

1 Organization for the Advancement of Structured Information Systems

2 Business Transaction Protocol

3
4
5 An OASIS Committee Specification

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

7
8 Version 1.0 [0.9.1]

9 DD Mmm 2001 [17 January 2002 22:12]

10
11

<i>Working draft 0.1 (pre-London)</i>	14 June 2001
<i>Working draft 0.2 (London)</i>	18 June 2001
<i>Working draft 0.3a (circulated)</i>	12 July 2001
<i>Working draft 0.3c (circulated)</i>	20 July 2001
<i>Working draft 0.4 (circulated; incorporates PRF material)</i>	25 July 2001
<i>Working draft 0.6 (State tables)</i>	31 August 2001
<i>Working Draft 0.9</i>	24 October 2001
<i>Working Draft 0.9.0.1 – minor editorials issues applied</i>	16 November 2001
<i>Working Draft 0.9.0.2 – issue resolutions balloting to 10 Dec 2001</i>	4 December 2001
<i>Working Draft 0.9.0.3 – possible solution to msging issues</i>	11 December 2001
<i>Working Draft 0.9.0.4 – issue 79 solution, revise msging issues</i>	12 January 2002
<i>Working Draft 0.9.1 – includes all issues agreed 16 Jan 2002</i>	17 January 2002

12
13 ***Change marks relative to 0.9.0.4 with its changes all accepted***

14 Copyright and related notices

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

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

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

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

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

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

54

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
99

Acknowledgements

Employees of the following companies participated in the finalization of this specification as members of the OASIS Business Transactions Technical Committee:

- BEA Systems, Inc.
- Bowstreet, Inc.
- Choreology Ltd.
- Entrust, Inc.
- Hewlett-Packard Co.
- Interwoven Inc.
- IONA Technologies PLC
- SeeBeyond Inc.
- Sun Microsystems Computer Corp.
- Talking Blocks Inc.

The primary authors and editors of the main body of the specification were:

- Alex Ceponkus (alex@ceponkus.org)
- Peter Furniss (peter.furniss@choreology.com)
- Alastair Green (alastair.green@choreology.com)

Additional contributions to its writing were made by

- Sanjay Dalal (sanjay.dalal@bea.com)
- Mark Little (mark_little@hp.com)

We thank Pal Takacsi-Nagy of BEA Systems Inc for his efforts in chairing the Technical Committee, and Karl Best of OASIS for his guidance on the organization of the Committee's work.

In memory of Ed Felt

Ed Felt of BEA Systems Inc. was an active and highly valued contributor to the work of the OASIS Business Transactions Technical Committee.

His many years of design and implementation experience with the Tuxedo system, Weblogic's Java transactions, and Weblogic Integration's Conversation Management Protocol were brought to bear in his comments on and proposals for this specification.

He was killed in the crash of the hijacked United Airlines flight 93 near to Pittsburgh, on 11 September 2001.

99 **Typographical and Linguistic Conventions and Style**

100

101

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

102

103

104

105

Cancel

106

Participant

107

Application Message

108

109

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

110

Coordinator

111

112

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

113

114

115

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

116

117

BEGIN

118

CONTEXT

119

RESIGN

120

121

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

122

123

BEGIN/atom

124

125

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

126

127

BEGIN/atom & CONTEXT

128

129

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

130

131

ENROL + VOTE

132

133

XML schemata and instances are given in Courier:

134

135

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

136

137

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

138

139

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

140

141

142

143

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

144

145

146

147
148
149
150
151

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

151	Contents	
152		
153	Copyright and related notices.....	2
154	Acknowledgements	3
155	Typographical and Linguistic Conventions and Style	4
156	Contents	6
157	Part 1. Purpose and Features of BTP	10
158	Introduction.....	10
159	Development and Maintenance of the Specification.....	11
160	Overview of the Business Transaction Protocol	12
161	Part 2. Normative Specification of BTP	15
162	Actors, Roles and Relationships	15
163	Relationships.....	15
164	Roles involved in the outcome relationships	17
165	Superior.....	17
166	Inferior	18
167	Enroller	19
168	Participant	20
169	Sub-coordinator.....	20
170	Sub-composer	21
171	Roles involved in the control relationships.....	21
172	Decider.....	21
173	Coordinator	22
174	Composer	22
175	Terminator.....	22
176	Initiator.....	23
177	Factory	24
178	Other roles	24
179	Redirector.....	24
180	Status Requestor.....	25
181	Abstract Messages and Associated Contracts	25
182	Addresses.....	26
183	Request/response pairs.....	27
184	Compounding messages	27
185	Extensibility	29
186	Inferior handle	29
187	Messages.....	30
188	Qualifiers	30
189	Messages not restricted to outcome or control relationships.	31
190	CONTEXT.....	31
191	CONTEXT_REPLY	32
192	REQUEST_STATUS	33
193	STATUS	34
194	FAULT.....	36
195	REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES	39
196	Messages used in the outcome relationships	39

197	ENROL	39
198	ENROLLED	40
199	RESIGN	41
200	RESIGNED.....	42
201	PREPARE	43
202	PREPARED	44
203	CONFIRM	45
204	CONFIRMED	46
205	CANCEL	47
206	CANCELLED.....	48
207	CONFIRM_ONE_PHASE	49
208	HAZARD.....	50
209	CONTRADICTION.....	51
210	SUPERIOR_STATE.....	52
211	INFERIOR_STATE.....	53
212	REDIRECT	55
213	Messages used in control relationships.....	56
214	BEGIN	56
215	BEGUN.....	57
216	PREPARE_INFERIORS	58
217	CONFIRM_TRANSACTION	59
218	TRANSACTION_CONFIRMED.....	61
219	CANCEL_TRANSACTION	62
220	CANCEL_INFERIORS.....	63
221	TRANSACTION_CANCELLED.....	64
222	REQUEST_INFERIOR_STATUSES	65
223	INFERIOR_STATUSES	66
224	Groups – combinations of related messages.....	68
225	CONTEXT & application message.....	69
226	CONTEXT_REPLY & ENROL.....	69
227	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED.....	70
228	CONTEXT_REPLY & ENROL & application message (& PREPARED)	71
229	BEGUN & CONTEXT	72
230	BEGIN & CONTEXT.....	72
231	Standard qualifiers	72
232	Transaction timelimit.....	72
233	Inferior timeout	73
234	Minimum inferior timeout	74
235	Inferior name.....	74
236	State Tables	76
237	Explanation of the state tables	76
238	Status queries	76
239	Decision events	76
240	Disruptions – failure events	77
241	Invalid cells and assumptions of the communication mechanism	77
242	Meaning of state table events.....	78
243	Persistent information	82

244	Failure Recovery	95
245	Types of failure	95
246	Persistent information	96
247	Redirection.....	97
248	Terminator:Decider failures.....	98
249	XML representation of Message Set.....	98
250	Addresses	99
251	Qualifiers	99
252	Identifiers	100
253	Message References.....	100
254	Messages.....	100
255	CONTEXT.....	100
256	CONTEXT_REPLY	100
257	BEGIN	101
258	BEGUN.....	101
259	ENROL	101
260	ENROLLED	102
261	RESIGN	102
262	RESIGNED.....	102
263	PREPARE	103
264	PREPARED	103
265	CONFIRM	103
266	CONFIRMED	104
267	CANCEL	104
268	CANCELLED.....	104
269	CONFIRM_ONE_PHASE	105
270	HAZARD.....	105
271	CONTRADICTION.....	105
272	SUPERIOR_STATE.....	106
273	INFERIOR_STATE.....	106
274	REDIRECT.....	106
275	PREPARE_INFERIORS	107
276	CONFIRM_TRANSACTION	107
277	TRANSACTION_CONFIRMED.....	108
278	CANCEL_TRANSACTION	108
279	CANCEL_INFERIORS	108
280	TRANSACTION_CANCELLED.....	109
281	REQUEST_INFERIOR_STATUSES	109
282	INFERIOR_STATUSES	109
283	REQUEST_STATUS	110
284	STATUS	110
285	FAULT.....	111
286	Standard qualifiers	112
287	Transaction timelimit.....	112
288	Inferior timeout	112
289	Minimum inferior timeout	112
290	Compounding of Messages.....	112

291	Carrier Protocol Bindings	114
292	Carrier Protocol Binding Proforma.....	114
293	Bindings for request/response carrier protocols	115
294	Request/response exploitation rules.....	116
295	SOAP Binding	117
296	Example scenario using SOAP binding	119
297	SOAP + Attachments Binding.....	121
298	XML Schema.....	123
299	Conformance.....	135
300	Part 3. Appendices.....	137
301	A. Glossary.....	137
302		
303		

Part 1. Purpose and Features of BTP

Introduction

This document, which describes and defines the Business Transaction Protocol (BTP), is a Committee Specification of the Organization for the Advancement of Structured Information Standards (OASIS). The standard has been authored by the collective work of representatives of ten software product companies (listed on page 3), grouped in the Business Transactions Technical Committee (BT TC) of OASIS.

The OASIS BTP Technical Committee began its work at an inaugural meeting in San Jose, Calif. on 13 March 2001, and this specification was endorsed as a Committee Specification by a [*** unanimous] vote on [*** date].

BTP uses a two-phase outcome coordination protocol to create atomic effects (results of computations). BTP also permits the composition of such atomic units of work (atoms) into cohesive business transactions (cohesions), which allow application intervention into the selection of the atoms which will be confirmed, and of those which will be cancelled.

BTP is designed to allow transactional coordination of participants, which are part of services offered by multiple autonomous organizations (as well as within a single organization). It is therefore ideally suited for use in a Web Services environment. For this reason this specification defines communications protocol bindings which target the emerging Web Services arena, while preserving the capacity to carry BTP messages over other communication protocols. Protocol message structure and content constraints are schematized in XML, and message content is encoded in XML instances.

The BTP allows great flexibility in the implementation of business transaction participants. Such participants enable the consistent reversal of the effects of atoms. BTP participants may use recorded before- or after-images, or compensation operations to provide the “roll-forward, roll-back” capacity which enables their subordination to the overall outcome of an atomic business transaction.

The BTP is an interoperation protocol which defines the roles which software agents (actors) may occupy, the messages that pass between such actors, and the obligations upon and commitments made by actors-in-roles. It does not define the programming interfaces to be used by application programmers to stimulate message flow or associated state changes.

The BTP is based on a permissive and minimal approach, where constraints on implementation choices are avoided. The protocol also tries to avoid unnecessary dependencies on other standards, with the aim of lowering the hurdle to implementation.

346
347
348
349
350
351
352
353
354
355
356
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

Development and Maintenance of the Specification

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

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

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

- ❑ 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

384 Overview of the Business Transaction Protocol

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

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

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

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

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

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

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

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

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

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

433

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

445

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

454

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

466

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

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

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

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);

534

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

537

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

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

560

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

566

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

570

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

574

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

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

589 **Roles involved in the outcome relationships**

590 **Superior**

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

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

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

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

614 A Superior receives

615
616 ENROL

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

620 A Superior sends
621

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

625
626 A Superior sends

627
628 PREPARE
629 CONFIRM
630 CANCEL
631 RESIGNED
632 CONFIRM_ONE_PHASE
633 SUPERIOR_STATE

634
635 to an enrolled Inferior.

636
637 A Superior receives

638
639 PREPARED
640 CANCELLED
641 CONFIRMED
642 HAZARD
643 RESIGN
644 INFERIOR_STATE

645
646 from an enrolled Inferior.

647
648 **Inferior**

649
650 Responsible for applying the Outcome to some set of associated operations – the application
651 determines which operations are the responsibility of a particular Inferior.

652
653 An Inferior is **Enrolled** with a single Superior (hereafter referred to as “its Superior”),
654 establishing a Superior:Inferior relationship. If the Inferior is able to ensure that either a
655 confirm or cancel decision can be applied to the associated operations, and can persist
656 information to retain that condition, it sends a PREPARED message to the Superior. When
657 the Outcome is received from the Superior, the Inferior applies it, deletes the persistent
658 information, and replies with CANCELLED or CONFIRMED as appropriate.

659
660 If an Inferior is unable to come to a prepared state, it cancels the associated operations and
661 informs the Superior with a CANCELLED message. If it is unable to either come to a
662 prepared state, or to cancel the associated operations, it informs the Superior with a
663 HAZARD message.

664
665 An Inferior that has become prepared may, exceptionally, make an autonomous decision to be
666 applied to the associated operations, without waiting for the Outcome from the Superior. It is
667 required to persist this autonomous decision and report it to the Superior with CONFIRMED
668 or CANCELLED as appropriate. If, when CONFIRM or CANCEL is received, the

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

672

673 An Inferior receives

674

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

681

682 from its Superior.

683

684 An Inferior sends

685

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

692

693 to its Superior.

694

695

696 **Enroller**

697

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

703

704 An Enroller sends

705

706 ENROL

707

708 to a Superior.

709

710 An Enroller receives

711

712 ENROLLED

713

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

715

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

721

722 Participant

723

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

729

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

733

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

738

Note – Possible approaches are:

739

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

743

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

746

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

749

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

751

752 Sub-coordinator

753

754 An Inferior which is also an Atomic Superior.

755

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

758
759 From the perspective of its Superior (the one the sub-coordinator is Inferior to), there is no
760 difference between a sub-coordinator and any other Inferior. From this perspective, the
761 “associated operations” of the sub-coordinator as an Inferior include the relationships with its
762 Inferiors.

763
764 A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and
765 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is
766 propagated to all Inferiors.

767 **Sub-composer**

768
769 An Inferior which is also a Cohesive Superior.

770
771 Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from
772 the perspective of its Superior.

773
774 A sub-composer is similar to a sub-coordinator, except that the constraints linking the
775 different Inferiors concern only those Inferiors in the confirm-set. How the confirm-set is
776 controlled, and when, is not defined in this specification.

777
778 If the sub-composer is instructed to cancel, by receiving a CANCEL message from its
779 Superior, the cancellation is propagated to all its Inferiors.

780
781
782

783 **Roles involved in the control relationships**

784

785 **Decider**

786

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

793

794 A Decider is instructed to cancel by receiving CANCEL_TRANSACTION.

795

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

799

800 All Deciders receive

801 CONFIRM_TRANSACTION

802 CANCEL_TRANSACTION

803 REQUEST_INFERIOR_STATUSES

804

805 All Deciders send
806 CONFIRM_COMPLETE
807 CANCEL_COMPLETE
808 INFERIOR_STATUSES

809
810

811 **Coordinator**

812

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

815

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

817

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

822

823 **Composer**

824

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

827

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

830

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

835

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

840

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

843

844

845 **Terminator**

846

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

849

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

852
853 A request to confirm is made by sending CONFIRM_TRANSACTION to the target Decider.
854 If the Decider is a Cohesion Composer, the Terminator may select which of the Composer's
855 Inferiors are to be included in the confirm-set. If the Decider is an Atom Coordinator, all
856 Inferiors are included. After applying the decision, the Decider replies with
857 CONFIRM_COMPLETE, CANCEL_COMPLETE or (in the case of problems)
858 INFERIOR_STATUSES.

859
860 A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its
861 Inferiors with PREPARE_INFERIORS. The Composer replies with
862 INFERIOR_STATUSES.

863
864 A Terminator may send CANCEL_TRANSACTION to instruct the Decider to cancel the
865 whole business transaction.,. The Decider replies with CANCEL_COMPLETE if all Inferiors
866 cancel successfully, and with INFERIOR_STATUSES in the case of problems.. If the
867 Decider is a Cohesion Composer, the Terminator may send CANCEL_INFERIORS to cancel
868 some of the Inferiors; the Decider always replies with INFERIOR_STATUSES.

869
870 A Terminator may check the status of the Inferiors of the Decider by sending
871 REQUEST_INFERIOR_STATUSES. The Decider replies with INFERIOR_STATUSES.

872
873 A Terminator sends
874 CONFIRM_TRANSACTION
875 CANCEL_TRANSACTION
876 CANCEL_INFERIORS
877 PREPARE_INFERIORS
878 REQUEST_INFERIOR_STATUSES

879
880 A Terminator receives
881 CONFIRM_COMPLETE
882 CANCEL_COMPLETE
883 INFERIOR_STATUSES

884 885 Initiator

886
887 Requests a **Factory** to create a Superior – this will either be a Decider (representing a new
888 top-level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an
889 existing business transaction.

890
891 An Initiator sends
892
893 BEGIN
894 BEGIN & CONTEXT
895
896 to a Factory, and receives in reply
897
898 BEGUN & CONTEXT

899
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

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.)

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

947

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

950

951 **Status Requestor**

952

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

958

959 A Status Requestor sends

960

961 REQUEST_STATUS

962 REQUEST_INFERIOR_STATUSES

963

964 and receives

965

966 STATUS

967 INFERIOR_STATUSES

968

969 in response.

970

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

974

975 **Abstract Messages and Associated Contracts**

976

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

980

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

982

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

985

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

988

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

990

991 and

992

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

994

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

997

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

1004

1005

1006 **Addresses**

1007

1008 All of the messages except CONTEXT and CONTEXT_REPLY have a “target address”
1009 parameter and many also have other address parameters. These latter identify the desired
1010 target of other messages in the set. In all cases, the exact value will invariably have been
1011 originally determined by the implementation that is the target or desired future target.

