “Qualifier name” shall be “inferior-timeout”.

2714
2715 The “Content” shall contain the following fields:
2716

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

2717
2718 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
2719 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
2720 effects of the associated operations, as ordered by the receiving Superior.

2721
2722 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an
2723 autonomous decision is made.

2724
2725 **Minimum inferior timeout**

2726
2727 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the
2728 Inferior. If a Superior knows that the decision for the business transaction will not be
2729 determined for some period, it can require that Inferiors do not send PREPARED messages
2730 with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to
2731 send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with
2732 CANCELLED.

2733
2734 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If
2735 present on more than one, and with different values of the MinimumTimeout field, the value
2736 on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall
2737 prevail over either of the others.

2738
2739 The “Qualifier name” shall be “minimum-inferior-timeout”.

2740
2741 The “Content” shall contain the following field:
2742

Content field	Type
MinimumTimeout	Integer

2743
2744 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be
2745 acceptable in the Inferior timeout qualifier on an answering PREPARED message.

2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775

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-handle~~” 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-~~handle~~identifier” on ENROLLED and BEGUN; ~~which is required to be unambiguous within the scope of the Decider~~). 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.)

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

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

The “Content” shall contain the following fields:

Content field	Type
inferior-name	String

2776
2777
2778

Inferior name the name assigned to the enrolling Inferior.

2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824

State Tables

Explanation of the state tables

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

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

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

Status queries

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

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

Decision events

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

2825 business transaction and on features of the implementation (e.g. making of a persistent record
2826 of the decision means that the information will survive at least some failures that otherwise
2827 lose state information, but the level of survival depends on the purpose of the
2828 implementation). [Table 2](#) and [Table 3](#) define the decision events.

2829
2830 In some cases, an implementation may not need to make an active change to have a persistent
2831 record of a decision, provided that the implementation will restore itself to the appropriate
2832 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,
2833 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via
2834 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a
2835 record of “decide to cancel autonomously”, provided it always updated such a record as part
2836 of the “apply ordered confirmation” decision event.

2837
2838 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
2839 PREPARE indicates that the application exchange (to associate operations with the Inferior)
2840 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
2841 state corresponding to an incomplete application exchange. However, implementations are
2842 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
2843 that experiences failure after sending PREPARE may, on recovery, have no information
2844 about the transaction, in which case it is considered to be in the completed state (Z), which
2845 will imply the cancellation of the Inferior and its associated operations.

2846
2847 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its
2848 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a
2849 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
2850 persistent information (which would combine both superior and inferior information, pointing
2851 both up and down the tree).

2852
2853

2854 **Disruptions – failure events**

2855
2856 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or
2857 may) cause a change of state. The disruption events in the state tables model different extents
2858 of loss of state information. An implementation is not required to exhibit all the possible
2859 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
2860 possible disruption.

2861
2862 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,
2863 which involves possible interruption of service and loss of messages in transit, but no change
2864 of state (either because no state information was lost, or because recovery from persistent
2865 information restores the implementation to the same state). The “disruption 0” event would
2866 typically be an appropriate abstraction for a communication failure.

2867
2868

2868 **Invalid cells and assumptions of the communication mechanism**

2869
2870 The empty cells in state table represent events that cannot happen. For events corresponding
2871 to sending a message or any of the decision events, this prohibition is absolute – e.g. a

2872 conformant implementation in the Superior active state “B1” will not send CONFIRM. For
2873 events corresponding to receiving a message, the interpretation depends on the properties of
2874 the underlying communications mechanism.

2875

2876 For all communication mechanisms, it is assumed that

- 2877 a) the two directions of the Superior:Inferior communication are not synchronised –
- 2878 that is messages travelling in opposite directions can cross each other to any
- 2879 degree; any number of messages may be in transit in either direction; and
- 2880 b) messages may be lost arbitrarily

2881

2882 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered
2883 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a
2884 state where the corresponding cell is empty indicates that the far-side has sent a message out
2885 of order – a FAULT message with the Fault Type “WrongState” can be returned.

2886

2887 If the communication mechanisms cannot guarantee ordered delivery, then messages received
2888 where the corresponding cell is empty should be ignored. Assuming the far-side is
2889 conformant, these messages can assumed to be “stale” and have been overtaken by messages
2890 sent later but already delivered. (If the far-side is non-conformant, there is a problem
2891 anyway).

2892

2893 **Meaning of state table events**

2894

2895 The tables in this section define the events (rows) in the state tables. [Table 1](#) defines
2896 the events corresponding to sending or receiving BTP messages and the disruption events.
2897 [Table 2](#) describes the decision events for an Inferior, [Table 3](#) those for a
2898 Superior.

2899

2900 The decision events for a Superior, defined in [Table 3](#) cannot be specified without
2901 reference to other Inferiors to which it is Superior and to its relation with the application or
2902 other entity that (acting ultimately on behalf of the application) drives it.

2903

2904 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and
2905 which are to be treated as an atomic decision unit with (and thus including) the Inferior on
2906 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior
2907 type” of “atom”, this will be all Inferiors established with same Superior address and Superior
2908 identifier except those from which RESIGN has been received. If the CONTEXT had
2909 “superior type” of “cohesion”, the “remaining Inferiors” excludes any that it has been
2910 determined will be cancelled, as well as any that have resigned – in other words it includes
2911 only those for which a confirm decision is still possible or has been made. The determination
2912 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some
2913 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this
2914 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.

2915

2916 In order to ensure that the confirmation decision is delivered to all remaining Inferiors,
2917 despite failures, the Superior must persistently record which these Inferiors are (i.e. their
2918 addresses and identifiers). It must also either record that the decision is confirm, or ensure

2919 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it
 2920 will be told about it. This latter would apply if the Superior were also BTP Inferior to another
 2921 entity which persisted a confirm decision (or recursively deferred it still higher). However,
 2922 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
 2923 behaviour of asking another entity to make (and persist) the confirm decision is termed
 2924 "offering confirmation" - the Superior offers the possible confirmation of itself, and its
 2925 remaining Inferiors to some other entity. If that entity (or something higher up) then does
 2926 make and persist a confirm decision, the Superior is "instructed to confirm" (which is
 2927 equivalent BTP CONFIRM).

2928
 2929 The application, or an entity acting indirectly on behalf of the application, may request a
 2930 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
 2931 more operations associated with the Inferior. Following a request to prepare all remaining
 2932 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
 2933 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
 2934 application.)

2935
 2936 The application, or an entity acting indirectly on behalf of the application, may also request
 2937 confirmation. This means the Superior is to attempt to make and persist a confirm decision
 2938 itself, rather than offer confirmation.

2939
 2940

2941

Table 1 : send, receive and disruption events

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

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

2942

2943

Table 2 : Decision events for Inferior

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

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

2944

2945

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"> o PREPARED or PREPARED/cancel has been received from all other remaining Inferiors; AND o Superior has been requested to confirm; AND o persistent information records the confirm decision and identifies all remaining Inferiors; • Or <ul style="list-style-type: none"> o persistent information records an offer of confirmation and has been instructed to confirm
decide to cancel	<ul style="list-style-type: none"> • Superior has not offered confirmation; OR • Superior has offered confirmation and has been instructed to cancel; OR

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

2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974

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

2978

2979

2980

2981

2982

2983

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	A 1	B 1	B 2	C 1	D 1	E 1	E 2	F 1	F 2
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					F1	K1		
deci de to confi rm					L1	G4		
deci de to cancel								
remove persi stent i nformati on							R1	R1
record contradi cti on								
di srupti on I	Z	Z	Z	Z	Z	Z	F1	Z
di srupti on II			G2	G2	E1	E1		G2
di srupti on III					D1	D1		
di srupti on IV					B1	B1		

Table 8: Superior state table – hazard and request confirm

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

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

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

2990

2991

Table 10: Inferior state table – normal forward progression

	i 1	a1	b1	c1	d1	e1	e2	f1	f2
send ENROL/rsp-req	a1	a1							
send ENROL/no-rsp-req	b1		b1						
send RESIGN/rsp-req				c1					
send RESIGN/no-rsp-req				z					
send PREPARED						e1			
send PREPARED/cancel							e2		
send CONFIRMED/auto									
send CONFIRMED/response									
send CANCELLED			z		z				
send HAZARD									
send INF_STATE/active/y		a1	b1		d1				
send INF_STATE/active			b1		d1				
send INF_STATE/unknown									
receive ENROLLED		b1	b1	c1		e1	e2		
receive RESIGNED				z					
receive PREPARE		d1	d1	c1	d1	e1	e2		
receive CONFIRM_ONE_PHASE		s2	s2	z		s1	s1		
receive CONFIRM						f1	f2	f1	f2
receive CANCEL		n1	n1	z	n1	g1	g2		
receive CONTRADICTION									
receive SUP_STATE/active/y		b1	b1	c1		e1	e2		
receive SUP_STATE/active		b1	b1	c1		e1	e2		
receive SUP_STATE/prepared-rcvd/y						e1	e2		
receive SUP_STATE/prepared-rcvd						e1	e2		
receive SUP_STATE/unknown		z	z	z	z	x1	x2		
decide to resign				c1	c1				
decide to be prepared				e1	e1				
decide to be prepared/cancel				e2	e2				
decide to confirm autonomously						h1			
decide to cancel autonomously						j1	z1		
apply ordered confirmation								m1	m1
remove persistent information									
detect problem		p1	p1		p1	p2	p2	p2	p2
detect and record problem									
disruption I		z	z	z	z			e1	e2
disruption II					b1				
disruption III									

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			h1		j1					
	g1	g2	h1	h2	j1	j2	k1		l1	
			l1		j2	j2	k2	k2	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			h1		j1					
			h1		j1					
			h1		j1					
	x1	x2	l1		j2	j2	k2	k2	l1	
decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem										
	n1	n1		m1		z		z		z
	p2	p2								
disruption I disruption II disruption III	e1	e2	h1		j1		j1	k1	h1	l1
							j1			h1

2995

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 p1 s5 p2 p1 z	p2 p2 s5 p2 p2 z	q1 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 p2 p2 p1	p2 p2 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		

2996

2997

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 appl y ordered confi rmati on remove persi stent i nformati on detect probl em detect and record probl em			s3 s4			s6
di srupti on I di srupti on II di srupti on III	e1	z		z	z	

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

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

3000

3001

3001 **Failure Recovery**

3002 **Types of failure**

3003
3004 BTP is designed to ensure the delivery of a consistent decision for a business transaction to
3005 the parties involved, even in the event of failure. Failures can be classified as:

3006
3007 **Communication failure:** messages between BTP actors are lost and not
3008 delivered. BTP assumes the carrier protocol ensures that messages are either
3009 delivered correctly (without corruption) or are lost, but does not assume that all
3010 losses are reported or that messages sent separately are delivered in the order of
3011 sending.

3012
3013 **Node failure (system failure, site failure):** a machine hosting one or more BTP
3014 actors stops processing and all its volatile data is lost. BTP assumes a site fails by
3015 stopping – it either operates correctly or not at all, it never operates incorrectly.

3016
3017 Communication failure may become known to a BTP implementation by an indication from
3018 the lower layers or may be inferred (or suspected) by the expiry of a timeout. Recovery from
3019 a communication failure requires only that the two actors can again send messages to each
3020 other and continue or complete the progress of the business transaction. In the state tables for
3021 the Superior:Inferior relationship, each side is either waiting to make a decision or can send a
3022 message. For some states, the message to be sent is a repetition of a regular message; for
3023 other states, the INFERIOR_STATE or SUPERIOR_STATE message can be sent, requesting
3024 a response. Thus, following a communication failure, either side can prompt the other to re-
3025 establish the relationship. Receiving one of the *_STATE messages asking for a response
3026 does not require an immediate response – especially if an implementation is waiting to
3027 determine a decision (perhaps because it is itself waiting for a decision from elsewhere), an
3028 implementation may choose not to reply until it wishes too.