1012

1013 The detailed format of the address will depend on the particular carrier protocol, but at this
1014 abstract level is considered to have three parts. The first part, the “binding name”, identifies
1015 the binding to a particular carrier protocol – some bindings are specified in this document,
1016 others can be specified elsewhere. The second part of the address, the “binding address”, is
1017 meaningful to the carrier protocol itself, which will use it for the communication (i.e. it will
1018 permit a message to be delivered to a receiver). The third part, “additional information”, is
1019 not used or understood by the carrier protocol. The “additional information” may be a
1020 structured value.

1021

1022 When a message is actually transmitted, the “binding name” of the target address will identify
1023 which carrier protocol is in use and the “binding address” will identify the destination, as
1024 known to the carrier protocol. The entire binding address is considered to be “consumed” by
1025 the carrier protocol implementation. All of it may be used by the sending implementation, or
1026 some of it may be transmitted in headers, or as part of a URL in the carrier protocol, but then
1027 used or consumed by the receiving implementation of the carrier protocol to direct the BTP
1028 message to a BTP-aware entity (BTP-aware in that it is capable of interpreting the BTP
1029 messages). The “additional information” of the target address will be part of the BTP
1030 message itself and used in some way by the receiving BTP-aware entity (it could be used to
1031 route the message on to some other BTP entity). Thus, for the target address, only the
1032 “additional information” field is transmitted in the BTP message and the “additional
1033 information” is opaque to parties other than the recipient.

1034

1035 For other addresses in BTP messages, all three components will be within the message.

1036

1037 All messages that concern a particular Superior:Inferior relationship have an identifier
1038 parameter for the target side as well as the compound target address. This allows full
1039 flexibility for implementation choices – an implementation can:

1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086

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

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

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

Request/response pairs

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

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

Compounding messages

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

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

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

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

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

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

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

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

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

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

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

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

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

1158

1159 **Extensibility**

1160

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

1170

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

1173

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

1175

1176 **Inferior handle**

1177

1178 Some of the messages exchanged between a Terminator and a Decider are concerned with the
1179 individual Inferiors enrolled with the Decider, and not with the business transaction as a

1180 whole. These messages distinguish the Inferiors of Decider using an “inferior handle”. This is
1181 created by the Decider and is unambiguous within the scope of the Decider .

1182
1183 The “inferior handle” is distinct from the “inferior identifier” passed on an ENROL message
1184 (among other places). The latter is created by the Inferior (or its enroller) and is required to be
1185 unambiguous within the scope of the address-as-inferior on the ENROL (and unambiguous
1186 within **any** of the individual addresses in that set of BTP addresses - the identifier must
1187 identify the Inferior across all the places it might migrate to or that have recovery
1188 responsibility for it).

1189
1190 The “inferior handle” is only used by the Terminator to refer to the inferiors of the Decider.
1191 In messages between the Decider and its Inferiors, the address-as-inferior and inferior
1192 identifier are used.

1193

1194 Messages

1195

1196 Qualifiers

1197

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

1200

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

1201

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

1208

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

1211

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

1216

1217 **To-be-propagated** if this has the value “true” and the receiving entity passes the
 1218 BTP message (which may be a CONTEXT, but can be other messages) onwards
 1219 to other entities, the same Qualifier value shall be included. If the value is
 1220 “false”, the Qualifier shall not be automatically included if the BTP message is
 1221 passed onwards. (If the receiving entity does support the qualifier type, it is
 1222 possible a propagated message may contain another instance of the same type,
 1223 even with the same Content – this is not considered propagation of the original
 1224 qualifier.). Default is “false”.

1225
 1226 **Content** the type (which may be structured) and meaning of the content is
 1227 defined by the specification of the Qualifier.
 1228
 1229

1230 **Messages not restricted to outcome or control relationships.**

1231
 1232 The messages in this section are used between various roles. CONTEXT message is used in
 1233 the Initiator:Factory relationship (when it is related to BEGIN or to BEGUN), and related to
 1234 an application ‘message’ to propagate the business transaction between parts of the
 1235 application. CONTEXT_REPLY is used as the reply to a CONTEXT.REQUEST_STATUS
 1236 can be issued to, and STATUS returned by any of Decider, Superior or Inferior. FAULT can
 1237 be used on any relationship to indicate an error condition back to the sender of a message.
 1238

1239 **CONTEXT**

1240
 1241 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more
 1242 application messages. (The means by which this relationship is represented is determined by
 1243 the binding and the binding mechanisms of the application protocol.) The “superior type”
 1244 parameter identifies whether the Superior will apply the same decision to all Inferiors
 1245 enrolled using the same superior identifier (“superior type” is “atom”) or whether it may
 1246 apply different decisions (“superior type” is “cohesion”).
 1247

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

1248
 1249
 1250 **address-as-superior** the address to which ENROL and other messages from an
 1251 enrolled Inferior are to be sent. This can be a set of alternative addresses.
 1252

1253 **superior identifier** identifies the Superior within the scope of the address-as-
 1254 superior

1255
1256 **reply-address** the address to which a replying CONTEXT_REPLY is to be sent.
1257 This may be different each time the CONTEXT is transmitted – it refers to the
1258 destination of a replying CONTEXT_REPLY for this particular transmission of
1259 the CONTEXT.

1260
1261 **superior type** identifies whether the CONTEXT refers to a Cohesion or an
1262 Atom. Default is atom.

1263
1264
1265 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
1266 timelimit” is carried by CONTEXT.

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

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

1273
1274

CONTEXT_REPLY

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

1284

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

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

1288
1289 **superior-address** one of the addresses from the address-as-superior from the
1290 CONTEXT. (The parameter is present in CONTEXT_REPLY to disambiguate
1291 the superior identifier.)

1292

1293 **superior identifier** the superior identifier from the CONTEXT

1294

1295 **completion_status:** reports whether all enrol operations made necessary by the
1296 receipt of the earlier CONTEXT message have completed. Values are

1297

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.

1298

1299 **qualifiers** standardised or other qualifiers.

1300

1301 The form CONTEXT_REPLY/completed, CONTEXT_REPLY/related and
1302 CONTEXT_REPLY/repudiated refer to CONTEXT_REPLY messages with status having the
1303 appropriate value. The form CONTEXT_REPLY/ok refers to either of
1304 CONTEXT_REPLY/completed or CONTEXT_REPLY/related.

1305

1306 If there are no necessary enrolments (e.g. the application messages related to the received
1307 CONTEXT did not require the enrolment of any Inferiors), then
1308 CONTEXT_REPLY/completed is used.

1309

1310 If a CONTEXT_REPLY/repudiated is received, the receiving implementation **must** ensure
1311 that the business transaction will not be confirmed.

1312

1313

1314 REQUEST_STATUS

1315

1316 Sent to an Inferior, Superior or to a Decider to ask it to reply with STATUS. The receiver
1317 may reject the request with a FAULT(StatusRefused).

1318

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

1319

1320 **target address** the address to which the REQUEST_STATUS message is sent.
1321 This can be any of address-as-decider, address-as-inferior or address-as-superior.

1322
 1323 **reply address** the address to which the replying STATUS should be sent.
 1324
 1325 **target identifier** The identifier for the business transaction, or part of business
 1326 transaction whose status is sought. If the target-address is an address-as-decider,
 1327 this parameter shall be the “transaction-identifier” on the BEGUN message. If the
 1328 target-address is an address-as-inferior, this parameter shall be the “inferior-
 1329 identifier” on the ENROL message. If the target-address is a an address-as-
 1330 superior, this parameter shall be the “superior-identifier” on the CONTEXT.
 1331 **qualifiers** standardised or other qualifiers.

1332
 1333 Types of FAULT possible (sent to reply address)

1334
 1335 **General**

1336 **StatusRefused** – if the receiver is not prepared to report its status to the
 1337 sender of this message

1338 **UnknownTransaction** – if the target-identifier is unknown

1339
 1340 **STATUS**

1341
 1342
 1343 Sent by a Inferior, Superior or Decider in reply to a REQUEST_STATUS, reporting the
 1344 overall state of the transaction tree node represented by the sender.

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

1346
 1347 **target address** the address to which the STATUS is sent. This will be the reply
 1348 address on the REQUEST_STATUS message
 1349
 1350 **responders-address** the address of the sender of the STATUS message – one of
 1351 address-as-inferior, address-as-decider, address-as-superior(with the responders-
 1352 identifier, this determines who the message is from).. If the sender has different
 1353 addresses as multiple roles (as Decider, Inferior or Superior), this shall be the
 1354 address on which the REQUEST_STATUS was received.
 1355
 1356 **responders-identifier** the identifier of the state, aligned with the responders-
 1357 address. If the sender has multiple roles in the transaction (as Decider, Inferior or
 1358 Superior), this shall be the target-identifier on the REQUEST_STATUS

1359
 1360
 1361
 1362
 1363
 1364

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

1365

1366

qualifiers standardised or other qualifiers.

1367

1368 Types of FAULT possible

1369

General

1370

1371

1372 **FAULT**

1373

Sent in reply to various messages to report an error condition

1374

1375

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

1376

1377

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

1378

1379

1380

1381

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).

1382

1383

1384

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

1385

1386

1387
1388
1389
1390
1391
1392

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:

1393

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.	

1394

1395 *UnknownParameter* A BTP message has been Free text explanation
 1396 received with an unrecognised
 1397 **q** parameter
 1398 **u**
 1399 **Qualifiers** standardised or other qualifiers.
 1400

1401 Note – If the carrier mechanism used for the transmission of BTP messages
 1402 is capable of delivering messages in a different order than they were sent in,
 1403 the “WrongState” FAULT is not sent and should be ignored if received.

1404
 1405 **REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES**

1406
 1407 REQUEST_INFERIOR_STATUSES may be sent to and INFERIOR_STATUSES sent from
 1408 any Decider, Superior or Inferior, asking it to report on the status of its relationships with
 1409 Inferiors (if any). Since Deciders are required to respond to
 1410 REQUEST_INFERIOR_STATUSES with INFERIOR_STATUSES but non-Deciders may
 1411 just issue FAULT(StatusRefused), and INFERIOR_STATUSES is also used as a reply to
 1412 other messages from Terminator to Decider, these messages are described below under the
 1413 messages used in the control relationships.
 1414

1415 **Messages used in the outcome relationships**

1416
 1417 **ENROL**

1418
 1419 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a
 1420 CONTEXT message in relation to an application request.
 1421 The actor issuing ENROL plays the role of Enroller.
 1422

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

1423
 1424 **target address** the address to which the ENROL is sent. This will be the
 1425 address-as-superior from the CONTEXT message.
 1426

1427 **superior identifier.** The superior identifier as on the CONTEXT message
 1428
 1429 **reply requested** true if an ENROLLED response is required, false otherwise.
 1430 Default is false.
 1431
 1432 **reply address** the address to which a replying ENROLLED is to be sent, if
 1433 “reply requested” is true. If this field is absent and “reply requested” is true, the
 1434 ENROLLED should be sent to the “address-as-inferior” (or one of them, at
 1435 sender’s option)
 1436
 1437 **address-as-inferior** the address to which PREPARE, CONFIRM, CANCEL and
 1438 SUPERIOR_STATE messages for this Inferior are to be sent.
 1439
 1440 **inferior identifier** an identifier that unambiguously identifies this Inferior within
 1441 the scope of any of the address-as-inferior set of BTP-addresses.
 1442
 1443 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
 1444 name” may be present.
 1445

1446 Types of FAULT possible (sent to Reply address)

1447
 1448 **General**

1449 **InvalidSuperior** – if superior identifier is unknown

1450 **DuplicateInferior** – if inferior with at least one of the set address-as-
 1451 inferior the same and the same inferior identifier is already enrolled

1452 **WrongState** – if it is too late to enrol new Inferiors (generally if the
 1453 Superior has already sent a PREPARED message to its superior or
 1454 terminator, or if it has already issued CONFIRM to other Inferiors).
 1455

1456 The form ENROL/rsp-req refers to an ENROL message with “reply requested” having the
 1457 value “true”; ENROL/no-rsp-req refers to an ENROL message with “reply requested” having
 1458 the value “false”
 1459

1460 ENROL/no-rsp-req is typically sent in relation to CONTEXT_REPLY/related. ENROL/rsp-
 1461 req is typically when CONTEXT_REPLY/completed will be used (after the ENROLLED
 1462 message has been received.)
 1463

1464 **ENROLLED**

1465
 1466 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been
 1467 successfully enrolled (and will therefore be included in the termination exchanges)
 1468

Parameter	Type
target address	BTP address
inferior identifier	Identifier

Parameter	Type
inferior-handle	Handle
Qualifiers	List of qualifiers

1469

1470

1471

1472

1473

1474

1475

1476

1477

1478

1479

1480

1481

1482

1483

1484

1485

1486

RESIGN

1487

1488

1489

1490

1491

1492

1493

1494

1495

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

inferior identifier The inferior identifier as on the ENROL message

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

qualifiers standardised or other qualifiers.

No FAULT messages are issued on receiving ENROLLED.

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

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

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

1496

1497

1498

1499

1500

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

superior-identifier The superior identifier as on the ENROL message

1501
 1502 **address-as-inferior** The address-as-inferior as on the earlier ENROL message
 1503 (with the inferior identifier, this determines who the message is from)
 1504
 1505 **inferior-identifier** The inferior identifier as on the earlier ENROL message
 1506
 1507 **response-requested** is set to “true” if a RESIGNED response is required.
 1508
 1509 **qualifiers** standardised or other qualifiers.

1510
 1511 Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued
 1512 early.
 1513

1514 Types of FAULT possible (sent to address-as-inferior)

- 1515
 1516 *General*
 1517 *InvalidSuperior* – if superior identifier is unknown
 1518 *InvalidInferior* – if no ENROL had been received for this address-as-
 1519 inferior and identifier (Inferior Identity)
 1520 *WrongState* – if a PREPARED or CANCELLED has already been
 1521 received by the Superior from this Inferior
 1522

1523 The form RESIGN/rsp-req refers to an RESIGN message with “reply requested” having the
 1524 value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “reply requested”
 1525 having the value “false”
 1526

1527
 1528 **RESIGNED**

1529
 1530 Sent in reply to a RESIGN/rsp-req message.
 1531

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

1532
 1533 **target address** the address to which the RESIGNED is sent. This will be the
 1534 address-as-inferior from the ENROL message.
 1535

1536 **inferior identifier** The inferior identifier as on the earlier ENROL message for
 1537 this Inferior.
 1538

1539 **qualifiers** standardised or other qualifiers.
 1540

1541 After receiving this message the Inferior will not receive any more messages with this
1542 address-as-inferior and identifier.

1543
1544 No FAULT messages are issued on receiving RESIGNED.
1545

1546 PREPARE

1547
1548 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor
1549 RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after
1550 receiving a PREPARED message.
1551

1552

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

1553
1554 **target address** the address to which the PREPARE message is sent. When sent
1555 from Superior to Inferior, this will be the address-as-inferior from the ENROL
1556 message.
1557

1558 **inferior identifier** When sent from Superior to Inferior, the inferior identifier as
1559 on the earlier ENROL message.
1560

1561
1562 **qualifiers** standardised or other qualifiers. The standard qualifier “Minimal
1563 inferior timeout” is carried by PREPARE.
1564

1565
1566 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or
1567 RESIGN.
1568

1569 Types of FAULT possible (sent to Superior address)
1570

1571 *General*

1572 **InvalidInferior** – if inferior identifier is unknown, or an inferior-handle
1573 on the inferiors-list is unknown

1574 **WrongState** – if a CONFIRM or CANCEL has already been received by
1575 this Inferior.
1576

1577

1578 **PREPARED**

1579
1580
1581
1582
1583
1584
1585
1586

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

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

1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613

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

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

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

inferior identifier The inferior identifier as on the ENROL message

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

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

On sending a PREPARED, the Inferior undertakes to maintain its ability to confirm or cancel the effects of the associated operations until it receives a CONFIRM or CANCEL message.

1614 Qualifiers may define a time limit or other constraints on this promise. The “default is
1615 cancel” parameter affects only the subsequent message exchanges and does not of itself state
1616 that cancellation will occur.

1617
1618 Types of FAULT possible (sent to address-as-inferior)

1619
1620 *General*

1621 *InvalidSuperior* – if Superior identifier is unknown

1622 *InvalidInferior* – if no ENROL has been received for this address-as-
1623 inferior and identifier, or if RESIGN has been received from this Inferior

1624

1625 The form PREPARED/cancel refers to a PREPARED message with “default is cancel” =
1626 “true”. The unqualified form PREPARED refers to a PREPARED message with “default is
1627 cancel” = “false”.

1628

1629

1630 **CONFIRM**

1631

1632 Sent by the Superior to an Inferior from whom PREPARED has been received.

1633

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

1634

1635 **target address** the address to which the CONFIRM message is sent. This will
1636 be the address-as-inferior from the ENROL message.

1637

1638 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1639 this Inferior.

1640

1641 **qualifiers** standardised or other qualifiers.

1642

1643 On receiving CONFIRM, the Inferior is released from its promise to be able to undo the
1644 operations of associated with the Inferior. The effects of the operations can be made available
1645 to everyone (if they weren’t already).

1646

1647 Types of FAULT possible (sent to Superior address)

1648

1649 *General*

1650 *InvalidInferior* – if inferior identifier is unknown

1651 *WrongState* – if no PREPARED has been sent by, or if CANCEL has
1652 been received by this Inferior.

1653

1654

1655 **CONFIRMED**

1656
1657
1658
1659
1660
1661

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

1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688

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

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.

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

qualifiers standardised or other qualifiers.

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

1689 *General*
 1690 *InvalidSuperior* – if Superior identifier is unknown
 1691 *InvalidInferior* – if no ENROL has been received for this address-as-
 1692 inferior and identifier, or if RESIGN has been received from this Inferior.
 1693

1694 Note – A CONFIRMED message arriving before a CONFIRM message is
 1695 sent, or after a CANCEL has been sent will occur when the Inferior has
 1696 taken an autonomous decision and is not regarded as occurring in the wrong
 1697 state. (The latter will cause a CONTRADICTION message to be sent.)

1698
 1699 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm
 1700 received” = “false”; CONFIRMED/response refers to a CONFIRMED message
 1701 with “confirm received” = “true”.
 1702
 1703

1704 **CANCEL**

1705
 1706 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.
 1707

Parameter	Type
target address	BTP address
inferior identifier	Identifier

qualifiers List of qualifiers

1708
 1709 **target address** the address to which the CANCEL message is sent. When sent
 1710 from Superior to Inferior, this will be the address-as-inferior from the ENROL
 1711 message.
 1712

1713 **inferior identifier** When sent from Superior to Inferior, the inferior identifier as
 1714 on the earlier ENROL message.
 1715

1716 **qualifiers** standardised or other qualifiers.
 1717

1718 When sent to an Inferior, the effects of any operations associated with the Inferior should be
 1719 undone. If the Inferior had sent PREPARED, the Inferior is released from its promise to be
 1720 able to confirm the operations.
 1721

1722 Types of FAULT possible (sent to Superior address)
 1723

1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749

General

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

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

CANCELLED

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

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

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

Parameter

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

1750
1751
1752
1753
1754
1755
1756
1757

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.

1758 **address-as-inferior** When the message is sent from an Inferior to the Superior,
 1759 this shall be the address-as-inferior as on the earlier ENROL message (with the
 1760 inferior identifier, this determines who the message is from).
 1761
 1762 **inferior identifier** When the message is sent from an Inferior to the Superior, this
 1763 shall be the inferior identifier as on the earlier ENROL message.
 1764
 1765 **qualifiers** standardised or other qualifiers.
 1766

1767 Types of FAULT possible (sent to address-as-inferior)

1768 *General*

1769 *InvalidSuperior* – if Superior identifier is unknown

1770 *InvalidInferior* – if no ENROL has been received for this address-as-
 1771 inferior and identifier, or if RESIGN has been received from this Inferior

1772 *WrongState* – if CONFIRM has been sent
 1773
 1774

1775 Note – A CANCELLED message arriving before a CANCEL message is
 1776 sent, or after a CONFIRM has been sent will occur when the Inferior has
 1777 taken an autonomous decision and is not regarded as occurring in the wrong
 1778 state. (The latter will cause a CONTRADICTION message to be sent.)

1779
 1780
 1781 **CONFIRM_ONE_PHASE**
 1782

1783 Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In
 1784 this case the two-phase exchange is not performed between the Superior and Inferior and the
 1785 outcome decision for the operations associated with the Inferior is determined by the Inferior.
 1786

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

1787
 1788 **target address** the address to which the CONFIRM_ONE_PHASE message is
 1789 sent This will be the address-as-inferior on the ENROL message.
 1790

1791 **inferior identifier** The inferior identifier as on the earlier ENROL message for
 1792 this Inferior.
 1793

1794 **report hazard** Defines whether the superior wishes to be informed if a mixed
1795 condition occurs for the operations associated with the Inferior. If “report hazard”
1796 is “true”, the Inferior will reply with HAZARD if a mixed condition occurs, or if
1797 the Inferior cannot determine that a mixed condition has not occurred. If “report
1798 hazard” is false, the Inferior will report only its own decision, regardless of
1799 whether that decision was correctly and consistently applied. Default is false.

1800
1801 **qualifiers** standardised or other qualifiers.

1802
1803 CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom
1804 PREPARED has been received (subject to the requirement that there is only one enrolled
1805 Inferior).

1806
1807 Types of FAULT possible (sent to Superior address)

1808
1809 *General*
1810 *InvalidInferior* – if inferior identifier is unknown
1811 *WrongState* – if a PREPARE has already been received from this
1812 Inferior

1813
1814 **HAZARD**

1815
1816 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly
1817 and consistently cancel or confirm the operations in accord with the decision (either the
1818 received decision of the superior or its own autonomous decision), or when the Inferior is
1819 unable to determine that a “mixed” condition has not occurred.

1820
1821 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there
1822 is a mixed condition within its associated operations or is unable to determine that there is not
1823 a mixed condition.

1824

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

1825
1826 **target address** the address to which the HAZARD is sent. This will be the
1827 superior address from the ENROL message.

1828
1829 **superior identifier** The superior identifier as used on the ENROL message

1830

1831 **address-as-inferior** The address-as-inferior as on the earlier ENROL message
 1832 (with the inferior identifier, this determines who the message is from)
 1833
 1834 **inferior identifier** The inferior identifier as on the earlier ENROL message
 1835
 1836 **level** indicates, with value “mixed” that a mixed condition has definitely
 1837 occurred; or, with value “possible” that it is unable to determine whether a mixed
 1838 condition has occurred or not.
 1839
 1840 **qualifiers** standardised or other qualifiers.

1841
 1842 Types of FAULT possible (sent to address-as-inferior)
 1843

1844 *General*

1845 *InvalidSuperior* – if Superior identifier is unknown

1846 *InvalidInferior* – if no ENROL has been received for this address-as-
 1847 inferior and identifier, or if RESIGN has been received from this Inferior
 1848

1849
 1850 The form HAZARD/mixed refers to a HAZARD message with “level” = “mixed”, the form
 1851 HAZARD/possible refers to a HAZARD message with “level” = “possible”.
 1852

1853 **CONTRADICTION**

1854
 1855 Sent by the Superior to an Inferior that has taken an autonomous decision contrary to the
 1856 decision for the atom. This is detected by the Superior when the ‘wrong’ one of
 1857 CONFIRMED or CANCELLED is received. CONTRADICTION is also sent in response to a
 1858 HAZARD message.
 1859

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

1860
 1861 **target address** the address to which the CONTRADICTION message is sent.
 1862 This will be the address-as-inferior from the ENROL message.
 1863

1864 **inferior identifier** The inferior identifier as on the earlier ENROL message for
 1865 this Inferior.
 1866

1867 **qualifiers** standardised or other qualifiers.
 1868

1869 Types of FAULT possible (sent to Superior address)
 1870

1871 *General*

1872 *InvalidInferior* – if inferior identifier is unknown
 1873 *WrongState* – if neither CONFIRMED or CANCELLED has been sent
 1874 by this Inferior
 1875

1876 **SUPERIOR_STATE**

1877
 1878 Sent by a Superior as a query to an Inferior when

- 1879
 1880 1. in the active state
 1881
 1882 2. there is uncertainty what state the Inferior has reached (due to recovery from
 1883 previous failure or other reason).
 1884

1885 Also sent by the Superior to the Inferior in response to a received INFERIOR_STATE, in
 1886 particular states.
 1887

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

1888
 1889 **target address** the address to which the SUPERIOR_STATE message is sent.
 1890 This will be the address-as-inferior from the ENROL message.
 1891

1892 **inferior identifier** The inferior identifier as on the earlier ENROL message for
 1893 this Inferior.
 1894

1895 **status** states the current state of the Superior, in terms of its relation to this
 1896 Inferior only.
 1897

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

unknown

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

1898
1899
1900
1901
1902
1903
1904
1905
1906
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

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

qualifiers standardised or other qualifiers.

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

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

1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950

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

superior identifier The superior identifier as used on the ENROL message

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

inferior identifier The inferior identifier as on the ENROL message

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

status value	meaning/previous message sent
<i>active</i>	The relationship with the Superior is in the active state from the perspective of the Inferior; ENROL has been sent, a decision to send PREPARED has not been made.
<i>inaccessible</i>	The state information for the relationship with the Superior, if it exists, cannot be accessed at the moment. This should be a transient condition
<i>unknown</i>	The Inferior is not known – it does not exist from the perspective of the Superior. The Inferior can be treated as cancelled

1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964

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

qualifiers standardised or other qualifiers.

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

A status of “unknown” shall only be sent if it has been determined for certain that the Inferior has no knowledge of a relationship with the Superior. If there could be persistent information

1965 corresponding to the Superior, but it is not accessible from the entity receiving an
1966 SUPERIOR_STATE/*/*y (or other) message targetted on the Inferior or the entity cannot
1967 determine whether any such persistent information exists, the response shall be
1968 “inaccessible”.

1969
1970 INFERIOR_STATE/unknown is also used as a response to messages, other than
1971 SUPERIOR_STATE/*/*y that are received when the Inferior is not known (and it is known
1972 there is no state information for it).

1973
1974 A SUPERIOR_STATE/INFERIOR_STATE exchange that determines that one or both sides
1975 are in the active state does not require that the Inferior be cancelled (unlike some other two-
1976 phase commit protocols). The relationship between Superior and Inferior, and related
1977 application elements may be continued, with new application messages carrying the same
1978 CONTEXT. Similarly, if the Inferior is prepared but the Superior is active, there is no
1979 required impact on the progression of the relationship between them.

1980
1981 The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a
1982 value equivalent to “abcd” (for active, unknown and inaccessible) and with “reply requested”
1983 = “false”. INFERIOR_STATE/abcd/y refers to a similar message, but with “reply requested”
1984 = “true”. The form INFERIOR_STATE/*/*y refers to a INFERIOR_STATE message with
1985 “reply requested” = “true” and any value for status.

1986
1987
1988
1989

1990 REDIRECT

1991
1992 Sent when the address previously given for a Superior or Inferior is no longer valid and the
1993 relevant state information is now accessible with a different address (but the same superior or
1994 inferior identifier).

1995

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

1996
1997 **target address** the address to which the REDIRECT is sent. This may be the
1998 reply address from a received message or the address of the opposite side
1999 (superior/inferior) as given in a CONTEXT or ENROL message
2000

2001 **superior identifier** The superior identifier as on the CONTEXT message and
 2002 used on an ENROL message. (present only if the REDIRECT is sent from the
 2003 Inferior).
 2004
 2005 **inferior identifier** The inferior identifier as on the ENROL message
 2006
 2007 **old address** The previous address of the sender of REDIRECT. A match is
 2008 considered to apply if any of the old addresses match one that is already known.
 2009
 2010 **new address** The (set of alternatives) new addresses to be used for messages
 2011 sent to this entity.
 2012
 2013 **qualifiers** standardised or other qualifiers.
 2014
 2015 If the actor whose address is changed is an Inferior, the new address value
 2016 replaces the address-as-inferior as present in the ENROL.
 2017
 2018 If the actor whose address is changed is a Superior, the new address value
 2019 replaces the Superior address as present in the CONTEXT message (or as present
 2020 in any other mechanism used to establish the Superior:Inferior relationship).
 2021
 2022
 2023

2024 **Messages used in control relationships**

2025 **BEGIN**

2026 A request to a Factory to create a new Business Transaction. This may either be a new top-
 2027 level transaction, in which case the Composer or Coordinator will be the Decider, or the new
 2028 Business Transaction may be immediately made the Inferior within an existing Business
 2029 Transaction (thus creating a sub-Composer or sub-Coordinator).
 2030
 2031
 2032

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

2033
 2034 **target address** the address of the entity to which the BEGIN is sent. How this
 2035 address is acquired and the nature of the entity are outside the scope of this
 2036 specification.
 2037
 2038 **reply address** the address to which the replying BEGUN and related
 2039 CONTEXT message should be sent.

2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054

2055
2056
2057

2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070

2071
2072
2073
2074

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

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

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

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

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

Types of FAULT possible (sent to Reply address)

General

BEGUN

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

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

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

2075 **address-as-decider** for a top-level transaction (no CONTEXT related to the
2076 BEGIN), this is the address to which PREPARE_INFERIORS,
2077 CONFIRM_TRANSACTION, CANCEL_TRANSACTION,
2078 CANCEL_INFERIORS and REQUEST_INFERIOR_STATUSES messages are
2079 to be sent; if a CONTEXT was related to the BEGIN this parameter is absent
2080

2081 **transaction-identifier** identifies the new Decider (Composer or Coordinator)
2082 within the scope of the address-as-decider. If this is not a top-level transaction,
2083 the transaction-identifier is optional, but if present shall be the inferior-identifier
2084 used in the enrolment with the Superior identified by the CONTEXT related to
2085 the BEGIN.
2086

2087 **inferior handle** Shall be absent if this is a top-level transaction and may or may
2088 not be present otherwise. (Presence or absence will be determined by the nature
2089 of the Superior identified in the CONTEXT related to the BEGIN). If present, the
2090 inferior handle will identify this new business transaction as in the inferiors-list
2091 parameters in messages between the Superior identified in the CONTEXT related
2092 to the BEGIN (acting as a Decider) and its Terminator. The value shall be
2093 different for each enrolled Inferior of that Superior.
2094

2095 **address-as-inferior** This parameter shall be absent if this is a top-level
2096 transaction and may be present, at implementation option otherwise. If present, it
2097 shall be the address-as-inferior used in the enrolment with the Superior identified
2098 by the CONTEXT related to the BEGIN. If this is a top-level transaction
2099

2100 **qualifiers** standardised or other qualifiers.
2101

2102 At implementation option, the “address-as-decider” and/or “address-as-inferior” and the
2103 “address-as-superior” in the related CONTEXT may be the same or may be different. There
2104 is no general requirement that they even use the same bindings. Any may also be the same as
2105 the target address of the BEGIN message (the inferior identifier on messages will ensure they
2106 are applied to the appropriate Composer or Coordinator).
2107

2108 No FAULT messages are issued on receiving BEGUN.

2109 **PREPARE_INFERIORS**

2110
2111 Sent from a Terminator to a Decider, but only if it is a Cohesion Composer, to tell it to
2112 prepare all or some of its inferiors, by sending PREPARE to any that have not already sent
2113 PREPARED, RESIGN or CANCELLED to the Decider (Composer) on its relationships as
2114 Superior. If the inferiors-list parameter is absent, the request applies to all the inferiors; if the
2115 parameter is present, it applies only to the identified inferiors of the Decider (Composer).
2116

Parameter	Type
target address	BTP address
reply address	BTP address

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

2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157

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

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

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

inferiors-list defines which of the Inferiors of this Decider preparation is requested for. If this parameter is absent, the PREPARE applies to all Inferiors.

qualifiers standardised or other qualifiers.

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).

Types of FAULT possible (sent to Superior address)

General

InvalidDecider – if Decider address is unknown

UnknownTransaction – if the transaction-identifier is unknown

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

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

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

CONFIRM_TRANSACTION

2158 Sent from a Terminator to a Decider to request confirmation of the business transaction. If the
 2159 business transaction is a Cohesion, the confirm-set is specified by the “inferiors-list”
 2160 parameter.
 2161

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

2162
 2163 **target address** the address to which the CONFIRM_TRANSACTION message
 2164 is sent. This will be the address-as-decider on the BEGUN message.
 2165

2166 **reply address** the address of the Terminator sending the
 2167 CONFIRM_TRANSACTION message.
 2168

2169 **transaction identifier** identifies the Decider. This will be the transaction-
 2170 identifier from the BEGUN message.
 2171

2172 **inferiors-list** defines which Inferiors enrolled with the Decider, if it is a
 2173 Cohesion Composer, are to be confirmed. Shall be absent if the Decider is an
 2174 Atom Coordinator.
 2175

2176 **report hazard** Defines whether the Terminator wishes to be informed of hazard
 2177 events and contradictory decisions within the business transaction. If “report
 2178 hazard” is “true”, the receiver will wait until responses (CONFIRMED,
 2179 CANCELLED or HAZARD) have been received from all of its inferiors,
 2180 ensuring that any hazard events are reported. If “report hazard” is “false”, the
 2181 Decider will reply with CONFIRM_COMPLETE or CANCEL_COMPLETE as
 2182 soon as the decision for the transaction is known.
 2183

2184 **qualifiers** standardised or other qualifiers.
 2185

2186 If the “inferiors-list” parameter is present, the Inferiors identified shall be the “confirm-set” of
 2187 the Cohesion. If the parameter is absent and the business transaction is a Cohesion, the
 2188 “confirm-set” shall be all remaining Inferiors. If the business transaction is an Atom, the
 2189 “confirm-set” is automatically all the Inferiors.
 2190

2191 Any Inferiors from which RESIGN is received are not counted in the confirm-set.
 2192

2193 If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED
 2194 has not been received, PREPARE shall be issued to that Inferior.

2195

2196
2197
2198
2199
2200

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

2201
2202
2203
2204
2205
2206
2207

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

2208
2209

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

2210
2211
2212

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

2213
2214

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

2215
2216
2217
2218
2219

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

2220
2221

Types of FAULT possible (sent to reply address)

2222
2223

General

2224

InvalidDecider – if Decider address is unknown

2225

UnknownTransaction – if the transaction-identifier is unknown

2226

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

2227

WrongState – if a CANCEL_TRANSACTION has already been

2228

received .