3029
3030 A node failure is distinguished from communication failure because there is loss of volatile
3031 state. To ensure consistent application of the decision of a business transaction, BTP requires
3032 that some state information will be persisted despite node failure. Exactly what real events
3033 correspond to node failure but leave the persistent information undamaged is a matter for
3034 implementation choice, depending on application requirements; however, for most
3035 application uses, power failure should be survivable (an exception would be if the data
3036 manipulated by the associated operations was volatile). There will always be some level of
3037 event sufficiently catastrophic to lose persistent information and the ability to recover–
3038 destruction of the computer or bankruptcy of the organisation, for example.

3039
3040 Recovery from node failure involves recreating the endpoint in a node that has access to the
3041 persistent information for incomplete transactions. This may be a recreation of the original
3042 node (including the ability to perform application work) using the same addresses; or there
3043 may be a distinct recovery entity, which can access the persistent data, but has a different
3044 address; other implementation approaches are possible. Restoration of the endpoint from
3045 persistent information will often result in a partial loss of state, relative to the volatile state
3046 reached before the failure. This is modelled in the state tables by the “disruption” events.

3047 After recovery from node failure, the implementation behaves much as if a communication
3048 failure had occurred.

3049

3050 Persistent information

3051

3052 BTP requires that some decision events are persisted – that information recording an
3053 Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s
3054 autonomous decision survive failure. Making the first two decisions persistent ensures that a
3055 consistent decision can be reached for the business transaction and that it is delivered to all
3056 involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to
3057 ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the
3058 contradiction will be reported to the Superior, despite failures.

3059

3060 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
3061 active state (unlike many transaction protocols, where a communication or endpoint failure in
3062 active state would invariably cause rollback of the transaction). Recovery in the active state
3063 may require that the application exchange is resynchronised as well – BTP does not directly
3064 support this, but does allow continuation of the business transaction as such. In the state
3065 tables, from some states, there are several levels of disruption, distinguished by which state
3066 the implementation transits to – this represents the survival of different extents of state
3067 information over failure and recovery. The different levels of disruption describe legitimate
3068 states for the endpoint to be in after it has recovered – **they do not require that all
3069 implementations are able to exhibit the appropriate partial loss of state information.**

3070 The absence of a destination state for the disruption events means that such a transition is not
3071 legitimate – thus, for example, an Inferior that has decided to be prepared will always recover
3072 to the same state, by virtue of the information persisted in the “decide to be prepared” event.

3073

3074 Apart from the (optional) recovery in active state, BTP follows the well-known presume-
3075 abort model – it is only required that information be persisted when decisions are made (and
3076 not, e.g. on enrolment). This means that on recovery, one side may have persistent
3077 information but the other does not. This occurs when an Inferior has decided to be prepared
3078 but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the
3079 Superior did confirm, and the Inferior applied the confirm, removed its persistent information
3080 but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it
3081 still had the persistent information when the failure occurred).

3082

3083 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to
3084 re-establish communication with the Superior, to apply a confirm decision and to apply a
3085 cancel decision. It will thus need to include

3086 Inferior identity (this may be an index used to locate the information)

3087 Superior address (as on CONTEXT)

3088 Superior identifier (as on CONTEXT)

3089 default-is-cancel value (as on PREPARED)

3090

3091 The information needed to apply confirm/cancel decisions will depend on the application and
3092 the associated operations. It may also normally be necessary to persist any qualifiers that

3093 were sent with the PREPARED message or application messages sent with the PREPARED,
3094 since the PREPARED message will be repeated if a failure occurs.

3095

3096 A Superior must record corresponding information to allow it to re-establish communication
3097 with the Inferior:

3098 Inferior address (as on ENROL)

3099 Inferior identifier (as on ENROL)

3100

3101 A Superior that is the Decider for the business transaction need only persist this information
3102 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A
3103 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as
3104 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as
3105 Superior (to this Inferior) as part of the persistent information of its decision to be prepared
3106 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and
3107 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If
3108 CONFIRM is received, the persistent information may be changed to show the confirm
3109 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.
3110 If the persistent information is left unchanged and there is a node failure, on recovery the
3111 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision
3112 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

3113

3114 After failure, an implementation may not be able to restore an endpoint to the appropriate
3115 state immediately – in particular, the necessary persistent information may be inaccessible,
3116 although the implementation can respond to received BTP messages. In such a case, a
3117 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a “reply-
3118 requested” value “false”) with SUPERIOR_STATE/inaccessible and an Inferior to any BTP
3119 message except SUPERIOR_STATE/* with “INFERIOR_STATE/inaccessible. Receipt of
3120 the *_STATE/inaccessible messages has no effect on the endpoint state.

3121

3122 **Redirection**

3123

3124 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
3125 that there are cases where one side will attempt to re-establish communication when there is
3126 no persistent information for the relationship at the far-end. In such cases, it is important the
3127 side that is attempting recovery can distinguish between unsuccessful attempts to connect to
3128 the holder of the persistent information and when the information no longer exists. If the peer
3129 information does not exist, this side can draw conclusions and complete appropriately; if they
3130 merely fail to get through they are stuck in attempting recovery.

3131

3132 Two mechanisms are provided to make it possible that even when one side of a
3133 Superior:Inferior relationship has completed, that a message can eventually get through to
3134 something that can definitively report the status, distinguishing this case from a temporary
3135 inability to access the state of a continuing transaction element. The mechanisms are:

- 3136 o Address fields which provide a “callback address” can be a set of addresses,
3137 which are alternatives one of which is chosen as the target address for the
3138 future message. If the sender of that message finds the address does not work,
3139 it can try a different alternative.

- 3140 o The REDIRECT message can be used to inform the peer that an address
3141 previously given is no longer valid and to supply a replacement address (or
3142 set of addresses). REDIRECT can be issued either as a response to receipt of
3143 a message or spontaneously.
3144

3145 The two mechanisms can be used in combination, with one or more of the original set of
3146 addresses just being a redirector, which does not itself ever have direct access to the state
3147 information for the transaction, but will respond to any message with an appropriate
3148 REDIRECT.

3149 An alternative implementation approach is to have a single addressable entity that uses the
3150 same address for all transactions, distinguishing them by identifier, and which always
3151 recovers to use the same address. Such an implementation would not need to supply
3152 “backup” addresses (and would only use REDIRECT if it was being permanently migrated).
3153

3154 **Terminator:Decider failures**

3155 BTP does not provide facilities or impose requirements on the recovery of
3156 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
3157 may survive failures (by retaining knowledge of the Decider’s address and identifier), but this
3158 is an implementation option. Although a Decider (if it decides to confirm) will persist
3159 information about the confirm decision, it is not required, after failure, to remain accessible
3160 using the inferior address it offered to the Terminator. Any such recovery is an
3161 implementation option.
3162

3163 A Decider’s address (as returned on BEGUN) may be a set of addresses, allowing a failed
3164 Decider to be recovered at a different address.
3165

3166 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
3167 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for
3168 a CONFIRM_TRANSACTION that will never arrive, the standard qualifier “Transaction
3169 timelimit” can be used (by the Initiator) to inform the Decider when it can assume the
3170 Terminator will not issue CONFIRM_TRANSACTION and so it (the Decider) should initiate
3171 cancellation.
3172

3173 **XML representation of Message Set**

3174 This section describes the syntax for BTP messages in XML. These XML messages represent
3175 a midpoint between the abstract messages and what actually gets sent on the wire.
3176

3177 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)
3178 [3121](#)
3179

3180 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:xml
3181

3182 In addition to an XML schema, this specification uses an informal syntax to describe the
3183 structure of the BTP messages. The syntax appears as an XML instance, but the values
3184

3187 contain data types instead of values. The following symbols are appended to some of the
3188 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
3189 these symbols corresponds to "one and only one."
3190

3191 Addresses

3192
3193 As described in the "Abstract Message and Associated Contracts – Addresses" section, a BTP
3194 address comprises three parts, and for a target address only the "additional information" field
3195 is inside the BTP messages. For all BTP messages whose abstract form includes a target
3196 address parameter, the corresponding XML representation includes a "target-additional-
3197 information" element. This element may be omitted if it would be empty.
3198

3199 For other addresses, all three fields are represent, as in:

```
3200 <ctp:some-address>  
3201   <ctp:binding-name>...carrier binding URI...</ctp:binding-name>  
3202   <ctp:binding-address>...carrier specific  
3203   address...</ctp:binding-address>  
3204   <ctp:additional-information>...optional additional addressing  
3205   information...</ctp:additional-information> ?  
3206 </ctp:some-address>
```

3207
3208
3209
3210 A "published" address can be a set of <some-address>, which are alternatives which can be
3211 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to
3212 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice
3213 of which address to use (depending on which binding is preferable.) In the case where
3214 multiple addresses are used for redundancy, a priority attribute can be specified to help the
3215 receiver choose among the addresses- the address with the highest priority should be used,
3216 other things being equal. The priority is used as a hint and does not enforce any behaviour in
3217 the receiver of the message. Default priority is a value of 1.
3218

3219 Qualifiers

3220 The "Qualifier name" is used as the element name, within the namespace of the "Qualifier
3221 group".
3222

3223 Examples:

```
3224 <ctpq:inferior-timeout  
3225   xmlns:ctpq="urn:oasis:names:tc:BTP:qualifiers"  
3226   xmlns:ctp="urn:oasis:names:tc:BTP:xml "  
3227   ctp:must-be-understood="false"  
3228   ctp:to-be-propagated="false">1800</ctpq:inferior-timeout>  
3229  
3230 <auth:username  
3231   xmlns:auth="http://www.example.com/ns/auth"  
3232   xmlns:ctp="urn:oasis:names:tc:BTP:xml "  
3233   ctp:must-be-understood="true"  
3234   ctp:to-be-propagated="true">jtauber</auth:username>  
3235
```

3236 Attributes must-be-understood **has default value “true”** and to-be-propagated has default
3237 value “false”.

3238

3239 Identifiers

3240

3241 Identifiers shall be URIs Unspecified length strings made of up hexadecimal digits (0->9, A-
3242 >F). Note: lower case a->f are not valid.

3243

3244 Examples: "01", "FAB224234CCCC2"

3245

3246

Note — Identifiers need to be globally unambiguous. Apart from their
3247 generation, Use of hexadecimal digits avoids problems with character code
3248 representations. The only operation the BTP implementations have to
3249 perform on identifiers is to match them.

3250

3251 Message References

3252 Each BTP message has an optional id attribute to give it a unique identifier. An application
3253 can make use of those identifiers, but no processing is enforced.