2229

2230

The form CONFIRM_TRANSACTION/all refers to a CONFIRM_TRANSACTION message

2231

where the “inferiors-list” parameter is absent. The form

2232

CONFIRM_TRANSACTION/specific refers to a CONFIRM_TRANSACTION message

2233

where the “inferiors-list” parameter is present.

2234

2235

TRANSACTION_CONFIRMED

2236

2237

A Decider sends TRANSACTION_CONFIRMED to a Terminator in reply to

2238

CONFIRM_TRANSACTION if all of the confirm-set confirms (and, for a Cohesion, all other

2239 Inferiors cancel) without reporting hazards, or if the Decider made a confirm decision and the
 2240 CONFIRM_TRANSACTION had a “report-hazards” value of “false”.
 2241

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

2242
 2243 **target address** the address to which the TRANSACTION_CONFIRMED is
 2244 sent., this will be the reply address from the CONFIRM_TRANSACTION
 2245 message.
 2246

2247 **address-as-decider** the address-as-decider of the Decider as on the BEGUN
 2248 message (with the transaction identifier, this determines who the message is
 2249 from).
 2250

2251 **transaction identifier** the transaction identifier as on the BEGUN message (i.e.
 2252 the identifier of the Decider as a whole).
 2253

2254 **qualifiers** standardised or other qualifiers.
 2255

2256 Types of FAULT possible (sent to address-as-decider)
 2257

2258 *General*

2259 *InvalidTerminator* – if Terminator address is unknown

2260 *UnknownTransaction* – if the transaction-identifier is unknown
 2261

2262 **CANCEL_TRANSACTION**

2263 Sent by a Terminator to a Decider at any time before CONFIRM_TRANSACTION has been
 2264 sent.
 2265
 2266

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

2267

2268 **target address** the address to which the CANCEL_TRANSACTION message is
2269 sent. This will be the decider-address from the BEGUN message.

2270
2271 **reply address** the address of the Terminator sending the
2272 CANCEL_TRANSACTION message.

2273
2274 **transaction identifier** identifies the Decider and will be the transaction-identifier
2275 from the BEGUN message.

2276
2277 **report hazard** Defines whether the Terminator wishes to be informed of hazard
2278 events and contradictory decisions within the business transaction. If “report
2279 hazard” is “true”, the receiver will wait until responses (CONFIRMED,
2280 CANCELLED or HAZARD) have been received from all of its inferiors,
2281 ensuring that any hazard events are reported. If “report hazard” is “false”, the
2282 Decider will reply with TRANSACTION_CANCELLED immediately.

2283
2284 **qualifiers** standardised or other qualifiers.

2285
2286 The business transaction is cancelled – this is propagated to any remaining Inferiors by
2287 issuing CANCEL to them. No more Inferiors will be permitted to enrol.

2288
2289 Types of FAULT possible (sent to Superior address)

2290
2291 *General*
2292 *InvalidDecider* – if Decider address is unknown
2293 *UnknownTransaction* – if the transaction-identifier is unknown
2294 *WrongState* – if a CONFIRM_TRANSACTION has been received by
2295 this Composer.

2296
2297
2298 **CANCEL_INFERIORS**

2299
2300 Sent by a Terminator to a Decider, but only if is a Cohesion Composer, at any time before
2301 CONFIRM_TRANSACTION or CANCEL_TRANSACTION has been sent.

2302

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

2303
2304 **target address** the address to which the CANCEL_TRANSACTION message is
2305 sent. This will be the decider-address from the BEGUN message.

2306
 2307 **reply address** the address of the Terminator sending the
 2308 CANCEL_TRANSACTION message.
 2309
 2310 **transaction identifier** identifies the Decider and will be the transaction-identifier
 2311 from the BEGUN message.
 2312
 2313 **inferiors-list** defines which of the Inferiors of this Decider are to be cancelled.
 2314
 2315 **qualifiers** standardised or other qualifiers.

2316
 2317
 2318 Only the Inferiors identified in the inferiors-list are to be cancelled. Any other inferiors are
 2319 unaffected by a CANCEL_INFERIORS. Further Inferiors may be enrolled.
 2320

2321 Note – A CANCEL_INFERIORS all of the currently enrolled Inferiors will
 2322 leave the cohesion ‘empty’, but permitted to continue with new Inferiors, if
 2323 any enrol.

2324
 2325 Types of FAULT possible (sent to Superior address)

2326
 2327 *General*
 2328 *InvalidDecider* – if Decider address is unknown
 2329 *UnknownTransaction* – if the transaction-identifier is unknown
 2330 *InvalidInferior* – if an inferior-handle on the inferiors-list is unknown
 2331 *WrongState* – if a CONFIRM_TRANSACTION or
 2332 CANCEL_TRANSACTION has been received by this Composer.
 2333
 2334
 2335

2336 **TRANSACTION_CANCELLED**

2337
 2338 A Decider sends TRANSACTION_CANCELLED to a Terminator in reply to
 2339 REQUEST_CANCEL or in reply to CONFIRM_TRANSACTION if the Decider decided to
 2340 cancel. In both cases, TRANSACTION_CANCELLED is used only if all Inferiors cancelled
 2341 without reporting hazards or the CANCEL_TRANSACTION or
 2342 CONFIRM_TRANSACTION had a “report-hazard” value of “false.”
 2343

Parameter

target address	BTP address
address-as-decider	BTP address
transaction-identifier	identifier

qualifiers List of qualifiers

2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374

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

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

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

qualifiers standardised or other qualifiers.

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

General

InvalidTerminator – if Terminator address is unknown

UnknownTransaction – if the transaction-identifier is unknown

REQUEST_INFERIOR_STATUSES

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

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

2375
2376
2377
2378
2379

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.

2380
2381 **reply address** the address to which the replying INFERIOR_STATUSES is to
2382 be sent

2383
2384 **target-identifier** identifies the transaction (or transaction tree node) within the
2385 scope of the target address. When the message is used to a Decider, this will be
2386 the transaction-identifier from the BEGUN message. Otherwise it will be the
2387 superior-identifier from a CONTEXT or an inferior-identifier from an ENROL
2388 message.

2389
2390 **inferiors-list** defines which inferiors enrolled with the target are to be included
2391 in the INFERIOR_STATUSES. If the list is absent, the status of all enrolled
2392 inferiors will be reported.

2393
2394 **qualifiers** standardised or other qualifiers.

2395
2396 Types of FAULT possible (sent to reply-address)

2397
2398 **General**
2399 **StatusRefused** – if the receiver is not prepared to report its status to the
2400 sender of this message. This FAULT type shall not be issued when a Decider
2401 receives REQUEST_STATUSES from the Terminator.

2402 **UnknownTransaction** – if the transaction-identifier is unknown

2403
2404
2405 The form REQUEST_INFERIOR_STATUSES/all refers to a REQUEST_STATUS with the
2406 inferiors-list absent. The form REQUEST_INFERIOR_STATUS/specific refers to a
2407 REQUEST_INFERIOR_STATUS with the inferiors-list present.

2408 2409 INFERIOR_STATUSES

2410
2411 Sent by a Decider to report the status of all or some of its inferiors in response to a
2412 REQUEST_INFERIOR_STATUSES, PREPARE_INFERIORS, CANCEL_INFERIORS,
2413 CANCEL_TRANSACTION with “report-hazard” value of “true” and
2414 CONFIRM_TRANSACTION with “report-hazard”value of “true”. It is also used by any
2415 actor in response to a received REQUEST_INFERIOR_STATUSES to report the status of
2416 inferiors, if there are any.

2417

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

2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433

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	Inferior handle, identifying which inferior this Status-item contains information for.
Status	One of the status values below (these are a subset of those for STATUS)
Qualifiers	A list of qualifiers as received from the particular inferior or associated with the inferior in earlier messages (e.g. an Inferior name qualifier).

2434
2435
2436
2437

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

status value	Meaning
<i>active</i>	The Inferior is enrolled
<i>resigned</i>	RESIGNED has been received from the Inferior
<i>preparing</i>	PREPARE has been sent to the inferior, none of PREPARED, RESIGNED, CANCELLED, HAZARD have been received
<i>prepared</i>	PREPARED has been received
<i>autonomously confirmed</i>	CONFIRMED/auto has been received, no completion message has been sent
<i>autonomously cancelled</i>	PREPARED had been received, and since then CANCELLED has been received but no completion message has been sent

status value	Meaning
<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)

2438

2439

2440

2441

2442

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.

2443

2444

2445

2446

2447

2448

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).

2449

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

2450

2451

General

2452

InvalidTerminator – if Terminator address is unknown

2453

UnknownTransaction – if the transaction-identifier is unknown

2454

2455

2456

2457

2458

Groups – combinations of related messages

2459

2460

2461

2462

2463

2464

2465

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.

2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493

CONTEXT & application message

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

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).

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).

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.

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.

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.

2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510

CONTEXT_REPLY & ENROL

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

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

2511 The target address of the ENROL message is omitted.
2512
2513 The actor receiving the related group will use the retained Superior address from the
2514 CONTEXT sent earlier to forward the ENROL. When doing so, it changes the ENROL to
2515 ask for a response (if it was an ENROL/no-rsp-req) and supplies its own address as the
2516 “reply-address”, remembering the original “reply-address” if there was one.
2517
2518 If ENROLLED is received and the original received ENROL was ENROL/rsp-req, the
2519 ENROLLED is forwarded back to the original “reply-address”.
2520
2521 If this attempt fails (i.e. ENROLLED is not received), and the “completion-status” of the
2522 CONTEXT_REPLY was “related”, the actor is required to ensure that the Superior does
2523 not proceed to confirmation. How this is achieved is an implementation option, but must
2524 take account of the possibility that direct communication with the Superior may fail. (One
2525 method is to prevent CONFIRM_TRANSACTION being sent to the Superior (in its role
2526 as Decider); another is to enrol as another Inferior before sending the original CONTEXT
2527 out with an application message). If the Superior is a sub-coordinator or sub-composer,
2528 an enrolment failure must ensure the sub-coordinator does not send PREPARED to its
2529 own Superior.
2530
2531 If the actor receiving the related group is also the Superior (i.e. it has the same binding
2532 address), the explicit forwarding of the ENROL is not required, but the resultant effect –
2533 that if enrolment fails the Superior does not confirm or issue PREPARED – shall be the
2534 same.
2535
2536 A CONTEXT_REPLY & ENROL group may contain multiple ENROL messages, for
2537 several Inferiors. Each ENROL shall be forwarded and an ENROLLED reply received
2538 before the Superior is allowed to confirm if the “completion-status” in the
2539 CONTEXT_REPLY was “related”.
2540
2541 When the group is constructed, if the CONTEXT had “superior-type” value of “atom”,
2542 the “completion-status” of the CONTEXT_REPLY shall be “related”. If the “superior-
2543 type” was “cohesive”, the “completion-status” shall be “completed” or “related” (as
2544 required by the application). If the value is “completed”, the actor receiving the group
2545 shall forward the ENROLS, but is not required to (though it may) prevent confirmation.
2546
2547 **CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED**
2548
2549 This combination is characterised by a related CONTEXT_REPLY and either or both of
2550 PREPARED and CANCELLED, with or without ENROL.
2551
2552 **Meaning:** If ENROL is present, the meaning and required processing is the same as for
2553 CONTEXT_REPLY & ENROL. The PREPARED or CANCELLED message(s) are
2554 forwarded to the Superior identified in the CONTEXT message this CONTEXT_REPLY
2555 is replying to.
2556

2557
2558

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

2559

2560

2561

2562

2563

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

2564

2565

2566

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

2567

2568

2569

2570

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

2571

2572

2573

2574

2575

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

2576

2577

2578

2579

2580

2581

2582

2583

2584

CONTEXT_REPLY & ENROL & application message (& PREPARED)

2585

2586

2587

2588

2589

The presence and details of this section are part of the proposed solution to issue 82, which was discussed at the BTP committee conference call on 16 January 2002, but for which decision was deferred. Accordingly it may be modified or removed when issue 82 is finalised.

2590

2591

2592

2593

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

2594

2595

2596

2597

2598

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

2599

2600

2601

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

2602
2603 The processing of ENROL and PREPARED messages is the same as for the previous
2604 groups.
2605
2606 This group can be used when participation in business transaction (normally a cohesion),
2607 is initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with
2608 some associated application semantic, performs some work for the transaction and sends
2609 an application message with a related ENROL. The CONTEXT_REPLY allows the
2610 addressing of the application (and the CONTEXT_REPLY) to be distinct from that of the
2611 Superior.
2612
2613 The actor receiving the group may associate the “inferior-handle” received on the
2614 ENROLLED with the application message in a manner that is visible to the application
2615 receiving the message.
2616

2617 **BEGUN & CONTEXT**

2618
2619 **Meaning:** the CONTEXT is that for the new business transaction, containing the
2620 Superior address.
2621
2622 **Target address:** the target address is that of the BEGUN message – this will be the reply
2623 address of the earlier BEGIN message.
2624

2625 **BEGIN & CONTEXT**

2626
2627 **Meaning:** the new business transaction is to be an Inferior (sub-coordinator or sub-
2628 composer) of the Superior identified by the CONTEXT. The Factory (receiver of the
2629 BEGIN) will perform the enrolment.
2630
2631 **Target address:** the target address is that of the BEGIN – this will be the address of the
2632 Factory.
2633

2634 **Standard qualifiers**

2635
2636 The following qualifiers are expected to be of general use to many applications and
2637 environments. The URI “urn:oasis:names:tc:BTP:qualifiers” is used in the
2638 Qualifier group value for the qualifiers defined here.
2639

2640 **Transaction timelimit**

2641
2642
2643 The transaction timelimit allows the Superior (or an application element initiating the
2644 business transaction) to indicate the expected length of the active phase, and thus give an
2645 indication to the Inferior of when it would be appropriate to initiate cancellation if the active
2646 phase appears to continue too long. The time limit ends (the clock stops) when the Inferior
2647 decides to be prepared and issues PREPARED to the Superior.
2648

2649 It should be noted that the expiry of the time limit does not change the permissible actions of
2650 the Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is
2651 **permitted** to initiate cancellation for internal reasons. The `timelimit` gives an indication to the
2652 entity of when it will be useful to exercise this right.

2653
2654 The qualifier is propagated on a `CONTEXT` message.

2655
2656 The “Qualifier name” shall be “`transaction-timelimit`”.

2657
2658 The “Content” shall contain the following field:

Content field	Type
<code>Timelimit</code>	Integer

2660
2661 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
2662 time of transmission of the containing `CONTEXT`, of the active phase of the business
2663 transaction.

2664
2665 **Inferior timeout**

2666
2667 This qualifier allows an Inferior to limit the duration of its “promise”, when sending
2668 `PREPARED`, that it will maintain the ability to confirm or cancel the effects of all associated
2669 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or
2670 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and
2671 can apply the decision indicated in the qualifier.

2672
2673 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply
2674 a confirm or cancel decision before the `CONFIRM` or `CANCEL` is received and before this
2675 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,
2676 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
2677 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
2678 expiry of this timeout, is liable to cause contradictory decisions across the business
2679 transaction. BTP ensures that at least the occurrence of such a contradiction will be
2680 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic
2681 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
2682 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
2683 change in the probability that it will.

2684
2685 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
2686 intended decision, only that is at liberty to do so. An implementation may choose to only
2687 apply the decision if there is contention for the underlying resource, for example.

2688 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for
2689 the business transaction are made before these timeouts expire (and allow a margin of error
2690 for network latency etc.).

2691

2692 The qualifier may be present on a PREPARED message. If the PREPARED message has the
2693 “default is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall
2694 have the value “cancel”.

2695
2696 The “Qualifier name” shall be “inferior-timeout”.

2697
2698 The “Content” shall contain the following fields:
2699

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

2700
2701 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
2702 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
2703 effects of the associated operations, as ordered by the receiving Superior.
2704

2705 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an
2706 autonomous decision is made.
2707

2708 **Minimum inferior timeout**

2709
2710 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the
2711 Inferior. If a Superior knows that the decision for the business transaction will not be
2712 determined for some period, it can require that Inferiors do not send PREPARED messages
2713 with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to
2714 send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with
2715 CANCELLED.
2716

2717 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If
2718 present on more than one, and with different values of the MinimumTimeout field, the value
2719 on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall
2720 prevail over either of the others.
2721

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

2723
2724 The “Content” shall contain the following field:
2725

Content field	Type
MinimumTimeout	Integer

2726
2727 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be
2728 acceptable in the Inferior timeout qualifier on an answering PREPARED message.
2729

2730 **Inferior name**

2731

2732 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on
2733 INFERIOR_STATUSES and thus allow the Terminator to determine which Inferior (of the
2734 Composer or Coordinator) is related to which application work. This is in addition to the
2735 “inferior handle” field. The name can be human-readable and can also be used in fault
2736 tracing, debugging and auditing.
2737

2738 The name is never used by the BTP actors themselves to identify each other or to direct
2739 messages. (The BTP actors use the addresses and the identifiers in the message parameters
2740 for those purposes.)
2741

2742 This specification makes no requirement that the names are unambiguous within any scope
2743 (unlike the “inferior-handle” on ENROLLED and BEGUN, which is required to be
2744 unambiguous within the scope of the Decider). Other specifications, including those defining
2745 use of BTP with a particular application may place requirements on the use and form of the
2746 names. (This may include reference to information passed in application messages or in other,
2747 non-standardised, qualifiers.)
2748

2749 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item
2750 in INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if
2751 present, the same qualifier value **should** be included in the consequent ENROL. If
2752 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an
2753 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.
2754

2755 The “Qualifier -name” shall be “inferior-name”
2756

2757 The “Content” shall contain the following fields:
2758

Content field	Type
inferior-name	String

2759 **Inferior name** the name assigned to the enrolling Inferior.
2760
2761

2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
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

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

2808 business transaction and on features of the implementation (e.g. making of a persistent record
2809 of the decision means that the information will survive at least some failures that otherwise
2810 lose state information, but the level of survival depends on the purpose of the
2811 implementation). [Table 2](#) and [Table 3](#) define the decision events.

2812
2813 In some cases, an implementation may not need to make an active change to have a persistent
2814 record of a decision, provided that the implementation will restore itself to the appropriate
2815 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,
2816 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via
2817 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a
2818 record of “decide to cancel autonomously”, provided it always updated such a record as part
2819 of the “apply ordered confirmation” decision event.

2820
2821 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
2822 PREPARE indicates that the application exchange (to associate operations with the Inferior)
2823 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
2824 state corresponding to an incomplete application exchange. However, implementations are
2825 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
2826 that experiences failure after sending PREPARE may, on recovery, have no information
2827 about the transaction, in which case it is considered to be in the completed state (Z), which
2828 will imply the cancellation of the Inferior and its associated operations.

2829
2830 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its
2831 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a
2832 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
2833 persistent information (which would combine both superior and inferior information, pointing
2834 both up and down the tree).

2835
2836

2837 **Disruptions – failure events**

2838
2839 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or
2840 may) cause a change of state. The disruption events in the state tables model different extents
2841 of loss of state information. An implementation is not required to exhibit all the possible
2842 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
2843 possible disruption.

2844
2845 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,
2846 which involves possible interruption of service and loss of messages in transit, but no change
2847 of state (either because no state information was lost, or because recovery from persistent
2848 information restores the implementation to the same state). The “disruption 0” event would
2849 typically be an appropriate abstraction for a communication failure.

2850
2851

2851 **Invalid cells and assumptions of the communication mechanism**

2852
2853 The empty cells in state table represent events that cannot happen. For events corresponding
2854 to sending a message or any of the decision events, this prohibition is absolute – e.g. a

2855 conformant implementation in the Superior active state “B1” will not send CONFIRM. For
2856 events corresponding to receiving a message, the interpretation depends on the properties of
2857 the underlying communications mechanism.

2858

2859 For all communication mechanisms, it is assumed that

- 2860 a) the two directions of the Superior:Inferior communication are not synchronised –
- 2861 that is messages travelling in opposite directions can cross each other to any
- 2862 degree; any number of messages may be in transit in either direction; and
- 2863 b) messages may be lost arbitrarily

2864

2865 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered
2866 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a
2867 state where the corresponding cell is empty indicates that the far-side has sent a message out
2868 of order – a FAULT message with the Fault Type “WrongState” can be returned.

2869

2870 If the communication mechanisms cannot guarantee ordered delivery, then messages received
2871 where the corresponding cell is empty should be ignored. Assuming the far-side is
2872 conformant, these messages can assumed to be “stale” and have been overtaken by messages
2873 sent later but already delivered. (If the far-side is non-conformant, there is a problem
2874 anyway).

2875

2876 **Meaning of state table events**

2877

2878 The tables in this section define the events (rows) in the state tables. [Table 1](#) defines
2879 the events corresponding to sending or receiving BTP messages and the disruption events.
2880 [Table 2](#) describes the decision events for an Inferior, [Table 3](#) those for a
2881 Superior.

2882

2883 The decision events for a Superior, defined in [Table 3](#) cannot be specified without
2884 reference to other Inferiors to which it is Superior and to its relation with the application or
2885 other entity that (acting ultimately on behalf of the application) drives it.

2886

2887 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and
2888 which are to be treated as an atomic decision unit with (and thus including) the Inferior on
2889 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior
2890 type” of “atom”, this will be all Inferiors established with same Superior address and Superior
2891 identifier except those from which RESIGN has been received. If the CONTEXT had
2892 “superior type” of “cohesion”, the “remaining Inferiors” excludes any that it has been
2893 determined will be cancelled, as well as any that have resigned – in other words it includes
2894 only those for which a confirm decision is still possible or has been made. The determination
2895 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some
2896 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this
2897 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.

2898

2899 In order to ensure that the confirmation decision is delivered to all remaining Inferiors,
2900 despite failures, the Superior must persistently record which these Inferiors are (i.e. their
2901 addresses and identifiers). It must also either record that the decision is confirm, or ensure

2902 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it
 2903 will be told about it. This latter would apply if the Superior were also BTP Inferior to another
 2904 entity which persisted a confirm decision (or recursively deferred it still higher). However,
 2905 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
 2906 behaviour of asking another entity to make (and persist) the confirm decision is termed
 2907 "offering confirmation" - the Superior offers the possible confirmation of itself, and its
 2908 remaining Inferiors to some other entity. If that entity (or something higher up) then does
 2909 make and persist a confirm decision, the Superior is "instructed to confirm" (which is
 2910 equivalent BTP CONFIRM).

2911
 2912 The application, or an entity acting indirectly on behalf of the application, may request a
 2913 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
 2914 more operations associated with the Inferior. Following a request to prepare all remaining
 2915 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
 2916 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
 2917 application.)

2918
 2919 The application, or an entity acting indirectly on behalf of the application, may also request
 2920 confirmation. This means the Superior is to attempt to make and persist a confirm decision
 2921 itself, rather than offer confirmation.

2922
 2923

2924

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

2925

2926

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)

2927

2928

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)

2929

2930

Persistent information

2931

2932

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.

2933

2934

2935

2936

2937

2938

2939

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.

2940

2941

2942

2943

2944

2945

2946

2947

2948

2949

2950

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.

2951

2952

2953

2954

2955

2956

2957

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

2961
2962

Table 6: Superior state table – normal forward progression

	I 1	A 1	B 1	C 1	D 1	E 1	E 2	F 1	F 2
recei ve ENROL/rsp-req	A1								
recei ve ENROL/no-rsp-req	B1								
recei ve RESI GN/rsp-req	Y1		C1	C1	C1				
recei ve RESI GN/no-rsp-req	Z		Z	Z	Z				
recei ve PREPARED	Y1		E1		E1	E1		F1	
recei ve PREPARED/cancel	Y1		E2		E2		E2	F1	
recei ve CONFIR MED/auto	Q1		H1		H1	H1		F1	
recei ve CONFIR MED/response								F2	F2
recei ve CANCELLED	Y1		Z		Z	J1	J1	K1	
recei ve HAZARD	P1	P1	P1		P1	P1	P1	P3	
recei ve INF_STATE/acti ve/y	Y1	A1	B1		D1				
recei ve INF_STATE/acti ve			B1		D1				
recei ve INF_STATE/unknown			Z	Z	Z				
send ENROLLED		B1							
send RESI GNED				Z					
send PREPARE					D1	E1	E2		
send CONFIR M_ONE_PHASE									
send CONFIR M								F1	
send CANCEL									
send CONTRADI CTI ON									
send SUP_STATE/acti ve/y			B1						
send SUP_STATE/acti ve			B1						
send SUP_STATE/prepared-rcvd/y						E1	E2		
send SUP_STATE/prepared-rcvd						E1	E2		
send SUP_STATE/unknown									
deci de to confi rm one-phase			S1			S1	S1		
deci de to prepare			D1						
deci de to confi rm						F1	F1		
deci de to cancel			G1		G1	G1	Z		
remove persi stent i nformati on									Z
record contradi cti on									
di srupti on I	Z	Z	Z	Z	Z	Z	Z		F1
di srupti on II						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								
recei ve ENROL/no-rsp-req								
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								
recei ve ENROL/no-rsp-req								
recei ve RESI GN/rsp-req								C1
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
recei ve ENROL/no-rsp-req		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		

2966

2967

2967

2968

Table 10: Inferior state table – normal forward progression

	i 1	a1	b1	c1	d1	e1	e2	f1	f2
send ENROL/rsp-req send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD	a1 b1			c1 z		e1 e2			
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown		a1	b1 b1		d1 d1				
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION		b1		z					
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown		b1 b1	b1 b1	c1 c1		e1 e1 e1 e1	e2 e2 e2 e2		
decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem			c1 e1 e2		c1 e1 e2				
di srupti on I di srupti on II di srupti on III		z	z	z	z b1			e1	e2

2969

2970

Table 11: Inferior state table – cancellation and contradiction

	g1	g2	h1	h2	j1	j2	k1	k2	l1	l2
send ENROL/rsp-req send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD			h1		j1		k1		l1	
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown										
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION	g1	g2	h1 s3 h2 h2 l1 l2		j1 s4 k1 j2 j2 k2		k1 k2 k2		l1 l2 l2	
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown	x1	x2	h1 h1 h1 h1 l1		j1 j1 j1 j1 j2 j2		k2 k2		l1	
decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem	n1 p2	n1 p2	m1		z		z		z	
disruption I disruption II disruption III	e1	e2	h1		j1		j1 k1 j1		h1 l1 h1	

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

	m1	n1	p1	p2	q1
send ENROL/rsp-req send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD	z	z	p1	p2	q1
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	m1	n1	p1 s5 z	p2 s5 z	q1 s6 q1 q1 z
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown		z	p1 p1 p1	p2 p2 p2	q1 q1 q1 q1
decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem					q1 q1
disruption I disruption II disruption III	z	z d1 b1	z		

Table 13: Inferior state table – request confirm states

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

2975

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

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

2976

2977

2978

2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023

Failure Recovery

Types of failure

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

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

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

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

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

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

3024 After recovery from node failure, the implementation behaves much as if a communication
3025 failure had occurred.

3026

3027 **Persistent information**

3028

3029 BTP requires that some decision events are persisted – that information recording an
3030 Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s
3031 autonomous decision survive failure. Making the first two decisions persistent ensures that a
3032 consistent decision can be reached for the business transaction and that it is delivered to all
3033 involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to
3034 ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the
3035 contradiction will be reported to the Superior, despite failures.

3036

3037 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
3038 active state (unlike many transaction protocols, where a communication or endpoint failure in
3039 active state would invariably cause rollback of the transaction). Recovery in the active state
3040 may require that the application exchange is resynchronised as well – BTP does not directly
3041 support this, but does allow continuation of the business transaction as such. In the state
3042 tables, from some states, there are several levels of disruption, distinguished by which state
3043 the implementation transits to – this represents the survival of different extents of state
3044 information over failure and recovery. The different levels of disruption describe legitimate
3045 states for the endpoint to be in after it has recovered – **they do not require that all
3046 implementations are able to exhibit the appropriate partial loss of state information.**

3047

3048 The absence of a destination state for the disruption events means that such a transition is not
3049 legitimate – thus, for example, an Inferior that has decided to be prepared will always recover
3050 to the same state, by virtue of the information persisted in the “decide to be prepared” event.

3051

3052 Apart from the (optional) recovery in active state, BTP follows the well-known presume-
3053 abort model – it is only required that information be persisted when decisions are made (and
3054 not, e.g. on enrolment). This means that on recovery, one side may have persistent
3055 information but the other does not. This occurs when an Inferior has decided to be prepared
3056 but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the
3057 Superior did confirm, and the Inferior applied the confirm, removed its persistent information
3058 but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it
3059 still had the persistent information when the failure occurred).

3060

3061 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to
3062 re-establish communication with the Superior, to apply a confirm decision and to apply a
3063 cancel decision. It will thus need to include

3063

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

3064

Superior address (as on CONTEXT)

3065

Superior identifier (as on CONTEXT)

3066

default-is-cancel value (as on PREPARED)

3067

3068 The information needed to apply confirm/cancel decisions will depend on the application and
3069 the associated operations. It may also normally be necessary to persist any qualifiers that

3070 were sent with the PREPARED message or application messages sent with the PREPARED,
3071 since the PREPARED message will be repeated if a failure occurs.

3072
3073 A Superior must record corresponding information to allow it to re-establish communication
3074 with the Inferior:

3075 Inferior address (as on ENROL)

3076 Inferior identifier (as on ENROL)

3077

3078 A Superior that is the Decider for the business transaction need only persist this information
3079 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A
3080 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as
3081 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as
3082 Superior (to this Inferior) as part of the persistent information of its decision to be prepared
3083 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and
3084 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If
3085 CONFIRM is received, the persistent information may be changed to show the confirm
3086 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.
3087 If the persistent information is left unchanged and there is a node failure, on recovery the
3088 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision
3089 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

3090

3091 After failure, an implementation may not be able to restore an endpoint to the appropriate
3092 state immediately – in particular, the necessary persistent information may be inaccessible,
3093 although the implementation can respond to received BTP messages. In such a case, a
3094 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a “reply-
3095 requested” value “false”) with SUPERIOR_STATE/inaccessible and an Inferior to any BTP
3096 message except SUPERIOR_STATE/* with “INFERIOR_STATE/inaccessible. Receipt of
3097 the *_STATE/inaccessible messages has no effect on the endpoint state.

3098

3099 Redirection

3100

3101 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
3102 that there are cases where one side will attempt to re-establish communication when there is
3103 no persistent information for the relationship at the far-end. In such cases, it is important the
3104 side that is attempting recovery can distinguish between unsuccessful attempts to connect to
3105 the holder of the persistent information and when the information no longer exists. If the peer
3106 information does not exist, this side can draw conclusions and complete appropriately; if they
3107 merely fail to get through they are stuck in attempting recovery.

3108

3109 Two mechanisms are provided to make it possible that even when one side of a
3110 Superior:Inferior relationship has completed, that a message can eventually get through to
3111 something that can definitively report the status, distinguishing this case from a temporary
3112 inability to access the state of a continuing transaction element. The mechanisms are:

- 3113 o Address fields which provide a “callback address” can be a set of addresses,
3114 which are alternatives one of which is chosen as the target address for the
3115 future message. If the sender of that message finds the address does not work,
3116 it can try a different alternative.

3117 o The REDIRECT message can be used to inform the peer that an address
3118 previously given is no longer valid and to supply a replacement address (or
3119 set of addresses). REDIRECT can be issued either as a response to receipt of
3120 a message or spontaneously.

3121
3122 The two mechanisms can be used in combination, with one or more of the original set of
3123 addresses just being a redirector, which does not itself ever have direct access to the state
3124 information for the transaction, but will respond to any message with an appropriate
3125 REDIRECT.

3126
3127 An alternative implementation approach is to have a single addressable entity that uses the
3128 same address for all transactions, distinguishing them by identifier, and which always
3129 recovers to use the same address. Such an implementation would not need to supply
3130 “backup” addresses (and would only use REDIRECT if it was being permanently migrated).

3131

3132 Terminator:Decider failures

3133

3134 BTP does not provide facilities or impose requirements on the recovery of
3135 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
3136 may survive failures (by retaining knowledge of the Decider’s address and identifier), but this
3137 is an implementation option. Although a Decider (if it decides to confirm) will persist
3138 information about the confirm decision, it is not required, after failure, to remain accessible
3139 using the inferior address it offered to the Terminator. Any such recovery is an
3140 implementation option.

3141

3142 A Decider’s address (as returned on BEGUN) may be a set of addresses, allowing a failed
3143 Decider to be recovered at a different address.

3144

3145 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
3146 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for
3147 a CONFIRM_TRANSACTION that will never arrive, the standard qualifier “Transaction
3148 timelimit” can be used (by the Initiator) to inform the Decider when it can assume the
3149 Terminator will not issue CONFIRM_TRANSACTION and so it (the Decider) should initiate
3150 cancellation.

3151

3152 XML representation of Message Set

3153

3154 This section describes the syntax for BTP messages in XML. These XML messages represent
3155 a midpoint between the abstract messages and what actually gets sent on the wire.