3254

3255 Messages

3256

3257 CONTEXT

3258

```
3259 <btpr:context id? superior-type="cohesion|atom" id?>  
3260 <btpr:superior-address>- +  
3261 ...address...  
3262 </btpr:superior-address>  
3263 <btpr:superior-identifier>...hexstringURI...</btpr:superior-  
3264 identifier>  
3265 <btpr:reply-address> ?  
3266 ...address...  
3267 </btpr:reply-address>  
3268 <btpr:superior-type>cohesion|atom</btpr:superior-type>  
3269 <btpr:qualifiers> ?  
3270 ...qualifiers...  
3271 </btpr:qualifiers>  
3272 </btpr:context>
```

3273

3274

3275 CONTEXT_REPLY

3276

```
3277 <btpr:context-reply id? superior-type="cohesion|atom" id?>  
3278 <btpr:target-additional-information> ?  
3279 ...additional address information...  
3280 </btpr:target-additional-information>  
3281 <btpr:superior-address> +  
3282 ...address...
```

```

3283 </btp:superior-address>
3284 <btp:superior-identifier>...hexstringURI...</btp:superior-
3285 identifier>
3286 <btp:completion-
3287 status>completed|related|repudiated</btp:completion-status>
3288 <btp:qualifiers> ?
3289 ...qualifiers...
3290 </btp:qualifiers>
3291 </btp:context-reply>

```

REQUEST_STATUS

```

3292
3293
3294
3295 <btp:request-status id?>
3296 <btp:target-additional-information> ?
3297 ...additional address information...
3298 </btp:target-additional-information>
3299 <btp:reply-address> ?
3300 ...address...
3301 </btp:reply-address>
3302 <btp:target-identifier>...URI...</btp:target-identifier>
3303 <btp:qualifiers> ?
3304 ...qualifiers...
3305 </btp:qualifiers>
3306 </btp:request-status>
3307

```

STATUS

```

3308
3309
3310 <btp:status id?>
3311 <btp:target-additional-information> ?
3312 ...additional address information...
3313 </btp:target-additional-information>
3314 <btp:responders-identifier>...URI...</btp:responders-identifier>
3315
3316 <btp:status-value>created|enrolling|active|resigning|
3317 resigned|preparing|prepared|
3318 confirming|confirmed|cancelling|cancelled|
3319 cancel-contradiction|confirm-contradiction|
3320 hazard|contradicted|unknown|inaccessible</btp:status-
3321 value>
3322 <btp:qualifiers> ?
3323 ...qualifiers...
3324 </btp:qualifiers>
3325 </btp:status>
3326

```

FAULT

```

3327
3328
3329 <btp:fault id?>
3330 <btp:target-additional-information> ?
3331 ...additional address information...
3332 </btp:target-additional-information>
3333 <btp:superior-identifier>...URI...</btp:superior-identifier> ?
3334 <btp:inferior-identifier>...URI...</btp:inferior-identifier> ?

```

3335 `<btp: fault-type>...fault type name...</btp: fault-type>`
3336 `<btp: fault-data>...fault data...</btp: fault-data> ?`
3337 `<btp: qualifiers> ?`
3338 `...qualifiers...`
3339 `</btp: qualifiers>`
3340 `</btp: fault>`

3341
3342 The following fault type names are represented by simple strings, corresponding to the entries
3343 defined in the abstract message set:

3344
3345 o communication-failure
3346 o duplicate-inferior
3347 o general
3348 o invalid-decider
3349 o invalid-inferior
3350 o invalid-superior
3351 o status-refused
3352 o invalid-terminator
3353 o unknown-parameter
3354 o unknown-transaction
3355 o unsupported-qualifier
3356 o wrong-state

3357
3358 Revisions of this specification may add other fault type names, which shall be simple strings
3359 of letters, numbers and hyphens. If other specifications define fault type names to be used
3360 with BTP, the names shall be URIs.

3361 Fault data can take on various forms:

3362
3363
3364 Free text:

3365 `<btp: fault-data>...string data...</btp: fault-data>`

3366
3367
3368 Identifier:

3369
3370 `<btp: fault-data>...URI...</btp: fault-data>`

3371
3372
3373 Inferior Identity:

3374
3375 `<btp: fault-data>`
3376 `<btp: inferior-address> +`
3377 `...address...`
3378 `</btp: inferior-address>`
3379 `<btp: inferior-identifier>...URI...</btp: inferior-identifier>`
3380 `</btp: fault-data>`

3381
3382

3383

3384

BEGIN

3385

3386

```
<btpt:begin id? transaction-type="cohesion|atom">
```

3387

```
<btpt:target-additional-information>
```

3388

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

3389

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

3390

```
<btpt:reply-address>
```

3391

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

3392

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

3393

```
<btpt:qualifiers> ?
```

3394

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

3395

```
</btpt:qualifiers>
```

3396

```
</btpt:begin>
```

3397

3398

3399

BEGUN

3400

```
<btpt:begin id? transaction-type="cohesion|atom">
```

3401

```
<btpt:target-additional-information>
```

3402

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

3403

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

3404

```
<btpt:decider-address> ?
```

3405

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

3406

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

3407

```
<btpt:transaction-identifier>...hexstring...</btpt:transaction-
```

3408

```
identifier> ?
```

3409

```
<btpt:inferior-handle>...hexstring...</btpt:inferior-handle> ?
```

3410

```
<btpt:inferior-address> ?
```

3411

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

3412

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

3413

```
<btpt:qualifiers> ?
```

3414

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

3415

```
</btpt:qualifiers>
```

3416

```
</btpt:begin>
```

3417

3418

3419

3420

ENROL

3421

```
<btpt:enrol reply-requested="true|false" _____ id?>
```

3422

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

3423

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

3424

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

3425

```
<btpt:superior-identifier>...hexstringURI...</btpt:superior-
```

3426

```
identifier>
```

3427

```
<btpt:reply-requested>true|false</btpt:reply-requested>
```

3428

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

3429

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

3430

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

3431

```
<btpt:inferior-address> +
```

3432

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

3433

```
3434 </btp:inferior-address>
3435 <btp:inferior-identifier>...hexstringURI...</btp:inferior-
3436 identifier>
3437 <btp:qualifiers> ?
3438 ...qualifiers...
3439 </btp:qualifiers>
3440 </btp:enrol>
```

ENROLLED

```
3445 <btp:enrolled id?>
3446 <btp:target-additional-information> ?
3447 ...additional address information...
3448 </btp:target-additional-information>
3449 <btp:inferior-identifier>...hexstringURI...</btp:inferior-
3450 identifier>
3451 <btp:inferior-handle>...hexstring...</btp:inferior-handle> ?
3452 <btp:qualifiers> ?
3453 ...qualifiers...
3454 </btp:qualifiers>
3455 </btp:enrolled>
```

RESIGN

```
3460 <btp:resign response-requested="true|false" id?>
3461 <btp:target-additional-information> ?
3462 ...additional address information...
3463 </btp:target-additional-information>
3464 <btp:superior-identifier>...hexstringURI...</btp:superior-
3465 identifier>
3466 <btp:inferior-address> +
3467 ...address...
3468 </btp:inferior-address>
3469 <btp:inferior-identifier>...hexstringURI...</btp:inferior-
3470 identifier>
3471 <btp:response-requested>true|false</btp:response-requested>
3472 <btp:qualifiers> ?
3473 ...qualifiers...
3474 </btp:qualifiers>
3475 </btp:resign>
```

RESIGNED

```
3480 <btp:resigned id?>
3481 <btp:target-additional-information> ?
3482 ...additional address information...
3483 </btp:target-additional-information>
```



```
3484 <btpr:inferior-identifier>...hexstringURI...</btpr:inferior-
3485 identifier>
3486 <btpr:qualifiers> ?
3487 ...qualifiers...
3488 </btpr:qualifiers>
3489 </btpr:resigned>
```

PREPARE

```
3493
3494 <btpr:prepare id?>
3495 <btpr:target-additional-information> ?
3496 ...additional address information...
3497 </btpr:target-additional-information>
3498 <btpr:inferior-identifier>...hexstringURI...</btpr:inferior-
3499 identifier>?
3500 <btpr:qualifiers> ?
3501 ...qualifiers...
3502 </btpr:qualifiers>
3503 </btpr:prepare>
```

PREPARED

```
3504
3505
3506
3507
3508 <btpr:prepared default-is-cancel="false|true" id?>
3509 <btpr:target-additional-information> ?
3510 ...additional address information...
3511 </btpr:target-additional-information>
3512 <btpr:superior-identifier>...hexstringURI...</btpr:superior-
3513 identifier>
3514 <btpr:inferior-address> +
3515 ...address...
3516 </btpr:inferior-address>
3517 <btpr:inferior-identifier>...hexstringURI...</btpr:inferior-
3518 identifier>
3519 <btpr:default-is-cancel>true|false</btpr:default-is-cancel>
3520 <btpr:qualifiers> ?
3521 ...qualifiers...
3522 </btpr:qualifiers>
3523 </btpr:prepared>
```

CONFIRM

```
3524
3525
3526
3527
3528 <btpr:confirm id?>
3529 <btpr:target-additional-information> ?
3530 ...additional address information...
3531 </btpr:target-additional-information>
3532 <btpr:inferior-identifier>...hexstringURI...</btpr:inferior-
3533 identifier>
3534 <btpr:qualifiers> ?
```

```
3535     ...qualifiers...
3536     </btp:qualifiers>
3537 </btp:confirm>
```

CONFIRMED

```
3541
3542 <btp:confirmed confirmed-received="true|false" id?>
3543   <btp:target-additional-information> ?
3544     ...additional address information...
3545   </btp:target-additional-information>
3546   <btp:superior-identifier>...hexstringURI...</btp:superior-
3547   identifier>
3548   <btp:inferior-address> ?
3549   ...address...
3550   </btp:inferior-address>
3551   <btp:inferior-identifier>...hexstringURI...</btp:inferior-
3552   identifier> ?
3553   <btp:confirmed-received>true|false</btp:confirmed-received>
3554   <btp:qualifiers> ?
3555     ...qualifiers...
3556   </btp:qualifiers>
3557 </btp:confirmed>
```

CANCEL

```
3561
3562 <btp:cancel id?>
3563   <btp:target-additional-information> ?
3564     ...additional address information...
3565   </btp:target-additional-information>
3566   <btp:inferior-identifier>...hexstringURI...</btp:inferior-
3567   identifier> ?
3568   <btp:reply-address> ?
3569     ...address...
3570   </btp:reply-address>
3571   <btp:qualifiers> ?
3572     ...qualifiers...
3573   </btp:qualifiers>
3574 </btp:cancel>
```

CANCELLED

```
3578
3579 <btp:cancelled id?>
3580   <btp:target-additional-information> ?
3581     ...additional address information...
3582   </btp:target-additional-information>
3583   <btp:superior-identifier>...hexstringURI...</btp:superior-
3584   identifier>
3585   <btp:inferior-address> ?
```

```
3586 ...address...
3587 </btp:inferior-address> ?
3588 <btp:inferior-identifier>...hexstringURI...</btp:inferior-
3589 identifier> ?
3590 <btp:qualifiers> ?
3591 ...qualifiers...
3592 </btp:qualifiers>
3593 </btp:cancelled>
```

CONFIRM_ONE_PHASE

```
3594
3595
3596
3597
3598 <btp:confirm-one-phase report-hazard="true|false" id?>
3599 <btp:target-additional-information> ?
3600 ...additional address information...
3601 </btp:target-additional-information>
3602 <btp:inferior-identifier>...hexstringURI...</btp:inferior-
3603 identifier>
3604 <btp:report-hazard>true|false</btp:report-hazard>
3605 <btp:qualifiers> ?
3606 ...qualifiers...
3607 </btp:qualifiers>
3608 </btp:confirm-one-phase>
```

HAZARD

```
3609
3610
3611
3612 <btp:hazard level="mixed|possible" id?>
3613 <btp:target-additional-information> ?
3614 ...additional address information...
3615 </btp:target-additional-information>
3616 <btp:superior-identifier>...hexstringURI...</btp:superior-
3617 identifier>
3618 <btp:inferior-address> +
3619 ...address...
3620 </btp:inferior-address>
3621 <btp:inferior-identifier>...hexstringURI...</btp:inferior-
3622 identifier>
3623 <btp:level>mixed|possible</btp:level>
3624 <btp:qualifiers> ?
3625 ...qualifiers...
3626 </btp:qualifiers>
3627 </btp:hazard>
```

CONTRADICTION

```
3628
3629
3630
3631
3632 <btp:contradiction id?>
3633 <btp:target-additional-information> ?
3634 ...additional address information...
3635 </btp:target-additional-information>
```

```
3636 <btpp:inferior-identifier>...hexstringURI...</btpp:inferior-
3637 identifier>
3638 <btpp:qualifiers> ?
3639 ...qualifiers...
3640 </btpp:qualifiers>
3641 </btpp:contradiction>
```

3644 SUPERIOR_STATE

```
3645
3646 <btpp:superior-state reply-requested="true|false"id?>
3647 <btpp:target-additional-information> ?
3648 ...additional address information...
3649 </btpp:target-additional-information>
3650 <btpp:inferior-identifier>...hexstringURI...</btpp:inferior-
3651 identifier>
3652 <btpp:status>active|prepared-
3653 received|inaccessible|unknown</btpp:status>
3654 <btpp:reply-requested>true|false</btpp:reply-requested>
3655 <btpp:qualifiers> ?
3656 ...qualifiers...
3657 </btpp:qualifiers>
3658 </btpp:superior-state>
```

3661 INFERIOR_STATE

```
3662
3663 <btpp:inferior-state reply-requested="true|false"id?>
3664 <btpp:target-additional-information> ?
3665 ...additional address information...
3666 </btpp:target-additional-information>
3667 <btpp:superior-identifier>...hexstringURI...</btpp:superior-
3668 identifier>
3669 <btpp:inferior-address> +
3670 ...address...
3671 </btpp:inferior-address>
3672 <btpp:inferior-identifier>...hexstringURI...</btpp:inferior-
3673 identifier>
3674 <btpp:status>-active|-inaccessible|unknown</btpp:status>
3675 <btpp:reply-requested>true|false</btpp:reply-requested>
3676 <btpp:qualifiers> ?
3677 ...qualifiers...
3678 </btpp:qualifiers>
3679 </btpp:inferior-state>
```

3684 REDIRECT

```
3685 <btpp:redirect id?>
3686
```

```

3687 <btptarget-additional-information> ?
3688   ...additional address information...
3689 </btptarget-additional-information>
3690 <btptsuperior-identifier>...hexstringURI...</btptsuperior-
3691 identifier> ?
3692 <btptinferior-identifier>...hexstringURI...</btptinferior-
3693 identifier>
3694 <btptold-address> +
3695   ...address...
3696 </btptold-address>
3697 <btptnew-address> +
3698   ...address...
3699 </btptnew-address>
3700 <btptqualifiers> ?
3701   ...qualifiers...
3702 </btptqualifiers>
3703 </btptredirect>

```

BEGIN

```

3704
3705
3706
3707 <btptbegin id?>
3708   <btpttarget-additional-information> ?
3709     ...additional address information...
3710   </btpttarget-additional-information>
3711   <btptreply-address> ?
3712     ...address...
3713   </btptreply-address>
3714   <btpttransaction-type>cohesion|atom</btpttransaction-type>
3715   <btptqualifiers> ?
3716     ...qualifiers...
3717   </btptqualifiers>
3718 </btptbegin>

```

BEGUN

```

3719
3720
3721
3722
3723 <btptbegun id?>
3724   <btpttarget-additional-information> ?
3725     ...additional address information...
3726   </btpttarget-additional-information>
3727   <btptdecider-address> *
3728     ...address...
3729   </btptdecider-address>
3730   <btptinferior-address> *
3731     ...address...
3732   </btptinferior-address>
3733   <btpttransaction-identifier>...URI...</btpttransaction-
3734 identifier>?
3735   <btptinferior-handle>...URI...</btptinferior-handle> ?
3736   <btptinferior-address> *
3737     ...address...
3738   </btptinferior-address>

```

```
3739 <btpr:qualifiers> ?
3740 ...qualifiers...
3741 </btpr:qualifiers>
3742 </btpr:begin>
```

PREPARE_INFERIORS

```
3746
3747 <btpr:prepare-inferiors id?>
3748 <btpr:target-additional-information> ?
3749 ...additional address information...
3750 </btpr:target-additional-information>
3751 <btpr:reply-address> ?
3752 ...address...
3753 </btpr:reply-address>
3754 <btpr:transaction-identifier>...hexstringURI...</btpr:transaction-
3755 identifier>
3756 <btpr:inferiors-list> ?
3757 <btpr:inferior-handle>...hexstringURI...</btpr:inferior-
3758 handle> +
3759 </btpr:inferiors-list>
3760 <btpr:qualifiers> ?
3761 ...qualifiers...
3762 </btpr:qualifiers>
3763 </btpr:prepare-inferiors>
```

CONFIRM_TRANSACTION

```
3764
3765
3766
3767
3768 <btpr:confirm-transaction report-hazard="true|false" id?>
3769 <btpr:target-additional-information> ?
3770 ...additional address information...
3771 </btpr:target-additional-information>
3772 <btpr:reply-address> ?
3773 ...address...
3774 </btpr:reply-address>
3775 <btpr:transaction-identifier>...hexstringURI...</btpr:transaction-
3776 identifier>
3777 <btpr:inferiors-list> ?
3778 <btpr:inferior-handle>...hexstringURI...</btpr:inferior-
3779 handle> +
3780 </btpr:inferiors-list>
3781 <btpr:report-hazard>true|false</btpr:report-hazard>
3782 <btpr:qualifiers> ?
3783 ...qualifiers...
3784 </btpr:qualifiers>
3785 </btpr:confirm_transaction>
```

TRANSACTION_CONFIRMED

3788
3789

```

3790 <btpt:transaction-confirmed id?>
3791   <btpt:target-additional-information> ?
3792   ...additional address information...
3793 </btpt:target-additional-information>
3794 <btpt:decider-address> ?
3795 ...address...
3796 </btpt:decider-address>
3797   <btpt:transaction-identifier>...hexstringURI...</btpt:transaction-
3798 identifier> -?
3799   <btpt:qualifiers> ?
3800   ...qualifiers...
3801 </btpt:qualifiers>
3802 </btpt:transaction-confirmed>

```

3803
3804

CANCEL_TRANSACTION

```

3806
3807 <btpt:cancel-transaction id?>
3808   <btpt:target-additional-information> ?
3809   ...additional address information...
3810 </btpt:target-additional-information>
3811   <btpt:reply-address> -?
3812   ...address...
3813 </btpt:reply-address>
3814   <btpt:transaction-identifier>...hexstringURI...</btpt:transaction-
3815 identifier> -?
3816   <btpt:report-hazard>true|false</btpt:report-hazard>
3817   <btpt:qualifiers> ?
3818   ...qualifiers...
3819 </btpt:qualifiers>
3820 </btpt:cancel-transaction>

```

3821
3822

CANCEL_INFERIORS

```

3823
3824 <btpt:cancel-inferiors id?>
3825   <btpt:target-additional-information> ?
3826   ...additional address information...
3827 </btpt:target-additional-information>
3828   <btpt:reply-address> - ?
3829   ...address...
3830 </btpt:reply-address>
3831   <btpt:transaction-identifier>...hexstringURI...</btpt:transaction-
3832 identifier> ?
3833   <btpt:inferiors-list>
3834   _____ <btpt:inferior-handle>...hexstringURI...</btpt:inferior-
3835 handle> +
3836 </btpt:inferiors-list>
3837   <btpt:qualifiers> ?
3838   ...qualifiers...
3839 </btpt:qualifiers>
3840 </btpt:cancel-inferiors>
3841

```

3842

3843

TRANSACTION_CANCELLED

3844

3845

```
<btpr:cancel-complete-transaction-cancelled id?>
```

3846

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

3847

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

3848

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

3849

```
<btpr:decider-address> ?
```

3850

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

3851

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

3852

```
<btpr:transaction-identifier>...hexstringURI...</btpr:transaction-
```

3853

```
identifier> ?
```

3854

```
<btpr:qualifiers> ?
```

3855

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

3856

```
</btpr:qualifiers>
```

3857

```
</btpr:cancel-complete-transaction-cancelled>
```

3858

3859

3860

REQUEST_INFERIOR_STATUSES

3861

```
<btpr:request-inferior-statuses id?>
```

3863

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

3864

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

3865

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

3866

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

3867

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

3868

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

3869

```
<btpr:target-identifier>...hexstringURI...</btpr:target-
```

3870

```
identifier>
```

3871

```
<btpr:inferiors-list> ?
```

3872

```
<btpr:inferior-handle>...hexstringURI...</btpr:inferior-
```

3873

```
handle> +
```

3874

```
</btpr:inferiors-list>
```

3875

```
<btpr:qualifiers> ?
```

3876

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

3877

```
</btpr:qualifiers>
```

3878

```
</btpr:request-inferior-statuses>
```

3879

3880

3881

INFERIOR_STATUSES

3882

```
<btpr:inferior-statuses id?>
```

3883

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

3884

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

3885

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

3886

```
<btpr:responders-address>
```

3887

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

3888

```
</btpr:responders-address>
```

3889

```
<btpr:responders-identifier>...hexstringURI...</btpr:responders-
```

3890

```
identifier>
```

3891

```
<btpr:status-list>
```

3892


```

3893         <btpr:status-item> +
3894         <btpr:inferior-handle>...hexstringURI...</btpr:inferior-
3895 handle>
3896         <btpr:status>active|resigned|preparing|prepared|
3897         autonomously-confirmed|autonomously-cancelled|
3898         confirming|confirmed|cancelling|cancelled|
3899         cancel-contradiction|confirm-contradiction|
3900         hazard|invalid</btpr:status>
3901         <btpr:qualifiers> ?
3902         ...qualifiers...
3903     </btpr:qualifiers>
3904 </btpr:status-item>
3905 </btpr:status-list>
3906 <btpr:qualifiers> ?
3907     ...qualifiers...
3908 </btpr:qualifiers>
3909 </btpr:inferior-statuses>

```

REQUEST_STATUS

```

3914 <btpr:request_status_id?>
3915 <btpr:target-additional-information>
3916 ...additional-address-information...
3917 </btpr:target-additional-information>
3918 <btpr:reply-address>
3919 ...address...
3920 </btpr:reply-address>
3921 <btpr:target-identifier>...hexstring...</btpr:target-identifier>
3922 <btpr:qualifiers> ?
3923 ...qualifiers...
3924 </btpr:qualifiers>
3925 </btpr:request_status>

```

STATUS

```

3929 <btpr:status_id?>
3930 <btpr:target-additional-information>
3931 ...additional-address-information...
3932 </btpr:target-additional-information>
3933 <btpr:responder-address>
3934 ...address...
3935 </btpr:responder-address>
3936 <btpr:responder-identifier>...hexstring...</btpr:responder-
3937 identifier>
3938
3939 <btpr:status-value> created|enrolling|active|resigning|
3940 resigned|preparing|prepared|
3941 confirming|confirmed|cancelling|cancelled|
3942 cancel-contradiction|confirm-contradiction|
3943 hazard|contradicted|unknown|inaccessible</btpr:status-
3944 value>

```

```
3945 <btq:qualifiers>?  
3946 ...qualifiers...  
3947 </btq:qualifiers>  
3948 </btq:status>
```

FAULT

```
3951  
3952 <btq:fault-id?>  
3953 <btq:target-additional-information>  
3954 ...additional-address-information...  
3955 </btq:target-additional-information>  
3956 <btq:superior-identifier>...hexstring...</btq:superior-  
3957 identifier?>  
3958 <btq:inferior-identifier>...hexstring...</btq:inferior-  
3959 identifier?>  
3960 <btq:fault-type>...fault-type-name...</btq:fault-type>  
3961 <btq:fault-data>...fault-data...</btq:fault-data?>  
3962 <btq:qualifiers>?  
3963 ...qualifiers...  
3964 </btq:qualifiers>  
3965 </btq:fault>
```

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

```
3971 ogeneral  
3972 ounknown-parameter  
3973 owrong-state  
3974 ocommunication-failure  
3975 oinvalid-superior  
3976 oduplicate-inferior  
3977 ounknown-inferior
```

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

Fault data can take on various forms:

Free text:

```
3987 <btq:fault-data>...string-data...</btq:fault-data>
```

Identifier:

```
3991 <btq:fault-data>...hexstring...</btq:fault-data>
```

3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004

Inferior Identity:

```
<btpr:fault-data>  
  <btpr:inferior-address> +  
  ...address...  
</btpr:inferior-address>  
  <btpr:inferior-identifier>...hexstring...</btpr:inferior-  
  identifier>  
</btpr:fault-data>
```