3156

3157 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)
3158 [3121](#)

3159

3160 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:xml

3161

3162 In addition to an XML schema, this specification uses an informal syntax to describe the
3163 structure of the BTP messages. The syntax appears as an XML instance, but the values

3164 contain data types instead of values. The following symbols are appended to some of the
3165 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
3166 these symbols corresponds to "one and only one."
3167

3168 Addresses

3169
3170 As described in the "Abstract Message and Associated Contracts – Addresses" section, a BTP
3171 address comprises three parts, and for a target address only the "additional information" field
3172 is inside the BTP messages. For all BTP messages whose abstract form includes a target
3173 address parameter, the corresponding XML representation includes a "target-additional-
3174 information" element. This element may be omitted if it would be empty.
3175

3176 For other addresses, all three fields are represent, as in:

```
3177 <ctp:some-address>  
3178   <ctp:binding-name>...carrier binding URI...</ctp:binding-name>  
3179   <ctp:binding-address>...carrier specific  
3180   address...</ctp:binding-address>  
3181   <ctp:additional-information>...optional additional addressing  
3182   information...</ctp:additional-information> ?  
3183 </ctp:some-address>
```

3184
3185
3186 A "published" address can be a set of <some-address>, which are alternatives which can be
3187 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to
3188 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice
3189 of which address to use (depending on which binding is preferable.) In the case where
3190 multiple addresses are used for redundancy, a priority attribute can be specified to help the
3191 receiver choose among the addresses- the address with the highest priority should be used,
3192 other things being equal. The priority is used as a hint and does not enforce any behaviour in
3193 the receiver of the message. Default priority is a value of 1.
3194
3195

3196 Qualifiers

3197 The "Qualifier name" is used as the element name, within the namespace of the "Qualifier
3198 group".
3199

3200 Examples:

```
3201 <ctpq:inferior-timeout  
3202   xmlns:ctpq="urn:oasis:names:tc:BTP:qualifiers"  
3203   xmlns:ctp="urn:oasis:names:tc:BTP:xml "  
3204   ctp:must-be-understood="false"  
3205   ctp:to-be-propagated="false">1800</ctpq:inferior-timeout>  
3206  
3207 <auth:username  
3208   xmlns:auth="http://www.example.com/ns/auth"  
3209   xmlns:ctp="urn:oasis:names:tc:BTP:xml "  
3210   ctp:must-be-understood="true"  
3211   ctp:to-be-propagated="true">jtauber</auth:username>  
3212
```

3213 Attributes must-be-understood **has default value “true”** and to-be-propagated has default
3214 value “false”.

3215

3216 Identifiers

3217 Unspecified length strings made of up hexadecimal digits (0->9, A->F). Note: lower case a->f
3218 are not valid.

3219

3220 Examples: "01", "FAB224234CCCC2"

3221

3222 Note – Use of hexadecimal digits avoids problems with character-code representations. The
3223 only operation the BTP implementations have to perform on identifiers is to match them.

3224

3225 Message References

3226 Each BTP message has an optional id attribute to give it a unique identifier. An application
3227 can make use of those identifiers, but no processing is enforced.

3228

3229 Messages

3230

3231 CONTEXT

3232

```
3233 <btp:context id? superior-type="cohesion|atom">  
3234 <btp:superior-address> +  
3235 ...address...  
3236 </btp:superior-address>  
3237 <btp:superior-identifier>...hexstring...</btp:superior-  
3238 identifier>  
3239 <btp:reply-address> ?  
3240 ...address...  
3241 </btp:reply-address>  
3242 <btp:qualifiers> ?  
3243 ...qualifiers...  
3244 </btp:qualifiers>  
3245 </btp:context>
```

3246

3247

3248 CONTEXT_REPLY

3249

```
3250 <btp:context-reply id? superior-type="cohesion|atom">  
3251 <btp:target-additional-information> ?  
3252 ...additional address information...  
3253 </btp:target-additional-information>  
3254 <btp:superior-address> +  
3255 ...address...  
3256 </btp:superior-address>  
3257 <btp:superior-identifier>...hexstring...</btp:superior-  
3258 identifier>  
3259 <completion-status>completed|related|repudiated</completion-  
3260 status>  
3261 <btp:qualifiers> ?  
3262 ...qualifiers...
```

```
3263     </btp:qualifiers>
3264 </btp:context>
```

3265
3266

BEGIN

3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279

```
<btp:begin id? transaction-type="cohesion|atom">
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:reply-address>
    ...address...
  </btp:reply-address>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:begin>
```

3280
3281

BEGUN

3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300

```
<btp:begun id? transaction-type="cohesion|atom">
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:decider-address> ?
    ...address...
  </btp:decider-address>
  <btp:transaction-identifier>...hexstring...</btp:transaction-
  identifier> ?
  <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?
  <btp:inferior-address> ?
    ...address...
  </btp:inferior-address>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:begun>
```

3301
3302

ENROL

3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313

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

```
3314 <btpr:inferior-address> +
3315   ...address...
3316 </btpr:inferior-address>
3317 <btpr:inferior-identifier>...hexstring...</btpr:inferior-
3318 identifier>
3319 <btpr:qualifiers> ?
3320   ...qualifiers...
3321 </btpr:qualifiers>
3322 </btpr:enrol>
```

3323
3324

ENROLLED

```
3326 <btpr:enrolled id?>
3327 <btpr:target-additional-information>
3328   ...additional address information...
3329 </btpr:target-additional-information>
3330 <btpr:inferior-identifier>...hexstring...</btpr:inferior-
3331 identifier>
3332 <btpr:inferior-handle>...hexstring...</btpr:inferior:handle> ?
3333 <btpr:qualifiers> ?
3334   ...qualifiers...
3335 </btpr:qualifiers>
3336 </btpr:enrolled>
```

3337
3338
3339

RESIGN

```
3340 <btpr:resign response-requested="true|false" id?>
3341 <btpr:target-additional-information>
3342   ...additional address information...
3343 </btpr:target-additional-information>
3344 <btpr:superior-identifier>...hexstring...</btpr:superior-
3345 identifier>
3346 <btpr:inferior-address> +
3347   ...address...
3348 </btpr:inferior-address>
3349 <btpr:inferior-identifier>...hexstring...</btpr:inferior-
3350 identifier>
3351 <btpr:qualifiers> ?
3352   ...qualifiers...
3353 </btpr:qualifiers>
3354 </btpr:resign>
```

3355
3356
3357
3358

RESIGNED

```
3359 <btpr:resigned id?>
3360 <btpr:target-additional-information>
3361   ...additional address information...
3362 </btpr:target-additional-information>
```

3363
3364

```
3365     <btp:inferior-identifier>...hexstring...</btp:inferior-
3366 identifier>
3367     <btp:qualifiers> ?
3368     ...qualifiers...
3369     </btp:qualifiers>
3370 </btp:resigned>
```

3371
3372

PREPARE

```
3373
3374
3375     <btp:prepare id?>
3376     <btp:target-additional-information>
3377     ...additional address information...
3378     </btp:target-additional-information>
3379     <btp:inferior-identifier>...hexstring...</btp:inferior-
3380 identifier> ?
3381     <btp:qualifiers> ?
3382     ...qualifiers...
3383     </btp:qualifiers>
3384 </btp:prepare>
```

3385
3386

PREPARED

```
3387
3388
3389     <btp:prepared default-is-cancel="false|true" id?>
3390     <btp:target-additional-information>
3391     ...additional address information...
3392     </btp:target-additional-information>
3393     <btp:superior-identifier>...hexstring...</btp:superior-
3394 identifier>
3395     <btp:inferior-address> +
3396     ...address...
3397     </btp:inferior-address>
3398     <btp:inferior-identifier>...hexstring...</btp:inferior-
3399 identifier>
3400     <btp:qualifiers> ?
3401     ...qualifiers...
3402     </btp:qualifiers>
3403 </btp:prepared>
```

3404
3405

CONFIRM

```
3406
3407
3408     <btp:confirm id?>
3409     <btp:target-additional-information>
3410     ...additional address information...
3411     </btp:target-additional-information>
3412     <btp:inferior-identifier>...hexstring...</btp:inferior-
3413 identifier>
3414     <btp:qualifiers> ?
3415     ...qualifiers...
```

3416 </btp:qualifiers>
3417 </btp:confirm>

3420 CONFIRMED

3421 <btp:confirmed confirmed-received="true|false" id?>
3422 <btp:target-additional-information>
3423 ...additional address information...
3424 </btp:target-additional-information>
3425 <btp:superior-identifier>...hexstring...</btp:superior-
3426 identifier>
3427 <btp:inferior-address> ?
3428 ...address...
3429 </btp:inferior-address>
3430 <btp:inferior-identifier>...hexstring...</btp:inferior-
3431 identifier> ?
3432 <btp:qualifiers> ?
3433 ...qualifiers...
3434 </btp:qualifiers>
3435 </btp:confirmed>

3439 CANCEL

3440 <btp:cancel id?>
3441 <btp:target-additional-information>
3442 ...additional address information...
3443 </btp:target-additional-information>
3444 <btp:inferior-identifier>...hexstring...</btp:inferior-
3445 identifier> ?
3446 <btp:reply-address> ?
3447 ...address...
3448 </btp:reply-address>
3449 <btp:qualifiers> ?
3450 ...qualifiers...
3451 </btp:qualifiers>
3452 </btp:cancel>

3456 CANCELLED

3457 <btp:cancelled id?>
3458 <btp:target-additional-information>
3459 ...additional address information...
3460 </btp:target-additional-information>
3461 <btp:superior-identifier>...hexstring...</btp:superior-
3462 identifier>
3463 <btp:inferior-address> +
3464 ...address...
3465 </btp:inferior-address> ?

```
3467     <btp:inferior-identifier>...hexstring...</btp:inferior-
3468 identifier> ?
3469     <btp:qualifiers> ?
3470     ...qualifiers...
3471     </btp:qualifiers>
3472 </btp:cancelled>
```

CONFIRM_ONE_PHASE

```
3476     <btp:confirm-one-phase report-hazard="true|false" id?>
3477     <btp:target-additional-information>
3478     ...additional address information...
3479     </btp:target-additional-information>
3480     <btp:inferior-identifier>...hexstring...</btp:inferior-
3481 identifier>
3482     <btp:qualifiers> ?
3483     ...qualifiers...
3484     </btp:qualifiers>
3485 </btp:confirm-one-phase>
```

HAZARD

```
3488     <btp:hazard level="mixed|possible" id?>
3489     <btp:target-additional-information>
3490     ...additional address information...
3491     </btp:target-additional-information>
3492     <btp:superior-identifier>...hexstring...</btp:superior-
3493 identifier>
3494     <btp:inferior-address> +
3495     ...address...
3496     </btp:inferior-address>
3497     <btp:inferior-identifier>...hexstring...</btp:inferior-
3498 identifier>
3499     <btp:qualifiers> ?
3500     ...qualifiers...
3501     </btp:qualifiers>
3502 </btp:hazard>
```

CONTRADICTION

```
3503     <btp:contradiction id?>
3504     <btp:target-additional-information>
3505     ...additional address information...
3506     </btp:target-additional-information>
3507     <btp:inferior-identifier>...hexstring...</btp:inferior-
3508 identifier>
3509     <btp:qualifiers> ?
3510     ...qualifiers...
3511     </btp:qualifiers>
```

3518 </btp:contradiction>

3519

3520

3521

SUPERIOR_STATE

3522

3523

```
<btp:superior-state reply-requested="true|false" id?>
```

3524

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

3525

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

3526

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

3527

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

3528

3529

```
<btp:status>active|prepared-
```

3530

```
received|inaccessible|unknown</btp:status>
```

3531

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

3532

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

3533

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

3534

```
</btp:superior-state>
```

3535

3536

3537

INFERIOR_STATE

3538

3539

```
<btp:inferior-state reply-requested="true|false" id?>
```

3540

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

3541

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

3542

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

3543

```
<btp:superior-identifier>...hexstring...</btp:superior-  
identifier>
```

3544

3545

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

3546

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

3547

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

3548

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

3549

3550

```
<btp:status> active| inaccessible|unknown</btp:status>
```

3551

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

3552

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

3553

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

3554

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

3555

3556

3557

3558

3559

REDIRECT

3560

3561

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

3562

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

3563

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

3564

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

3565

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

3566

3567

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

3568

```
3569 <btp:old-address> +
3570 ...address...
3571 </btp:old-address>
3572 <btp:new-address> +
3573 ...address...
3574 </btp:new-address>
3575 <btp:qualifiers> ?
3576 ...qualifiers...
3577 </btp:qualifiers>
3578 </btp:redirect>
```

3579

PREPARE_INFERIORS

3580

3581

3582

3583

3584

3585

3586

3587

3588

3589

3590

3591

3592

3593

3594

3595

3596

3597

3598

3599

3600

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

3601

CONFIRM_TRANSACTION

3602

3603

3604

3605

3606

3607

3608

3609

3610

3611

3612

3613

3614

3615

3616

3617

3618

3619

3620

```
<btp:confirm-transaction report-hazard="true|false" id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:reply-address>
    ...address...
  </btp:reply-address>
  <btp:transaction-identifier>...hexstring...</btp:transaction-
identifier>
  <btp:inferiors-list> ?
    <btp:inferior-handle>...hexstring...</btp:inferior-handle>
+
  </btp:inferiors-list>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:confirm_transaction>
```

3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672

TRANSACTION_CONFIRMED

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

CANCEL_TRANSACTION

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

CANCEL_INFERIORS

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

3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691

TRANSACTION_CANCELLED

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

3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711

REQUEST_INFERIOR_STATUSES

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

3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723

INFERIOR_STATUSES

```
<btp:inferior_statuses id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:responders-address>
    ...address...
  </btp:responders-address>
  <btp:responders-identifier>...hexstring...</btp:responders-
identifier>
  <btp:status-list>
```

```

3724         <btp:status-item> +
3725         <btp:inferior-handle>...hexstring...</btp:inferior-
3726 handle>
3727         <btp:status>active|resigned|preparing|prepared|
3728         autonomously-confirmed|autonomously-cancelled|
3729         confirming|confirmed|cancelling|cancelled|
3730         cancel-contradiction|confirm-contradiction|
3731         hazard</btp:status>
3732         <btp:qualifiers> ?
3733         ...qualifiers...
3734         </btp:qualifiers>
3735         </btp:status-item>
3736 </btp:status-list>
3737 <btp:qualifiers> ?
3738     ...qualifiers...
3739 </btp:qualifiers>
3740 </btp:inferior_statuses>

```

3741

3742

REQUEST_STATUS

3743

3744

3745

3746

3747

3748

3749

3750

3751

3752

3753

3754

3755

3756

3757

```

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

```

3758

STATUS

3759

3760

3761

3762

3763

3764

3765

3766

3767

3768

3769

3770

3771

3772

3773

3774

3775

```

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

```

```
3776 <btq:qualifiers> ?
3777   ...qualifiers...
3778 </btq:qualifiers>
3779 </btq:status>
```

3781 FAULT

```
3782
3783 <btq:fault id?>
3784   <btq:target-additional-information>
3785     ...additional address information...
3786   </btq:target-additional-information>
3787   <btq:superior-identifier>...hexstring...</btq:superior-
3788   identifier> ?
3789   <btq:inferior-identifier>...hexstring...</btq:inferior-
3790   identifier> ?
3791   <btq:fault-type>...fault type name...</btq:fault-type>
3792   <btq:fault-data>...fault data...</btq:fault-data> ?
3793   <btq:qualifiers> ?
3794     ...qualifiers...
3795   </btq:qualifiers>
3796 </btq:fault>
```

3797
3798
3799 The following fault type names are represented by simple strings, corresponding to the entries
3800 defined in the abstract message set:

- 3801
- 3802 o general
- 3803 o unknown-parameter
- 3804 o wrong-state
- 3805 o communication-failure
- 3806 o invalid-superior
- 3807 o duplicate-inferior
- 3808 o unknown-inferior
- 3809

3810 Revisions of this specification may add other fault type names, which shall be simple strings
3811 of letters, numbers and hyphens. If other specifications define fault type names to be used
3812 with BTP, the names shall be URIs.

3813
3814 Fault data can take on various forms:

3815
3816 Free text:

```
3817 <btq:fault-data>...string data...</btq:fault-data>
```

3818
3819 Identifier:

```
3820 <btq:fault-data>...hexstring...</btq:fault-data>
```

3821
3822
3823
3824

3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835

Inferior Identity:

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

3836
3837
3838
3839

Standard qualifiers

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

3840

Transaction timelimit

3841
3842
3843
3844
3845
3846
3847

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

3848

Inferior timeout

3849
3850
3851
3852
3853
3854
3855

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

3856

Minimum inferior timeout

3857
3858
3859
3860
3861
3862

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

3863

Compounding of Messages

3864

3865

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

3866

3867

3868

3869

3870

3871

3872

3873

3874

```
<btq: relatedgroup>
  <btq: context-reply>
  ...<completion-status>related</completion-status> ...
  </btq: context-reply>
  <btq: enrol>...</btq: enrol>
  <btq: prepared>...</btq: prepared>
```

3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892

```
</btp:relatedgroup>
```

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

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

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

3892

3893 **Carrier Protocol Bindings**

3894

3895 The notion of bindings is introduced to act as the glue between the BTP messages and an
3896 underlying transport. A binding specification must define various particulars of how the BTP
3897 messages are carried and some aspects of how the related application messages are carried.
3898 This document specifies two bindings: a SOAP binding and a SOAP + Attachments binding.
3899 However, other bindings could be specified by the Oasis BTP technical committee or by a
3900 third party. For example, in the future a binding might exist to put a BTP message directly on
3901 top of HTTP without the use of SOAP, or a closed community could define their own
3902 binding. To ensure that such specifications are complete, the Binding Proforma defines the
3903 information that must be included in a binding specification.
3904

3905 **Carrier Protocol Binding Proforma**

3906

3907 A BTP carrier binding specification should provide the following information:

3908

3909 **Binding name:** A name for the binding, as used in the “binding name” field of BTP
3910 addresses (and available for declaring the capabilities of an implementation). Binding
3911 specified in this document, and future revisions of this document have binding names that are
3912 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).
3913 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in
3914 this document use numbers to identify the version of the binding, not the version(s) of the
3915 carrier protocol.