4005
4006
4007
4008

Standard qualifiers

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

4009
4010

Transaction timelimit

```
<btpr:transaction-timelimit>  
  <btpr:timelimit>  
  ...time in seconds...  
</btpr:timelimit>  
</btpr:transaction-timelimit>
```

4011
4012
4013
4014
4015
4016

Inferior timeout

```
<btpr:inferior-timeout>  
  <btpr:timeout>  
  ...time in seconds...  
</btpr:timeout>  
  <btpr:intended-decision>confirm|cancel</btpr:intended-decision>  
</btpr:inferior-timeout>
```

4017
4018
4019
4020
4021
4022
4023
4024

Minimum inferior timeout

```
<btpr:minimum-inferior-timeout>  
  <btpr:minimum-timeout>  
  ...time in seconds...  
</btpr:minimum-timeout>  
</btpr:minimum-inferior-timeout>
```

4025
4026
4027
4028
4029
4030
4031

Inferior name

```
<btpr:inferior-name>  
  <btpr:inferior-name>  
  ...string...  
</btpr:inferior-name>  
</btpr:inferior-name>
```

4032
4033
4034
4035
4036
4037
4038

Compounding of Messages

4039
4040

4041 Relating BTP to one another, in a “group” is represented by containing them within the
4042 `btp:related_group` element, with the related messages as child elements. The processing for
4043 the group is defined in the section “Groups – combinations of related messages”. For example

```
4044 <btp:related_group>  
4045   <btp:context-reply>  
4046     ...<completion-status>related</completion-status> ...  
4047   </btp:context-reply>  
4048   <btp:enrol>...</btp:enrol>  
4049   <btp:prepared>...</btp:prepared>  
4050 </btp:related_group>
```

4051
4052
4053 If the rules for the group state that the target address of the abstract message is omitted, the
4054 corresponding `target-address-information` element shall be absent in the message in the
4055 `related_group`. The carrier protocol binding specifies how a relation between application and
4056 BTP messages is represented.

4057
4058 Bundling (semantically insignificant combination) of BTP messages and related groups is
4059 indicated with the "`btp:messages`" element, with the bundled messages and related groups as
4060 child elements. For example (confirming one and cancelling another inferiors of a cohesion):

```
4061 <btp:messages>  
4062   <btp:confirm>...</btp:confirm>  
4063   <btp:cancel>...</btp:cancel>  
4064 </btp:messages>
```

4065
4066
4067
4068

4068

XML Schemas

4069

4070

4071

XML schema for BTP messages

4072

4073

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

4074

```
<schema
```

4075

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

4076

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

4077

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

4078

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

4079

4080

4081

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

4082

4083

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

4084

```
    <simpleContent>
```

4085

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

4086

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

4087

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

4088

```
      </extension>
```

4089

```
    </simpleContent>
```

4090

```
  </complexType>
```

4091

4092

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

4093

4094

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

4095

```
    <complexType>
```

4096

```
      <sequence>
```

4097

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

4098

```
      </sequence>
```

4099

```
    </complexType>
```

4100

```
  </element>
```

4101

4102

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

4103

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

4104

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

4105

```
  -->
```

4106

4107

4108

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

4109

4110

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

4111

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

4112

```
  </simpleType>
```

4113

4114

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

4115

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

4116

```
  </simpleType>
```

4117

4118

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

4119

```
    <sequence>
```

```

4120         <element name="binding-name" type="anyURI"/>
4121         <element name="binding-address" type="string"/>
4122         <element name="additional-information" type="btp:additional-
4123 information" minOccurs="0" />
4124     </sequence>
4125 </complexType>
4126
4127     <simpleType name="superior-type">
4128         <restriction base="string">
4129             <enumeration value="cohesion"/>
4130             <enumeration value="atom"/>
4131         </restriction>
4132     </simpleType>
4133
4134     <simpleType name="transaction-type">
4135         <restriction base="string">
4136             <enumeration value="cohesion"/>
4137             <enumeration value="atom"/>
4138         </restriction>
4139     </simpleType>
4140
4141
4142     <!-- Compounding -->
4143
4144     <element name="messages">
4145         <complexType>
4146             <sequence>
4147                 <element ref="btp:message" minOccurs="0"
4148 maxOccurs="unbounded"/>
4149             </sequence>
4150         </complexType>
4151     </element>
4152
4153     <element name="related-group" substitutionGroup="btp:message">
4154         <complexType>
4155             <sequence>
4156                 <element ref="btp:message" minOccurs="0"
4157 maxOccurs="unbounded"/>
4158             </sequence>
4159         </complexType>
4160     </element>
4161
4162
4163     <!-- Message set -->
4164
4165     <element name="message" abstract="true" />
4166
4167     <element name="context" substitutionGroup="btp:message">
4168         <complexType>
4169             <sequence>
4170                 <element name="superior-address" type="btp:address"
4171 maxOccurs="unbounded"/>
4172                 <element name="superior-identifier" type="btp:identifier"/>

```

```

4173         <element name="reply-address" type="btp:address"
4174 minOccurs="0"/>
4175         <element name="superior-type" type="btp:superior-type"/>
4176         <element ref="btp:qualifiers" minOccurs="0"/>
4177     </sequence>
4178     <attribute name="id" type="ID" use="optional"/>
4179 </complexType>
4180 </element>
4181
4182     <element name="context-reply" substitutionGroup="btp:message">
4183     <complexType>
4184     <sequence>
4185         <element name="target-additional-information"
4186 type="btp:additional-information" minOccurs="0"/>
4187         <element name="superior-identifier" type="btp:identifier"/>
4188         <element name="completion-status">
4189         <simpleType>
4190         <restriction base="string">
4191         <enumeration value="completed"/>
4192         <enumeration value="related"/>
4193         <enumeration value="repudiated"/>
4194         </restriction>
4195         </simpleType>
4196         </element>
4197         <element ref="btp:qualifiers" minOccurs="0"/>
4198     </sequence>
4199     <attribute name="id" type="ID"/>
4200 </complexType>
4201 </element>
4202
4203     <element name="request-status" substitutionGroup="btp:message">
4204     <complexType>
4205     <sequence>
4206         <element name="target-additional-information"
4207 type="btp:additional-information" minOccurs="0"/>
4208         <element name="reply-address" type="btp:address"
4209 minOccurs="0"/>
4210         <element name="target-identifier" type="btp:identifier"/>
4211         <element ref="btp:qualifiers" minOccurs="0"/>
4212     </sequence>
4213     <attribute name="id" type="ID"/>
4214 </complexType>
4215 </element>
4216
4217     <element name="status" substitutionGroup="btp:message">
4218     <complexType>
4219     <sequence>
4220         <element name="target-additional-information"
4221 type="btp:additional-information" minOccurs="0"/>
4222         <element name="responders-identifier"
4223 type="btp:identifier"/>
4224         <element name="status-value">
4225         <simpleType>

```

```

4226         <restriction base="string">
4227             <enumeration value="created"/>
4228             <enumeration value="enrolling"/>
4229             <enumeration value="active"/>
4230             <enumeration value="resigning"/>
4231             <enumeration value="resigned"/>
4232             <enumeration value="preparing"/>
4233             <enumeration value="prepared"/>
4234             <enumeration value="confirming"/>
4235             <enumeration value="confirmed"/>
4236             <enumeration value="cancelling"/>
4237             <enumeration value="cancelled"/>
4238             <enumeration value="cancel-contradiction"/>
4239             <enumeration value="confirm-contradiction"/>
4240             <enumeration value="hazard"/>
4241             <enumeration value="contradicted"/>
4242             <enumeration value="unknown"/>
4243             <enumeration value="inaccessible"/>
4244         </restriction>
4245     </simpleType>
4246 </element>
4247     <element ref="btp:qualifiers" minOccurs="0"/>
4248 </sequence>
4249     <attribute name="id" type="ID"/>
4250 </complexType>
4251 </element>
4252
4253     <element name="fault" substitutionGroup="btp:message">
4254         <complexType>
4255             <sequence>
4256                 <element name="target-additional-information"
4257 type="btp:additional-information" minOccurs="0"/>
4258                 <element name="superior-identifier" type="btp:identifier"
4259 minOccurs="0"/>
4260                 <element name="inferior-identifier" type="btp:identifier"
4261 minOccurs="0"/>
4262                 <element name="fault-type">
4263                     <simpleType>
4264                         <restriction base="string">
4265                             <enumeration value="communication-failure"/>
4266                             <enumeration value="duplicate-inferior"/>
4267                             <enumeration value="general"/>
4268                             <enumeration value="invalid-decider"/>
4269                             <enumeration value="invalid-inferior"/>
4270                             <enumeration value="invalid-superior"/>
4271                             <enumeration value="status-refused"/>
4272                             <enumeration value="invalid-terminator"/>
4273                             <enumeration value="unknown-parameter"/>
4274                             <enumeration value="unknown-transaction"/>
4275                             <enumeration value="unsupported-qualifier"/>
4276                             <enumeration value="wrong-state"/>
4277                         </restriction>
4278                     </simpleType>

```



```

4279         </element>
4280         <element name="fault-data" type="anyType" minOccurs="0"/>
4281         <element ref="btp:qualifiers" minOccurs="0"/>
4282     </sequence>
4283     <attribute name="id" type="ID"/>
4284 </complexType>
4285 </element>
4286
4287     <element name="enrol" substitutionGroup="btp:message">
4288         <complexType>
4289             <sequence>
4290                 <element name="target-additional-information"
4291 type="btp:additional-information" minOccurs="0"/>
4292                 <element name="superior-identifier" type="btp:identifier"/>
4293                 <element name="reply-requested" type="boolean"/>
4294                 <element name="reply-address" type="btp:address"
4295 minOccurs="0"/>
4296                 <element name="inferior-address" type="btp:address"
4297 minOccurs="1" maxOccurs="unbounded"/>
4298                 <element name="inferior-identifier" type="btp:identifier"/>
4299                 <element ref="btp:qualifiers" minOccurs="0"/>
4300             </sequence>
4301             <attribute name="id" type="ID"/>
4302         </complexType>
4303     </element>
4304
4305
4306     <element name="enrolled" substitutionGroup="btp:message">
4307         <complexType>
4308             <sequence>
4309                 <element name="target-additional-information"
4310 type="btp:additional-information" minOccurs="0"/>
4311                 <element name="inferior-identifier" type="btp:identifier"/>
4312                 <element ref="btp:qualifiers" minOccurs="0"/>
4313             </sequence>
4314             <attribute name="id" type="ID"/>
4315         </complexType>
4316     </element>
4317
4318     <element name="resign" substitutionGroup="btp:message">
4319         <complexType>
4320             <sequence>
4321                 <element name="target-additional-information"
4322 type="btp:additional-information" minOccurs="0"/>
4323                 <element name="superior-identifier" type="btp:identifier"/>
4324                 <element name="inferior-identifier" type="btp:identifier"/>
4325                 <element name="response-requested" type="boolean"/>
4326                 <element ref="btp:qualifiers" minOccurs="0"/>
4327             </sequence>
4328             <attribute name="id" type="ID"/>
4329         </complexType>
4330     </element>
4331

```

```

4332     <element name="resigned" substitutionGroup="btp:message">
4333         <complexType>
4334             <sequence>
4335                 <element name="target-additional-information"
4336 type="btp:additional-information" minOccurs="0"/>
4337                 <element name="inferior-identifier" type="btp:identifier"/>
4338                 <element ref="btp:qualifiers" minOccurs="0"/>
4339             </sequence>
4340             <attribute name="id" type="ID"/>
4341         </complexType>
4342     </element>
4343
4344     <element name="prepare" substitutionGroup="btp:message">
4345         <complexType>
4346             <sequence>
4347                 <element name="target-additional-information"
4348 type="btp:additional-information" minOccurs="0"/>
4349                 <element name="inferior-identifier" type="btp:identifier"/>
4350                 <element ref="btp:qualifiers" minOccurs="0"/>
4351             </sequence>
4352             <attribute name="id" type="ID"/>
4353         </complexType>
4354     </element>
4355
4356     <element name="prepared" substitutionGroup="btp:message">
4357         <complexType>
4358             <sequence>
4359                 <element name="target-additional-information"
4360 type="btp:additional-information" minOccurs="0"/>
4361                 <element name="superior-identifier" type="btp:identifier"/>
4362                 <element name="inferior-identifier" type="btp:identifier"/>
4363                 <element name="default-is-cancel" type="boolean"/>
4364                 <element ref="btp:qualifiers" minOccurs="0"/>
4365             </sequence>
4366             <attribute name="id" type="ID"/>
4367         </complexType>
4368     </element>
4369
4370     <element name="confirm" substitutionGroup="btp:message">
4371         <complexType>
4372             <sequence>
4373                 <element name="target-additional-information"
4374 type="btp:additional-information" minOccurs="0"/>
4375                 <element name="inferior-identifier" type="btp:identifier"/>
4376                 <element ref="btp:qualifiers" minOccurs="0"/>
4377             </sequence>
4378             <attribute name="id" type="ID"/>
4379         </complexType>
4380     </element>
4381
4382     <element name="confirmed" substitutionGroup="btp:message">
4383         <complexType>
4384             <sequence>

```

```

4385         <element name="target-additional-information"
4386 type="btp:additional-information" minOccurs="0"/>
4387         <element name="superior-identifier" type="btp:identifier"/>
4388         <element name="inferior-identifier" type="btp:identifier"/>
4389         <element name="confirmed-received" type="boolean"/>
4390         <element ref="btp:qualifiers" minOccurs="0"/>
4391     </sequence>
4392     <attribute name="id" type="ID"/>
4393 </complexType>
4394 </element>
4395
4396     <element name="cancel" substitutionGroup="btp:message">
4397     <complexType>
4398     <sequence>
4399         <element name="target-additional-information"
4400 type="btp:additional-information" minOccurs="0"/>
4401         <element name="inferior-identifier" type="btp:identifier"/>
4402         <element name="reply-address" type="btp:address"
4403 minOccurs="0"/>
4404         <element ref="btp:qualifiers" minOccurs="0"/>
4405     </sequence>
4406     <attribute name="id" type="ID"/>
4407 </complexType>
4408 </element>
4409
4410     <element name="cancelled" substitutionGroup="btp:message">
4411     <complexType>
4412     <sequence>
4413         <element name="target-additional-information"
4414 type="btp:additional-information" minOccurs="0"/>
4415         <element name="superior-identifier" type="btp:identifier"/>
4416         <element name="inferior-identifier" type="btp:identifier"
4417 minOccurs="0"/>
4418         <element ref="btp:qualifiers" minOccurs="0"/>
4419     </sequence>
4420     <attribute name="id" type="ID"/>
4421 </complexType>
4422 </element>
4423
4424     <element name="confirm-one-phase" substitutionGroup="btp:message">
4425     <complexType>
4426     <sequence>
4427         <element name="target-additional-information"
4428 type="btp:additional-information" minOccurs="0"/>
4429         <element name="inferior-identifier" type="btp:identifier"/>
4430         <element name="report-hazard" type="boolean"/>
4431         <element ref="btp:qualifiers" minOccurs="0"/>
4432     </sequence>
4433     <attribute name="id" type="ID"/>
4434 </complexType>
4435 </element>
4436
4437     <element name="hazard" substitutionGroup="btp:message">

```

```

4438     <complexType>
4439         <sequence>
4440             <element name="target-additional-information"
4441 type="btp:additional-information" minOccurs="0"/>
4442             <element name="superior-identifier" type="btp:identifier"/>
4443             <element name="inferior-identifier" type="btp:identifier"/>
4444             <element name="level">
4445                 <simpleType>
4446                     <restriction base="string">
4447                         <enumeration value="mixed"/>
4448                         <enumeration value="possible"/>
4449                     </restriction>
4450                 </simpleType>
4451             </element>
4452             <element ref="btp:qualifiers" minOccurs="0"/>
4453         </sequence>
4454         <attribute name="id" type="ID"/>
4455     </complexType>
4456 </element>
4457
4458     <element name="contradiction" substitutionGroup="btp:message">
4459         <complexType>
4460             <sequence>
4461                 <element name="target-additional-information"
4462 type="btp:additional-information" minOccurs="0"/>
4463                 <element name="inferior-identifier" type="btp:identifier"/>
4464                 <element ref="btp:qualifiers" minOccurs="0"/>
4465             </sequence>
4466             <attribute name="id" type="ID"/>
4467         </complexType>
4468     </element>
4469
4470     <element name="superior-state" substitutionGroup="btp:message">
4471         <complexType>
4472             <sequence>
4473                 <element name="target-additional-information"
4474 type="btp:additional-information" minOccurs="0"/>
4475                 <element name="inferior-identifier" type="btp:identifier"/>
4476                 <element name="status">
4477                     <simpleType>
4478                         <restriction base="string">
4479                             <enumeration value="active"/>
4480                             <enumeration value="prepared-received"/>
4481                             <enumeration value="inaccessible"/>
4482                             <enumeration value="unknown"/>
4483                         </restriction>
4484                     </simpleType>
4485                 </element>
4486                 <element name="reply-requested" type="boolean"/>
4487                 <element ref="btp:qualifiers" minOccurs="0"/>
4488             </sequence>
4489             <attribute name="id" type="ID"/>
4490         </complexType>

```

```

4491     </element>
4492
4493     <element name="inferior-state" substitutionGroup="btp:message">
4494         <complexType>
4495             <sequence>
4496                 <element name="target-additional-information"
4497 type="btp:additional-information" minOccurs="0"/>
4498                 <element name="superior-identifier" type="btp:identifier"/>
4499                 <element name="inferior-identifier" type="btp:identifier"/>
4500                 <element name="status">
4501                     <simpleType>
4502                         <restriction base="string">
4503                             <enumeration value="active"/>
4504                             <enumeration value="inaccessible"/>
4505                             <enumeration value="unknown"/>
4506                         </restriction>
4507                     </simpleType>
4508                 </element>
4509                 <element name="reply-requested" type="boolean"/>
4510                 <element ref="btp:qualifiers" minOccurs="0"/>
4511             </sequence>
4512             <attribute name="id" type="ID"/>
4513         </complexType>
4514     </element>
4515
4516     <element name="redirect" substitutionGroup="btp:message">
4517         <complexType>
4518             <sequence>
4519                 <element name="target-additional-information"
4520 type="btp:additional-information" minOccurs="0"/>
4521                 <element name="superior-identifier" type="btp:identifier"
4522 minOccurs="0"/>
4523                 <element name="inferior-identifier" type="btp:identifier"
4524 />
4525                 <element name="old-address" type="btp:address"
4526 maxOccurs="unbounded"/>
4527                 <element name="new-address" type="btp:address"
4528 maxOccurs="unbounded"/>
4529                 <element ref="btp:qualifiers" minOccurs="0"/>
4530             </sequence>
4531             <attribute name="id" type="ID"/>
4532         </complexType>
4533     </element>
4534
4535
4536     <element name="begin" substitutionGroup="btp:message">
4537         <complexType>
4538             <sequence>
4539                 <element name="target-additional-information"
4540 type="btp:additional-information" minOccurs="0"/>
4541                 <element name="reply-address" type="btp:address"
4542 minOccurs="0"/>
4543                 <element name="transaction-type" type="btp:superior-type"/>

```

```

4544         <element ref="btp:qualifiers" minOccurs="0"/>
4545     </sequence>
4546     <attribute name="id" type="ID"/>
4547 </complexType>
4548 </element>
4549
4550     <element name="begun" substitutionGroup="btp:message">
4551         <complexType>
4552             <sequence>
4553                 <element name="target-additional-information"
4554 type="btp:additional-information" minOccurs="0"/>
4555                 <element name="decider-address" type="btp:address"
4556 minOccurs="0" maxOccurs="unbounded"/>
4557                 <element name="transaction-identifier"
4558 type="btp:identifier" minOccurs="0"/>
4559                 <element name="inferior-handle" type="btp:identifier"
4560 minOccurs="0"/>
4561                 <element name="inferior-address" type="btp:address"
4562 minOccurs="0" maxOccurs="unbounded"/>
4563                 <element ref="btp:qualifiers" minOccurs="0"/>
4564             </sequence>
4565             <attribute name="id" type="ID"/>
4566         </complexType>
4567     </element>
4568
4569     <element name="prepare-inferiors" substitutionGroup="btp:message">
4570         <complexType>
4571             <sequence>
4572                 <element name="target-additional-information"
4573 type="btp:additional-information" minOccurs="0"/>
4574                 <element name="reply-address" type="btp:address"
4575 minOccurs="0"/>
4576                 <element name="transaction-identifier"
4577 type="btp:identifier"/>
4578                 <element name="inferiors-list" minOccurs="0">
4579                     <complexType>
4580                         <sequence>
4581                             <element name="inferior-handle"
4582 type="btp:identifier" maxOccurs="unbounded"/>
4583                         </sequence>
4584                     </complexType>
4585                 </element>
4586                 <element ref="btp:qualifiers" minOccurs="0"/>
4587             </sequence>
4588             <attribute name="id" type="ID"/>
4589         </complexType>
4590     </element>
4591
4592     <element name="confirm-transaction" substitutionGroup="btp:message">
4593         <complexType>
4594             <sequence>
4595                 <element name="target-additional-information"
4596 type="btp:additional-information" minOccurs="0"/>

```

```

4597         <element name="reply-address" type="btp:address"
4598 minOccurs="0"/>
4599         <element name="transaction-identifier"
4600 type="btp:identifier"/>
4601         <element name="inferiors-list" minOccurs="0">
4602             <complexType>
4603                 <sequence>
4604                     <element name="inferior-handle"
4605 type="btp:identifier" maxOccurs="unbounded"/>
4606                 </sequence>
4607             </complexType>
4608         </element>
4609         <element name="report-hazard" type="boolean"/>
4610         <element ref="btp:qualifiers" minOccurs="0"/>
4611     </sequence>
4612     <attribute name="id" type="ID"/>
4613 </complexType>
4614 </element>
4615
4616     <element name="transaction-confirmed" substitutionGroup="btp:message">
4617         <complexType>
4618             <sequence>
4619                 <element name="target-additional-information"
4620 type="btp:additional-information" minOccurs="0"/>
4621                 <element name="transaction-identifier"
4622 type="btp:identifier"/>
4623                 <element ref="btp:qualifiers" minOccurs="0"/>
4624             </sequence>
4625             <attribute name="id" type="ID"/>
4626         </complexType>
4627     </element>
4628
4629     <element name="cancel-transaction" substitutionGroup="btp:message">
4630         <complexType>
4631             <sequence>
4632                 <element name="target-additional-information"
4633 type="btp:additional-information" minOccurs="0"/>
4634                 <element name="reply-address" type="btp:address"
4635 minOccurs="0"/>
4636                 <element name="transaction-identifier"
4637 type="btp:identifier"/>
4638                 <element name="report-hazard" type="boolean"/>
4639                 <element ref="btp:qualifiers" minOccurs="0"/>
4640             </sequence>
4641             <attribute name="id" type="ID"/>
4642         </complexType>
4643     </element>
4644
4645     <element name="cancel-inferiors" substitutionGroup="btp:message">
4646         <complexType>
4647             <sequence>
4648                 <element name="target-additional-information"
4649 type="btp:additional-information" minOccurs="0"/>