3916

3917 **Binding address format:** This section states the format of the “binding address” field of a
3918 BTP address for this binding. For many bindings, this will be a URL of some kind; for other
3919 bindings it may be some other form

3920

3921 **BTP message representation:** This section will define how BTP messages are represented.
3922 For many bindings, the BTP message syntax will be as specified in the XML schema defined
3923 in this document, and the normal string encoding of that XML will be used.

3924

3925 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that
3926 are not related to application messages are sent in either direction between Superior and
3927 Inferior. (i.e. those messages sent directly between BTP actors). This mapping need not be
3928 symmetric (i.e. Superior to Inferior may differ to some degree to Inferior to Superior). The
3929 mapping may define particular rules for particular BTP messages, or messages with particular
3930 parameter values (e.g. the FAULT message with “fault-type” “CommunicationFailure” will
3931 typically not be sent as a BTP message). The mapping states any constraints or requirements
3932 on which BTP may or must be bundled together by compounding.

3933

3934 **Mapping for BTP messages related to application messages:** This section will define how
3935 BTP messages that are related to application messages are sent. A binding specification may
3936 defer details of this to a particular application (e.g. a mapping specification could just say
3937 “the CONTEXT may be carried as a parameter of an application invocation”). Alternatively,

3938 the binding may specify a general method that represents the relationship between application
3939 and BTP messages.

3940

3941 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly
3942 but are treated as implicit in application messages or other BTP messages. This may depend
3943 on particular parameter values of the BTP messages or the application messages.

3944

3945 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
3946 protocol and of BTP shall be defined. This may include definition of which carrier protocol
3947 exceptions are equivalent to a FAULT/communication-failure message.

3948

3949 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.
3950 If BTP addresses with different bindings are be considered to match (for purposes of
3951 identifying the peer Superior/Inferior and redirection), this should be specified here.

3952

3953 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are
3954 imposed by use of this binding should be listed. This would include limitations on which
3955 messages can be sent, which event sequences are supported and restrictions on parameter
3956 values. Such limitations may reduce the usefulness of an implementation, but may be
3957 appropriate in certain environments.

3958

3959 **Other:** Other features of the binding, especially any that will potentially affect interoperability
3960 should be specified here. This may include restrictions or requirements on the use or support
3961 of optional carrier parameters or mechanisms.

3962

3963 **Bindings for request/response carrier protocols**

3964

3965 BTP does not generally follow request/response pattern. In particular, on the outcome
3966 relationship either side may initiate a message – this is an essential part of the presume-abort
3967 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
3968 messages, especially in the control relationship, that do have a request/response pattern.
3969 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The
3970 specification of a binding specification to a request/response carrier protocol needs to state
3971 what rules apply – which messages can be carried by requests, which by responses. The
3972 simplest rule is to send all BTP messages on requests, and let the carrier responses travel back
3973 empty. This would be inefficient in use of network resources, and possibly inconvenient
3974 when used for the BTP request/response pairs.

3975

3976 This section defines a set of rules that allow more efficient use of the carrier, while allowing
3977 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the
3978 carrier response. These rules are specified in this section to enable binding specifications to
3979 reference them, without requiring each binding specification to repeat similar information.

3980

3981 A binding to a request/response carrier is not required to use these rules. It may define other
3982 rules.

3983

3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029

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.

- 4030 e) A BTP actor that receives a carrier protocol request carrying BTP messages that
4031 do have a reply address, or which initiate processing that produces BTP messages
4032 whose target binding address matches the origin of the request, **may** freely
4033 choose whether to use the carrier protocol response for the replies, or to send
4034 back an “empty carrier protocol response”, and send the BTP replies in a
4035 separately initiated carrier protocol request. The characteristics of an “empty
4036 carrier protocol response” shall be stated in the particular binding specification.
4037
- 4038 f) A BTP actor that sends BTP messages on a carrier protocol request **must** be able
4039 to accept returning BTP messages on the corresponding carrier protocol response
4040 and, if the actor has offered an address on which it will receive carrier requests,
4041 must be able to accept “replying” BTP messages on a separate carrier protocol
4042 request.
4043

4044 SOAP Binding

4045
4046 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
4047 specification, using the SOAP literal messaging style conventions. If no application message
4048 is sent at the same time, the BTP messages are contained within the SOAP Body element. If
4049 application messages are sent, the BTP messages are contained in the SOAP Header element.
4050

4051 **Binding name:** soap-http-1

4052
4053 **Binding address format:** shall be a URL, of type HTTP.
4054

4055 **BTP message representation:** The string representation of the XML, as specified in the
4056 XML schema defined in this document shall be used. The BTP XML messages are embedded
4057 in the SOAP message without the use of any specific encoding rules (literal style SOAP
4058 message); hence the encodingStyle attribute need not be set or can be set to an empty string.
4059

4060 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be
4061 used.
4062

4063 BTP messages sent on an HTTP request or HTTP response which is not carrying an
4064 application message, the messages are contained in a single btp:messages element which is
4065 the immediate child element of the SOAP Body element.
4066

4067 An “empty carrier protocol response” sent after receiving an HTTP request containing a
4068 btp:messages element in the SOAP Body and the implementation BTP actor chooses just to
4069 reply at the lower level (and when the request/response exploitation rules allow an empty
4070 carrier protocol response), shall be any of:

- 4071 a) an empty HTTP response
- 4072 b) an HTTP response containing an empty SOAP Envelope
- 4073 c) an HTTP response containing a SOAP Envelope containing a single, empty
4074 btp:messages element.
4075

4076 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they
4077 have no effect on the BTP sequence (other than indicating that the earlier sending did not
4078 cause a communication failure.)

4079
4080
4081

4082 If an application message is being sent at the same time, the mapping for related messages
4083 shall be used, as if the BTP messages were related to the application message. (There is no
4084 ambiguity in whether the BTP messages are related, because only CONTEXT and ENROL
4085 can be related to an application message.)

4086

4087 **Mapping for BTP messages related to application messages:** All BTP messages sent with
4088 an application message, whether related to the application message or not, shall be sent in a
4089 single btp:messages element in the SOAP Header. There shall be precisely one btp:messages
4090 element in the SOAP Header.

4091

4092 The “request/response exploitation” rules shall apply to the BTP messages carried in the
4093 SOAP Header, as if they had been carried in a SOAP Body, unrelated to an application
4094 message, sent to the same binding address.

4095

Note – The application protocol itself (which is using the SOAP Body) may
4096 use the SOAP RPC or document approach – this is determined by the
4097 application.

4098

4099 Only CONTEXT and ENROL messages are related (&) to application messages. If there is
4100 only one CONTEXT or one ENROL message present in the SOAP Header, it is assumed to
4101 be related to the whole of the application message in the SOAP Body. If there are multiple
4102 CONTEXT or ENROL messages, any relation of these BTP messages shall be indicated by
application specific means.

4103

Note 1 – An application protocol could use references to the ID values of the
4104 BTP messages to indicate relation between BTP CONTEXT or ENROL
4105 messages and the application message.

4106

Note 2 -- However indicated, what the relatedness means, or even whether it
4107 has any significance at all, is a matter for the application.

4108

4109 **Implicit messages:** A SOAP FAULT, or other communication failure received in response to
4110 a SOAP request that had a CONTEXT in the SOAP Header shall be treated as if a
4111 CONTEXT_REPLY/repudiated had been received. See also the discussion under “other”
4112 about the SOAP mustUnderstand attribute.

4113

4114 **Faults:** A SOAP FAULT or other communication failure shall be treated as
4115 FAULT/communication-failure.

4116

4117 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4118 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
4119 http-1” if the binding address and additional information fields match.

4120
4121 **Limitations on BTP use:** None

4122
4123 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
4124 attribute. The SOAPAction HTTP header is left to be application specific when there are
4125 application messages in the SOAP Body, as an already existing web service that is being
4126 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
4127 header shall be omitted when the SOAP message carries only BTP messages in the SOAP
4128 Body.

4129
4130 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP
4131 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to
4132 determine whether any enrolments are necessary and replies with CONTEXT_REPLY as
4133 appropriate. The sender of the CONTEXT (and related application message) can use this to
4134 ensure that the application work is performed as part of the business transaction, assuming the
4135 receiver’s SOAP implementation supports the mustUnderstand attribute. If mustUnderstand if
4136 false, a receiver can ignore the CONTEXT (if BTP is not supported there), and no
4137 CONTEXT_REPLY will be returned. It is a local option on the sender (client) side whether
4138 the absence of a CONTEXT_REPLY is assumed to be equivalent to aCONTEXT_REPLY/ok
4139 (and the business transaction allowed to proceed to confirmation).

4140
4141 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to
4142 enforce these requirements.

4143 **Example scenario using SOAP binding**

4144
4145 The example below shows an application request with CONTEXT message sent from
4146 client.example.com (which includes the Superior) to services.example.com (Service).

```
4147  
4148  
4149 <soap:Envelope  
4150   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
4151   soapencodingStyle=" " >  
4152  
4153   <soap:Header>  
4154  
4155     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">  
4156       <btp:context superior-type="atom">  
4157         <btp:superior-address>  
4158           <btp:binding>soap-http-1</btp:binding>  
4159           <btp:binding-  
4160 address>http://client.example.com/soaphandler</btp:binding-  
4161 address>  
4162           <btp:additional-information>btpengine</btp:additional-  
4163 information>  
4164           </btp:superior-address>  
4165           <btp:superior-identifier>1001</btp:superior-identifier>
```

```

4166         <btp:qualifiers>
4167             <btpq:transaction-timelimit
4168 xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers">1800</btpq:transact
4169 ion-timelimit>
4170         </btp:qualifiers>
4171     </btp:context>
4172 </btp:messages>
4173
4174 </soap:Header>
4175
4176 <soap:Body>
4177
4178     <ns1:orderGoods
4179 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4180         <custID>ABC8329045</custID>
4181         <itemID>224352</itemID>
4182         <quantity>5</quantity>
4183     </ns1:orderGoods>
4184
4185 </soap:Body>
4186
4187 </soap:Envelope>
4188

```

4189
4190 The example below shows CONTEXT_REPLY and a related ENROL message sent from
4191 services.example.com to client.example.com, in reply to the previous message. There is no
4192 application response, so the BTP messages are in the SOAP Body. The ENROL message
4193 does not contain the target-additional-information, since the grouping rules for
4194 CONTEXT_REPLY & ENROL omit the target address (the receiver of this example
4195 remembers the superior address from the original CONTEXT)

```

4196
4197 <soap:Envelope
4198     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
4199     soap:encodingStyle="">
4200
4201 <soap:Header>
4202 </soap:Header>
4203
4204 <soap:Body>
4205
4206     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
4207         <btp:relatedgroup>
4208             <btp:context-reply>
4209                 <btp:superior-address>
4210                     <btp:binding>soap-http-1</btp:binding>
4211                     <btp:binding-address>
4212                         http://client.example.com/soaphandler
4213                     </btp:binding-address>
4214                     <btp:additional-information>
4215                         btpengine
4216                     </btp:additional-information>
4217                 </btp:superior-address>

```

```
4218     <btp:superior-identifier>1001</btp:superior-identifier>
4219     <completion-status>related</completion-status>
4220     </btp:context-reply>
4221
4222     <btp:enrol reply-requested="false">
4223         <btp:superior-identifier>
4224             1001
4225         </btp:superior-identifier>
4226         <btp:inferior-address>
4227             <btp:binding>soap-http-1</btp:binding>
4228             <btp:binding-address>
4229                 http://services.example.com/soaphandler
4230             </btp:binding-address>
4231         </btp:inferior-address>
4232         <btp:inferior-identifier>
4233             AAAB
4234         </btp:inferior-identifier>
4235     </btp:enrol>
4236
4237     </btp:relatedgroup>
4238
4239     </btp:messages>
4240
4241 </soap:Body>
4242
4243 </soap:Envelope>
4244
4245
4246
```

4247 SOAP + Attachments Binding

4248
4249 This binding describes how BTP messages will be carried using SOAP as in the [SOAP](#)
4250 [Messages with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-
4251 http-1. The two bindings only differ when application messages are sent.

4252
4253 **Binding name:** soap-attachments-http-1

4254
4255 **Binding address format:** as for soap-http-1

4256
4257 **BTP message representation:** As for soap-http-1

4258
4259 **Mapping for BTP messages (unrelated):** As for “soap-http-1”, except the SOAP Envelope
4260 containing the SOAP Body containing the BTP messages shall be in a MIME body part, as
4261 specified in [SOAP Messages with Attachments](#) specification. If an application message is
4262 being sent at the same time, the mapping for related messages for this binding shall be used,
4263 as if the BTP messages were related to the application message(s).

4264
4265 **Mapping for BTP messages related to application messages:** MIME packaging shall be
4266 used. One of the MIME multipart/related parts shall contain a SOAP Envelope, whose SOAP

4267 Headers element shall contain precisely one `btm:messages` element, containing any BTP
4268 messages. Any BTP CONTEXT in the `btm:messages` is considered to be related to the
4269 application message(s) in the SOAP Body, and to also any of the MIME parts referenced
4270 from the SOAP Body (using the “href” attribute).

4271

4272 **Implicit messages:** As for `soap-http-1`.

4273

4274 **Faults:** As for `soap-http-1`.

4275

4276 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4277 string “`soap-http-1`” is considered to match one that has the binding string “`soap-`
4278 `attachements-http-1`” if the binding address and additional information fields match.

4279

4280 **Limitations on BTP use:** None

4281

4282 **Other:** As for `soap-http-1`

4283

4284 *Example using SOAP + Attachments binding*

4285

```
4286 MIME-Version: 1.0
```

```
4287 Content-Type: Multipart/Related; boundary=MIME_boundary;
```

```
4288 type=text/xml;
```

```
4289     start="someID"
```

4290

```
4291 --MIME_boundary
```

```
4292 Content-Type: text/xml; charset=UTF-8
```

```
4293 Content-ID: someID
```

4294

```
4295 <?xml version='1.0' ?>
```

```
4296 <soap:Envelope
```

```
4297     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
```

```
4298     soap-
```

```
4299     env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

4300

```
4301     <soap:Header>
```

4302

```
4303         <btm:messages xmlns:btm="urn:oasis:names:tc:BTP:xml">
```

```
4304             <btm:context superior-type="atom">
```

```
4305                 <btm:superior-address>
```

```
4306                     <btm:binding>soap-http-1</btm:binding>
```

```
4307                     <btm:binding-address>
```

```
4308                         http://client.example.com/soaphandler
```

```
4309                     </btm:binding-address>
```

```
4310                     </btm:superior-address>
```

```
4311                     <btm:superior-identifier>1001</btm:superior-identifier>
```

```
4312                 </btm:context>
```

```
4313             </btm:messages>
```

4314

```
4315         </soap:Header>
```

4316

```

4317     <soap:Body>
4318         <orderGoods href="cid:anotherID"/>
4319     </soap:Body>
4320
4321 </soap:Envelope>
4322
4323 --MIME_boundary
4324 Content-Type: text/xml
4325 Content-ID: anotherID
4326
4327     <ns1:orderGoods
4328 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4329         <custID>ABC8329045</custID>
4330         <itemID>224352</itemID>
4331         <quantity>5</quantity>
4332     </ns1:orderGoods>
4333
4334
4335 --MIME_boundary--
4336
4337

```

XML Schema

```

4338
4339
4340 <?xml version="1.0"?>
4341 <schema targetNamespace="urn:oasis:names:tc:BTP:xml"
4342     xmlns="http://www.w3.org/2001/XMLSchema"
4343     xmlns:tns="urn:oasis:names:tc:BTP:xml">
4344
4345     <complexType name="qualifier_type">
4346         <simpleContent>
4347             <extension base="string">
4348                 <attribute name="must-be-understood" type="boolean"/>
4349                 <attribute name="to-be-propagated" type="boolean"/>
4350             </extension>
4351         </simpleContent>
4352     </complexType>
4353     <element name="qualifier" type="tns:qualifier_type"/>
4354     <element name="qualifiers">
4355         <complexType>
4356             <sequence>
4357                 <element ref="tns:qualifier" maxOccurs="unbounded"/>
4358             </sequence>
4359         </complexType>
4360     </element>
4361
4362     <complexType name="address">
4363         <sequence>
4364             <element name="binding-name" type="string"/>
4365             <element name="binding-address" type="string"/>
4366             <element name="additional-information" type="string"
4367 minOccurs="0"/>
4368         </sequence>

```

```

4369     </complexType>
4370
4371     <simpleType name="identifier">
4372         <restriction base="string">
4373             <pattern value="([0-9,A-Z])*"/>
4374         </restriction>
4375     </simpleType>
4376
4377     <simpleType name="superior-type">
4378         <restriction base="string">
4379             <enumeration value="cohesion"/>
4380             <enumeration value="atom"/>
4381         </restriction>
4382     </simpleType>
4383
4384     <simpleType name="transaction-type">
4385         <restriction base="string">
4386             <enumeration value="cohesion"/>
4387             <enumeration value="atom"/>
4388         </restriction>
4389     </simpleType>
4390
4391     <element name="context">
4392         <complexType>
4393             <sequence>
4394                 <element name="superior-address" type="tns:address"
4395 maxOccurs="unbounded"/>
4396                 <element name="superior-identifier" type="tns:identifier"/>
4397                 <element ref="tns:qualifiers" minOccurs="0"/>
4398             </sequence>
4399             <attribute name="id" type="ID" use="optional"/>
4400             <attribute name="superior-type" type="tns:superior-type"
4401 use="required"/>
4402         </complexType>
4403     </element>
4404
4405     <element name="context-reply">
4406         <complexType>
4407             <sequence>
4408                 <element name="superior-address" type="tns:address"
4409 maxOccurs="unbounded"/>
4410                 <element name="superior-identifier" type="tns:identifier"/>
4411                 <element name="completion-status">
4412                     <simpleType>
4413                         <restriction base="string">
4414                             <enumeration value="completed"/>
4415                             <enumeration value="related"/>
4416                             <enumeration value="repudiated"/>
4417                         </restriction>
4418                     </simpleType>
4419                 </element>
4420                 <element ref="tns:qualifiers" minOccurs="0"/>
4421             </sequence>

```