```

```

4650         <element name="reply-address" type="btp:address"
4651 minOccurs="0"/>
4652         <element name="transaction-identifier"
4653 type="btp:identifier" minOccurs="0"/>
4654         <element name="inferiors-list">
4655             <complexType>
4656                 <sequence>
4657                     <element name="inferior-handle"
4658 type="btp:identifier" maxOccurs="unbounded"/>
4659                 </sequence>
4660             </complexType>
4661         </element>
4662         <element ref="btp:qualifiers" minOccurs="0"/>
4663     </sequence>
4664     <attribute name="id" type="ID"/>
4665 </complexType>
4666 </element>
4667
4668     <element name="transaction-cancelled" substitutionGroup="btp:message">
4669         <complexType>
4670             <sequence>
4671                 <element name="target-additional-information"
4672 type="btp:additional-information" minOccurs="0"/>
4673                 <element name="transaction-identifier"
4674 type="btp:identifier"/>
4675                 <element ref="btp:qualifiers" minOccurs="0"/>
4676             </sequence>
4677             <attribute name="id" type="ID"/>
4678         </complexType>
4679     </element>
4680
4681     <element name="request-inferior-statuses"
4682 substitutionGroup="btp:message">
4683         <complexType>
4684             <sequence>
4685                 <element name="target-additional-information"
4686 type="btp:additional-information" minOccurs="0"/>
4687                 <element name="reply-address" type="btp:address"
4688 minOccurs="0"/>
4689                 <element name="target-identifier" type="btp:identifier"/>
4690                 <element name="inferiors-list" minOccurs="0">
4691                     <complexType>
4692                         <sequence>
4693                             <element name="inferior-handle"
4694 type="btp:identifier" maxOccurs="unbounded"/>
4695                         </sequence>
4696                     </complexType>
4697                 </element>
4698                 <element ref="btp:qualifiers" minOccurs="0"/>
4699             </sequence>
4700             <attribute name="id" type="ID"/>
4701         </complexType>
4702     </element>

```



```

4703
4704     <element name="inferior-statuses" substitutionGroup="btp:message">
4705         <complexType>
4706             <sequence>
4707                 <element name="target-additional-information"
4708 type="btp:additional-information" minOccurs="0"/>
4709                 <element name="responders-identifier"
4710 type="btp:identifier"/>
4711                 <element name="status-list">
4712                     <complexType>
4713                         <sequence>
4714                             <element name="status-item" maxOccurs="unbounded">
4715                                 <complexType>
4716                                     <sequence>
4717                                         <element name="inferior-handle"
4718 type="btp:identifier"/>
4719                                         <element name="status">
4720                                             <simpleType>
4721                                                 <restriction base="string">
4722                                                     <enumeration value="active"/>
4723                                                     <enumeration value="resigned"/>
4724                                                     <enumeration value="preparing"/>
4725                                                     <enumeration value="prepared"/>
4726                                                     <enumeration value="autonomously-confirmed"/>
4727                                                     <enumeration value="autonomously-cancelled"/>
4728                                                     <enumeration value="confirming"/>
4729                                                     <enumeration value="confirmed"/>
4730                                                     <enumeration value="cancelling"/>
4731                                                     <enumeration value="cancelled"/>
4732                                                     <enumeration value="cancel-contradiction"/>
4733                                                     <enumeration value="confirm-contradiction"/>
4734                                                     <enumeration value="hazard"/>
4735                                                     <enumeration value="invalid"/>
4736                                                 </restriction>
4737                                             </simpleType>
4738                                         </element>
4739                                         <element ref="btp:qualifiers" minOccurs="0"/>
4740                                     </sequence>
4741                                 </complexType>
4742                             </element>
4743                         </sequence>
4744                     </complexType>
4745                 </element>
4746                 <element ref="btp:qualifiers" minOccurs="0"/>
4747             </sequence>
4748             <attribute name="id" type="ID"/>
4749         </complexType>
4750     </element>
4751
4752
4753 </schema>
4754

```

XML schema for standard qualifiers

4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806

```
<?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>
```

```
4807         <element name="minimum-timeout"  
4808         type="nonNegativeInteger"/>  
4809     </sequence>  
4810 </extension>  
4811 </complexContent>  
4812 </complexType>  
4813 </element>  
4814  
4815     <element name="inferior-name" substitutionGroup="btp:qualifier">  
4816         <complexType>  
4817             <complexContent>  
4818                 <extension base="btp:qualifier-type">  
4819                     <sequence>  
4820                         <element name="inferior-name" type="string"/>  
4821                     </sequence>  
4822                 </extension>  
4823             </complexContent>  
4824         </complexType>  
4825     </element>  
4826  
4827 </schema>  
4828
```

4828
4829

4830 **Carrier Protocol Bindings**

4831

4832 The notion of bindings is introduced to act as the glue between the BTP messages and an
4833 underlying transport. A binding specification must define various particulars of how the BTP
4834 messages are carried and some aspects of how the related application messages are carried.
4835 This document specifies two bindings: a SOAP binding and a SOAP + Attachments binding.
4836 However, other bindings could be specified by the Oasis BTP technical committee or by a
4837 third party. For example, in the future a binding might exist to put a BTP message directly on
4838 top of HTTP without the use of SOAP, or a closed community could define their own
4839 binding. To ensure that such specifications are complete, the Binding Proforma defines the
4840 information that must be included in a binding specification.
4841

4842 **Carrier Protocol Binding Proforma**

4843

4844 A BTP carrier binding specification should provide the following information:
4845

4846 **Binding name:** A name for the binding, as used in the “binding name” field of BTP
4847 addresses (and available for declaring the capabilities of an implementation). Binding
4848 specified in this document, and future revisions of this document have binding names that are
4849 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).
4850 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in
4851 this document use numbers to identify the version of the binding, not the version(s) of the
4852 carrier protocol.
4853

4854 **Binding address format:** This section states the format of the “binding address” field of a
4855 BTP address for this binding. For many bindings, this will be a URL of some kind; for other
4856 bindings it may be some other form
4857

4858 **BTP message representation:** This section will define how BTP messages are represented.
4859 For many bindings, the BTP message syntax will be as specified in the XML schema defined
4860 in this document, and the normal string encoding of that XML will be used.
4861

4862 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that
4863 are not related to application messages are sent in either direction between Superior and
4864 Inferior. (i.e. those messages sent directly between BTP actors). This mapping need not be
4865 symmetric (i.e. Superior to Inferior may differ to some degree to Inferior to Superior). The
4866 mapping may define particular rules for particular BTP messages, or messages with particular
4867 parameter values (e.g. the FAULT message with “fault-type” “CommunicationFailure” will
4868 typically not be sent as a BTP message). The mapping states any constraints or requirements
4869 on which BTP may or must be bundled together by compounding.
4870

4871 **Mapping for BTP messages related to application messages:** This section will define how
4872 BTP messages that are related to application messages are sent. A binding specification may
4873 defer details of this to a particular application (e.g. a mapping specification could just say

4874 “the CONTEXT may be carried as a parameter of an application invocation”). Alternatively,
4875 the binding may specify a general method that represents the relationship between application
4876 and BTP messages.

4877
4878 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly
4879 but are treated as implicit in application messages or other BTP messages. This may depend
4880 on particular parameter values of the BTP messages or the application messages.

4881
4882 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
4883 protocol and of BTP shall be defined. This may include definition of which carrier protocol
4884 exceptions are equivalent to a FAULT/communication-failure message.

4885
4886 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.
4887 If BTP addresses with different bindings are be considered to match (for purposes of
4888 identifying the peer Superior/Inferior and redirection), this should be specified here.

4889
4890 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are
4891 imposed by use of this binding should be listed. This would include limitations on which
4892 messages can be sent, which event sequences are supported and restrictions on parameter
4893 values. Such limitations may reduce the usefulness of an implementation, but may be
4894 appropriate in certain environments.

4895
4896 **Other:** Other features of the binding, especially any that will potentially affect interoperation
4897 should be specified here. This may include restrictions or requirements on the use or support
4898 of optional carrier parameters or mechanisms.

4899

4900 **Bindings for request/response carrier protocols**

4901

4902 BTP does not generally follow request/response pattern. In particular, on the outcome
4903 relationship either side may initiate a message – this is an essential part of the presume-abort
4904 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
4905 messages, especially in the control relationship, that do have a request/response pattern.
4906 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The
4907 specification of a binding specification to a request/response carrier protocol needs to state
4908 what rules apply – which messages can be carried by requests, which by responses. The
4909 simplest rule is to send all BTP messages on requests, and let the carrier responses travel back
4910 empty. This would be inefficient in use of network resources, and possibly inconvenient
4911 when used for the BTP request/response pairs.

4912

4913 This section defines a set of rules that allow more efficient use of the carrier, while allowing
4914 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the
4915 carrier response. These rules are specified in this section to enable binding specifications to
4916 reference them, without requiring each binding specification to repeat similar information.

4917

4918 A binding to a request/response carrier is not required to use these rules. It may define other
4919 rules.

4920

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

Request/response exploitation rules

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

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

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

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

The rules:

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

- 4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
- e) A BTP actor that receives a carrier protocol request carrying BTP messages that do have a reply address, or which initiate processing that produces BTP messages whose target binding address matches the origin of the request, **may** freely choose whether to use the carrier protocol response for the replies, or to send back an “empty carrier protocol response”, and send the BTP replies in a separately initiated carrier protocol request. The characteristics of an “empty carrier protocol response” shall be stated in the particular binding specification.
 - f) A BTP actor that sends BTP messages on a carrier protocol request **must** be able to accept returning BTP messages on the corresponding carrier protocol response and, if the actor has offered an address on which it will receive carrier requests, must be able to accept “replying” BTP messages on a separate carrier protocol request.

4981 SOAP Binding

4982
4983 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
4984 specification, using the SOAP literal messaging style conventions. If no application message
4985 is sent at the same time, the BTP messages are contained within the SOAP Body element. If
4986 application messages are sent, the BTP messages are contained in the SOAP Header element.
4987

4988 **Binding name:** soap-http-1
4989

4990 **Binding address format:** shall be a URL, of type HTTP.
4991

4992 **BTP message representation:** The string representation of the XML, as specified in the
4993 XML schema defined in this document shall be used. The BTP XML messages are embedded
4994 in the SOAP message without the use of any specific encoding rules (literal style SOAP
4995 message); hence the encodingStyle attribute need not be set or can be set to an empty string.
4996

4997 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be
4998 used.
4999

5000 BTP messages sent on an HTTP request or HTTP response which is not carrying an
5001 application message, the messages are contained in a single btp:messages element which is
5002 the immediate child element of the SOAP Body element.
5003

5004 An “empty carrier protocol response” sent after receiving an HTTP request containing a
5005 btp:messages element in the SOAP Body and the implementation BTP actor chooses just to
5006 reply at the lower level (and when the request/response exploitation rules allow an empty
5007 carrier protocol response), shall be any of:

- 5008 a) an empty HTTP response
- 5009 b) an HTTP response containing an empty SOAP Envelope
- 5010 c) an HTTP response containing a SOAP Envelope containing a single, empty
5011 btp:messages element.
5012

5013 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they
5014 have no effect on the BTP sequence (other than indicating that the earlier sending did not
5015 cause a communication failure.)

5016
5017
5018

5019 If an application message is being sent at the same time, the mapping for related messages
5020 shall be used, as if the BTP messages were related to the application message. (There is no
5021 ambiguity in whether the BTP messages are related, because only CONTEXT and ENROL
5022 can be related to an application message.)

5023

5024 **Mapping for BTP messages related to application messages:** All BTP messages sent with
5025 an application message, whether related to the application message or not, shall be sent in a
5026 single `btpr:messages` element in the SOAP Header. There shall be precisely one `btpr:messages`
5027 element in the SOAP Header.

5028

5029 The “request/response exploitation” rules shall apply to the BTP messages carried in the
5030 SOAP Header, as if they had been carried in a SOAP Body, unrelated to an application
5031 message, sent to the same binding address.

5032

Note – The application protocol itself (which is using the SOAP Body) may
5033 use the SOAP RPC or document approach – this is determined by the
5034 application.

5035

5036 Only CONTEXT and ENROL messages are related (&) to application messages. If there is
5037 only one CONTEXT or one ENROL message present in the SOAP Header, it is assumed to
5038 be related to the whole of the application message in the SOAP Body. If there are multiple
5039 CONTEXT or ENROL messages, any relation of these BTP messages shall be indicated by
application specific means.

5040

Note 1 – An application protocol could use references to the ID values of the
5041 BTP messages to indicate relation between BTP CONTEXT or ENROL
5042 messages and the application message.

5043

Note 2 -- However indicated, what the relatedness means, or even whether it
5044 has any significance at all, is a matter for the application.

5045

5046 **Implicit messages:** A SOAP FAULT, or other communication failure received in response to
5047 a SOAP request that had a CONTEXT in the SOAP Header shall be treated as if a
5048 CONTEXT_REPLY/repudiated had been received. See also the discussion under “other”
5049 about the SOAP `mustUnderstand` attribute.

5050

5051 **Faults:** A SOAP FAULT or other communication failure shall be treated as
5052 FAULT/communication-failure.

5053

5054 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
5055 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
5056 http-1” if the binding address and additional information fields match.

5057
5058 **Limitations on BTP use:** None

5059
5060 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
5061 attribute. The SOAPAction HTTP header is left to be application specific when there are
5062 application messages in the SOAP Body, as an already existing web service that is being
5063 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
5064 header shall be omitted when the SOAP message carries only BTP messages in the SOAP
5065 Body.

5066
5067 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP
5068 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to
5069 determine whether any enrolments are necessary and replies with CONTEXT_REPLY as
5070 appropriate. The sender of the CONTEXT (and related application message) can use this to
5071 ensure that the application work is performed as part of the business transaction, assuming the
5072 receiver’s SOAP implementation supports the mustUnderstand attribute. If mustUnderstand if
5073 false, a receiver can ignore the CONTEXT (if BTP is not supported there), and no
5074 CONTEXT_REPLY will be returned. It is a local option on the sender (client) side whether
5075 the absence of a CONTEXT_REPLY is assumed to be equivalent to aCONTEXT_REPLY/ok
5076 (and the business transaction allowed to proceed to confirmation).

5077
5078 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to
5079 enforce these requirements.

5080 **Example scenario using SOAP binding**

5081
5082 The example below shows an application request with CONTEXT message sent from
5083 client.example.com (which includes the Superior) to services.example.com (Service).

```
5084  
5085  
5086 <soap:Envelope  
5087     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
5088     soap:encodingStyle="-">  
5089  
5090     <soap:Header>  
5091  
5092         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">  
5093             <btp:context superior-type="atom">  
5094                 <btp:superior-address>  
5095                     <btp:binding>soap-http-1</btp:binding>  
5096                     <btp:binding-  
5097 address>http://client.example.com/soaphandler</btp:binding-  
5098 address>  
5099                     <btp:additional-information>btpengine</btp:additional-  
5100 information>  
5101                 </btp:superior-address>
```

```

5102         <btp:superior-
5103 identifier>http://example.com/1001</btp:superior-identifier>
5104         <btp:qualifiers>
5105             <btpq:transaction-timelimit
5106 xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers"><btpq:timelimit>180
5107 0</btpq:timelimit></btpq:transaction-timelimit>
5108             </btp:qualifiers>
5109         </btp:context>
5110     </btp:messages>
5111
5112 </soap:Header>
5113
5114 <soap:Body>
5115
5116     <ns1:orderGoods
5117 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
5118         <custID>ABC8329045</custID>
5119         <itemID>224352</itemID>
5120         <quantity>5</quantity>
5121     </ns1:orderGoods>
5122
5123 </soap:Body>
5124
5125 </soap:Envelope>
5126

```

5127
5128 The example below shows CONTEXT_REPLY and a related ENROL message sent from
5129 services.example.com to client.example.com, in reply to the previous message. There is no
5130 application response, so the BTP messages are in the SOAP Body. The ENROL message
5131 does not contain the target-additional-information, since the grouping rules for
5132 CONTEXT_REPLY & ENROL omit the target address (the receiver of this example
5133 remembers the superior address from the original CONTEXT)
5134

```

5135 <soap:Envelope
5136     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
5137     soap:encodingStyle="">
5138
5139     <soap:Header>
5140     </soap:Header>
5141
5142     <soap:Body>
5143
5144         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
5145             <btp:related_group>
5146                 <btp:context-reply>
5147                     <btp:target-additional-information>btpengine</btp:target-
5148 additional-information>
5149                     <btp:superior-address>
5150                     <btp:binding>soap-http-1</btp:binding>
5151                     <btp:binding-address>
5152                         http://client.example.com/soaphandler
5153                     </btp:binding-address>

```

```

5154 <del><ctp:additional-information>  

5155 <del>    btpengine  

5156 <del></ctp:additional-information>  

5157 <del></ctp:superior-address>  

5158 <ctp:superior-  

5159 identifier>http://example.com/1001</ctp:superior-identifier>  

5160 <completion-status>related</completion-status>  

5161 </ctp:context-reply>  

5162  

5163 <ctp:enrol reply-requested="false">  

5164 <del>    <ctp:target-additional-  

5165 information>btpengine</ctp:target-additional-information>  

5166 <ctp:superior-identifier>  

5167 <del>    <a href="http://example.com/1001">http://example.com/1001  

5168 </ctp:superior-identifier>  

5169 <ctp:inferior-address>  

5170 <ctp:binding>soap-http-1</ctp:binding>  

5171 <ctp:binding-address>  

5172 <del>    <a href="http://services.example.com/soaphandler">http://services.example.com/soaphandler  

5173 </ctp:binding-address>  

5174 </ctp:inferior-address>  

5175 <ctp:inferior-identifier>  

5176 <del>    <a href="http://example.com/AAAB">http://example.com/AAAB  

5177 </ctp:inferior-identifier>  

5178 </ctp:enrol>  

5179  

5180 </ctp:related-group>  

5181  

5182 </ctp:messages>  

5183  

5184 </soap:Body>  

5185  

5186 </soap:Envelope>

```

5187
5188
5189

SOAP + Attachments Binding

5190

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

5191

Binding name: soap-attachments-http-1

5192

Binding address format: as for soap-http-1

5193

BTP message representation: As for soap-http-1

5194

Mapping for BTP messages (unrelated): As for “soap-http-1”, except the SOAP Envelope containing the SOAP Body containing the BTP messages shall be in a MIME body part, as

5195
5196
5197
5198
5199
5200
5201
5202
5203

5204 specified in [SOAP Messages with Attachments](#) specification. If an application message is
5205 being sent at the same time, the mapping for related messages for this binding shall be used,
5206 as if the BTP messages were related to the application message(s).

5207

5208 **Mapping for BTP messages related to application messages:** MIME packaging shall be
5209 used. One of the MIME multipart/related parts shall contain a SOAP Envelope, whose SOAP
5210 Headers element shall contain precisely one `btm:messages` element, containing any BTP
5211 messages. Any BTP CONTEXT in the `btm:messages` is considered to be related to the
5212 application message(s) in the SOAP Body, and to also any of the MIME parts referenced
5213 from the SOAP Body (using the “href” attribute).

5214

5215 **Implicit messages:** As for `soap-http-1`.

5216

5217 **Faults:** As for `soap-http-1`.

5218

5219 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
5220 string “`soap-http-1`” is considered to match one that has the binding string “`soap-`
5221 `attachements-http-1`” if the binding address and additional information fields match.

5222

5223 **Limitations on BTP use:** None

5224

5225 **Other:** As for `soap-http-1`

5226

5227 *Example using SOAP + Attachments binding*

5228

5229

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary;
type=text/xml;
    start="someID"
--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-ID: someID

<?xml version='1.0' ?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  soap-env:encodingStyle="
http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Header>
    <btm:messages xmlns:btm="urn:oasis:names:tc:BTP:xml">
      <btm:context superior-type="atom">
        <btm:superior-address>
          <btm:binding>soap-http-1</btm:binding>
          <btm:binding-address>
            http://client.example.com/soaphandler
          </btm:binding-address>
        </btm:superior-address>
      </btm:context>
    </btm:messages>
  </soap:Header>
</soap:Envelope>
```

5230

5231

5232

5233

5234

5235

5236

5237

5238

5239

5240

5241

5242

5243

5244

5245

5246

5247

5248

5249

5250

5251

5252

```

5253         </btp:superior-address>
5254         <btp:superior-
5255 identifier>http://example.com/1001</btp:superior-identifier>
5256         </btp:context>
5257         </btp:messages>
5258
5259     </soap:Header>
5260
5261     <soap:Body>
5262         <orderGoods href="cid:anotherID" />
5263     </soap:Body>
5264
5265 </soap:Envelope>
5266
5267 --MIME_boundary
5268 Content-Type: text/xml
5269 Content-ID: anotherID
5270
5271     <ns1:orderGoods
5272 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
5273         <custID>ABC8329045</custID>
5274         <itemID>224352</itemID>
5275         <quantity>5</quantity>
5276     </ns1:orderGoods>
5277
5278
5279 --MIME_boundary--
5280
5281

```

5282 Conformance

5283
5284 A BTP implementation need not implement all aspects of the protocol to be useful. The level
5285 of conformance of an implementation is defined by which roles it can support using the
5286 specified messages and carrier protocol bindings for interoperation with other
5287 implementations.

5288
5289 A partially conformant implementation may implement some roles in a non-interoperable
5290 way, giving that implementation's users comparable proprietary functionality.

5291
5292 The following Roles and Role Groups are used to define conformance:
5293

Role Group	Role
Initiator/Terminator	Initiator Terminator
Cohesive Hub	Factory

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

5294
5295
5296
5297
5298

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

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

Participant

Cohesive Coordination Hub

Initiator/Terminator

Cohesive Coordination Hub

Participant

5299

5300

5301

5302

5303

5304

5305

5306

5307

5308

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

5308 Part 3. Appendices

5309

5310

5311

5312

~~These terms seem to be all either not used, or effectively defined elsewhere~~The glossary is the subject of issue 4

5313 A. Glossary

5314

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 The address of an endpoint for a particular carrier protocol.

(CPA)

Business Transaction Protocol Address A compound address consisting of a mandatory *carrier protocol address* and an optional opaque suffix.

(BTPA)

PRF - suffix ? I've used "additional information"

Actor An entity which executes procedures, a software agent.

Application An actor which uses the Business Transaction Protocol.

Application Message A message produced by an application and consumed by an application.

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

Countereffect Contract	<p>The contract which governs the relationship between the effect and the countereffect of a procedure. In the absence of any other overriding contracts the countereffect contract is the promise that</p> <p>“The Countereffect will attempt so far as is possible to reverse or cancel the Effect such that an observer (on completion of the Countereffect) is unaware that the Effect ever occurred, but this attempt cannot be guaranteed to succeed”.</p>
Cancel	Process a countereffect for the current effect of a set of procedures.
Confirm	Ensure that the effect of a set of procedures is completed.
Prepare	Ensure that of a set of procedures is capable of being successfully instructed to cancel or to confirm.
Outcome	A decision to either cancel or confirm.
Participant	A set of procedures which is capable of receiving instructions from a coordinator to prepare, cancel and confirm. A participant must also have a BTPA to which these instructions will be delivered, in the form of BTP messages. A participant is identified by a participant identifier.
Inferior Identifier	An identifier assigned to an Inferior which is unique within the scope of an Address-as-Inferior.
Atomic Business Transaction	A set of participants (which may have only one member), all of which will receive instructions that will result in a homogeneous outcome.
<i>or</i>	(Transitively, a set of operations, whose effect is capable of countereffect.)
Atom	An atom is identified by an atom identifier.
Atom Identifier	A globally unique identifier assigned to an atom.
	<p><i>PRF – abs msgs define as unambiguous in scope of its address-as-superior, I think.</i></p>

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

5315