```

4422         </sequence>
4423         <attribute name="id" type="ID"/>
4424         <attribute name="superior-type" type="tns:superior-type"
4425 use="required"/>
4426     </complexType>
4427 </element>
4428
4429     <element name="begin">
4430         <complexType>
4431             <sequence>
4432                 <element name="target-additional-information"
4433 type="string"/>
4434                 <element name="reply-address" type="tns:address"/>
4435                 <element ref="tns:qualifiers" minOccurs="0"/>
4436             </sequence>
4437             <attribute name="id" type="ID"/>
4438             <attribute name="transaction-type" type="tns:superior-type"
4439 use="required"/>
4440         </complexType>
4441     </element>
4442
4443     <element name="begun">
4444         <complexType>
4445             <sequence>
4446                 <element name="target-additional-information"
4447 type="string"/>
4448                 <element name="decider-address" type="tns:address"
4449 minOccurs="0"/>
4450                 <element name="transaction-identifier"
4451 type="tns:identifier" minOccurs="0"/>
4452                 <element name="inferior-handle" type="tns:identifier"
4453 minOccurs="0"/>
4454                 <element name="inferior-address" type="tns:address"
4455 minOccurs="0"/>
4456                 <element ref="tns:qualifiers" minOccurs="0"/>
4457             </sequence>
4458             <attribute name="id" type="ID"/>
4459             <attribute name="transaction-type" type="tns:superior-type"
4460 use="required"/>
4461         </complexType>
4462     </element>
4463
4464     <element name="enrol">
4465         <complexType>
4466             <sequence>
4467                 <element name="target-additional-information"
4468 type="string"/>
4469                 <element name="superior-identifier" type="tns:identifier"/>
4470                 <element name="reply-address" type="tns:address"
4471 minOccurs="0"/>
4472                 <element name="inferior-address" type="tns:address"
4473 minOccurs="1" maxOccurs="unbounded"/>
4474                 <element name="inferior-identifier" type="tns:identifier"/>

```

```

4475         <element ref="tns:qualifiers" minOccurs="0"/>
4476     </sequence>
4477     <attribute name="id" type="ID"/>
4478     <attribute name="reply-requested" type="boolean"/>
4479 </complexType>
4480 </element>
4481
4482
4483     <element name="enrolled">
4484         <complexType>
4485             <sequence>
4486                 <element name="target-additional-information"
4487 type="string"/>
4488                 <element name="inferior-identifier" type="tns:identifier"/>
4489                 <element name="inferior-handle" type="tns:identifier"
4490 minOccurs="0"/>
4491                 <element ref="tns:qualifiers" minOccurs="0"/>
4492             </sequence>
4493             <attribute name="id" type="ID"/>
4494         </complexType>
4495     </element>
4496
4497     <element name="resign">
4498         <complexType>
4499             <sequence>
4500                 <element name="target-additional-information"
4501 type="string"/>
4502                 <element name="superior-identifier" type="tns:identifier"/>
4503                 <element name="inferior-address" type="tns:address"
4504 minOccurs="1" maxOccurs="unbounded"/>
4505                 <element name="inferior-identifier" type="tns:identifier"/>
4506                 <element ref="tns:qualifiers" minOccurs="0"/>
4507             </sequence>
4508             <attribute name="id" type="ID"/>
4509             <attribute name="response-requested" type="boolean"/>
4510         </complexType>
4511     </element>
4512
4513     <element name="resigned">
4514         <complexType>
4515             <sequence>
4516                 <element name="target-additional-information"
4517 type="string"/>
4518                 <element name="inferior-identifier" type="tns:identifier"/>
4519                 <element ref="tns:qualifiers" minOccurs="0"/>
4520             </sequence>
4521             <attribute name="id" type="ID"/>
4522         </complexType>
4523     </element>
4524
4525     <element name="prepare">
4526         <complexType>
4527             <sequence>

```

```

4528         <element name="target-additional-information"
4529 type="string"/>
4530         <element name="inferior-identifier" type="tns:identifier"
4531 minOccurs="0"/>
4532         <element name="reply-address" type="tns:address"
4533 minOccurs="0"/>
4534         <element name="transaction-identifier"
4535 type="tns:identifier" minOccurs="0"/>
4536         <element name="inferiors-list" minOccurs="0">
4537             <complexType>
4538                 <sequence>
4539                     <element name="inferior-handle"
4540 type="tns:identifier" maxOccurs="unbounded"/>
4541                 </sequence>
4542             </complexType>
4543         </element>
4544         <element ref="tns:qualifiers" minOccurs="0"/>
4545     </sequence>
4546     <attribute name="id" type="ID"/>
4547 </complexType>
4548 </element>
4549
4550     <element name="prepared">
4551         <complexType>
4552             <sequence>
4553                 <element name="target-additional-information"
4554 type="string"/>
4555                 <element name="superior-identifier" type="tns:identifier"/>
4556                 <element name="inferior-address" type="tns:address"
4557 maxOccurs="unbounded"/>
4558                 <element name="inferior-identifier" type="tns:identifier"/>
4559                 <element ref="tns:qualifiers" minOccurs="0"/>
4560             </sequence>
4561             <attribute name="id" type="ID"/>
4562             <attribute name="default-is-cancel" type="boolean"/>
4563         </complexType>
4564     </element>
4565
4566     <element name="confirm">
4567         <complexType>
4568             <sequence>
4569                 <element name="target-additional-information"
4570 type="string"/>
4571                 <element name="inferior-identifier" type="tns:identifier"/>
4572                 <element ref="tns:qualifiers" minOccurs="0"/>
4573             </sequence>
4574             <attribute name="id" type="ID"/>
4575         </complexType>
4576     </element>
4577
4578     <element name="confirmed">
4579         <complexType>
4580             <sequence>

```

```

4581         <element name="target-additional-information"
4582 type="string"/>
4583         <element name="superior-identifier" type="tns:identifier"/>
4584         <element name="inferior-address" type="tns:address"
4585 minOccurs="0"/>
4586         <element name="inferior-identifier" type="tns:identifier"
4587 minOccurs="0"/>
4588         <element name="decider-address" type="tns:address"
4589 minOccurs="0"/>
4590         <element name="transaction-identifier"
4591 type="tns:identifier" minOccurs="0"/>
4592         <element ref="tns:qualifiers" minOccurs="0"/>
4593     </sequence>
4594     <attribute name="id" type="ID"/>
4595     <attribute name="confirmed-received" type="boolean"/>
4596 </complexType>
4597 </element>
4598
4599     <element name="cancel">
4600         <complexType>
4601             <sequence>
4602                 <element name="target-additional-information"
4603 type="string"/>
4604                 <element name="inferior-identifier" type="tns:identifier"
4605 minOccurs="0"/>
4606                 <element name="reply-address" type="tns:address"
4607 minOccurs="0"/>
4608                 <element name="transaction-identifier"
4609 type="tns:identifier" minOccurs="0"/>
4610                 <element name="decider-address" type="tns:address"
4611 minOccurs="0"/>
4612                 <element name="transaction-identifier"
4613 type="tns:identifier" minOccurs="0"/>
4614                 <element name="inferiors-list" minOccurs="0">
4615                     <complexType>
4616                         <sequence>
4617                             <element name="inferior-handle"
4618 type="tns:identifier" maxOccurs="unbounded"/>
4619                         </sequence>
4620                     </complexType>
4621                 </element>
4622                 <element ref="tns:qualifiers" minOccurs="0"/>
4623             </sequence>
4624             <attribute name="id" type="ID"/>
4625         </complexType>
4626     </element>
4627
4628     <element name="cancelled">
4629         <complexType>
4630             <sequence>
4631                 <element name="target-additional-information"
4632 type="string"/>
4633                 <element name="superior-identifier" type="tns:identifier"/>

```

```

4634         <element name="inferior-address" type="tns:address"
4635 maxOccurs="unbounded"/>
4636         <element name="inferior-identifier" type="tns:identifier"
4637 minOccurs="0"/>
4638         <element name="decider-address" type="tns:address"
4639 minOccurs="0"/>
4640         <element name="transaction-identifier"
4641 type="tns:identifier" minOccurs="0"/>
4642         <element ref="tns:qualifiers" minOccurs="0"/>
4643     </sequence>
4644     <attribute name="id" type="ID"/>
4645 </complexType>
4646 </element>
4647
4648     <element name="hazard">
4649         <complexType>
4650             <sequence>
4651                 <element name="target-additional-information"
4652 type="string"/>
4653                 <element name="superior-identifier" type="tns:identifier"/>
4654                 <element name="inferior-address" type="tns:address"
4655 maxOccurs="unbounded"/>
4656                 <element name="inferior-identifier" type="tns:identifier"/>
4657                 <element ref="tns:qualifiers" minOccurs="0"/>
4658             </sequence>
4659             <attribute name="id" type="ID"/>
4660         </complexType>
4661     </element>
4662
4663     <element name="contradiction">
4664         <complexType>
4665             <sequence>
4666                 <element name="target-additional-information"
4667 type="string"/>
4668                 <element name="inferior-identifier" type="tns:identifier"/>
4669                 <element ref="tns:qualifiers" minOccurs="0"/>
4670             </sequence>
4671             <attribute name="id" type="ID"/>
4672         </complexType>
4673     </element>
4674
4675     <element name="superior-state">
4676         <complexType>
4677             <sequence>
4678                 <element name="target-additional-information"
4679 type="string"/>
4680                 <element name="inferior-identifier" type="tns:identifier"/>
4681                 <element name="status">
4682                     <simpleType>
4683                         <restriction base="string">
4684                             <enumeration value="active"/>
4685                             <enumeration value="prepared-received"/>
4686                             <enumeration value="inaccessible"/>

```

```

4687         <enumeration value="unknown"/>
4688     </restriction>
4689 </simpleType>
4690 </element>
4691     <element ref="tns:qualifiers" minOccurs="0"/>
4692 </sequence>
4693 <attribute name="id" type="ID"/>
4694 <attribute name="reply-requested" type="boolean"/>
4695 </complexType>
4696 </element>
4697
4698 <element name="inferior-state">
4699     <complexType>
4700         <sequence>
4701             <element name="target-additional-information"
4702 type="string"/>
4703             <element name="superior-identifier" type="tns:identifier"/>
4704             <element name="inferior-address" type="tns:address"
4705 maxOccurs="unbounded"/>
4706             <element name="inferior-identifier" type="tns:identifier"/>
4707             <element name="status">
4708                 <simpleType>
4709                     <restriction base="string">
4710                         <enumeration value="active"/>
4711                         <enumeration value="prepared-received"/>
4712                         <enumeration value="inaccessible"/>
4713                         <enumeration value="unknown"/>
4714                     </restriction>
4715                 </simpleType>
4716             </element>
4717             <element ref="tns:qualifiers" minOccurs="0"/>
4718         </sequence>
4719         <attribute name="id" type="ID"/>
4720         <attribute name="reply-requested" type="boolean"/>
4721     </complexType>
4722 </element>
4723
4724 <element name="confirm-one-phase">
4725     <complexType>
4726         <sequence>
4727             <element name="target-additional-information"
4728 type="string"/>
4729             <element name="inferior-identifier" type="tns:identifier"/>
4730             <element ref="tns:qualifiers" minOccurs="0"/>
4731         </sequence>
4732         <attribute name="id" type="ID"/>
4733         <attribute name="report-hazard" type="boolean"/>
4734     </complexType>
4735 </element>
4736
4737 <element name="request-confirm">
4738     <complexType>
4739         <sequence>

```

```

4740         <element name="target-additional-information"
4741 type="string"/>
4742         <element name="reply-address" type="tns:address"/>
4743         <element name="transaction-identifier"
4744 type="tns:identifier"/>
4745         <element name="inferiors-list" minOccurs="0">
4746             <complexType>
4747                 <sequence>
4748                     <element name="inferior-handle"
4749 type="tns:identifier" maxOccurs="unbounded"/>
4750                 </sequence>
4751             </complexType>
4752         </element>
4753         <element ref="tns:qualifiers" minOccurs="0"/>
4754     </sequence>
4755     <attribute name="id" type="ID"/>
4756     <attribute name="report-hazard" type="boolean"/>
4757 </complexType>
4758 </element>
4759
4760 <element name="request-statuses">
4761     <complexType>
4762         <sequence>
4763             <element name="target-additional-information"
4764 type="string"/>
4765             <element name="reply-address" type="tns:address"/>
4766             <element name="transaction-identifier"
4767 type="tns:identifier"/>
4768             <element name="inferiors-list" minOccurs="0">
4769                 <complexType>
4770                     <sequence>
4771                         <element name="inferior-handle"
4772 type="tns:identifier" maxOccurs="unbounded"/>
4773                     </sequence>
4774                 </complexType>
4775             </element>
4776             <element ref="tns:qualifiers" minOccurs="0"/>
4777         </sequence>
4778         <attribute name="id" type="ID"/>
4779     </complexType>
4780 </element>
4781
4782 <element name="inferior-statuses">
4783     <complexType>
4784         <sequence>
4785             <element name="target-additional-information"
4786 type="string"/>
4787             <element name="decider-address" type="tns:address"/>
4788             <element name="transaction-identifier"
4789 type="tns:identifier"/>
4790             <element name="status-list">
4791                 <complexType>
4792                     <sequence>

```

```

4793         <element name="status-item" maxOccurs="unbounded">
4794             <complexType>
4795                 <sequence>
4796                     <element name="inferior-handle"
4797 type="tns:identifier"/>
4798                     <element name="status">
4799                         <simpleType>
4800                             <restriction base="string">
4801                                 <enumeration value="active"/>
4802                                 <enumeration value="resigned"/>
4803                                 <enumeration value="preparing"/>
4804                                 <enumeration value="prepared"/>
4805                                 <enumeration value="autonomously-confirmed"/>
4806                                 <enumeration value="autonomously-cancelled"/>
4807                                 <enumeration value="confirming"/>
4808                                 <enumeration value="confirmed"/>
4809                                 <enumeration value="cancelling"/>
4810                                 <enumeration value="cancelled"/>
4811                                 <enumeration value="cancel-contradiction"/>
4812                                 <enumeration value="confirm-contradiction"/>
4813                                 <enumeration value="hazard"/>
4814                             </restriction>
4815                         </simpleType>
4816                     </element>
4817                     <element ref="tns:qualifiers" minOccurs="0"/>
4818                 </sequence>
4819             </complexType>
4820         </element>
4821     </sequence>
4822 </complexType>
4823 </element>
4824     <element ref="tns:qualifiers" minOccurs="0"/>
4825 </sequence>
4826     <attribute name="id" type="ID"/>
4827 </complexType>
4828 </element>
4829
4830 <element name="request-status">
4831     <complexType>
4832         <sequence>
4833             <element name="target-additional-information"
4834 type="string"/>
4835             <element name="reply-address" type="tns:address"/>
4836             <element name="inferior-identifier" type="tns:identifier"
4837 minOccurs="0"/>
4838             <element name="transaction-identifier"
4839 type="tns:identifier" minOccurs="0"/>
4840             <element ref="tns:qualifiers" minOccurs="0"/>
4841         </sequence>
4842         <attribute name="id" type="ID"/>
4843     </complexType>
4844 </element>
4845

```

```

4846     <element name="status">
4847         <complexType>
4848             <sequence>
4849                 <element name="target-additional-information"
4850 type="string"/>
4851                 <element name="inferior-address" type="tns:address"
4852 minOccurs="0"/>
4853                 <element name="inferior-identifier" type="tns:identifier"
4854 minOccurs="0"/>
4855                 <element name="decider-address" type="tns:address"
4856 minOccurs="0"/>
4857                 <element name="transaction-identifier"
4858 type="tns:identifier" minOccurs="0"/>
4859                 <element name="status-value">
4860                     <simpleType>
4861                         <restriction base="string">
4862                             <enumeration value="created"/>
4863                             <enumeration value="enrolling"/>
4864                             <enumeration value="active"/>
4865                             <enumeration value="resigning"/>
4866                             <enumeration value="resigned"/>
4867                             <enumeration value="preparing"/>
4868                             <enumeration value="prepared"/>
4869                             <enumeration value="confirming"/>
4870                             <enumeration value="confirmed"/>
4871                             <enumeration value="cancelling"/>
4872                             <enumeration value="cancelled"/>
4873                             <enumeration value="cancel-contradiction"/>
4874                             <enumeration value="confirm-contradiction"/>
4875                             <enumeration value="hazard"/>
4876                             <enumeration value="contradicted"/>
4877                             <enumeration value="unknown"/>
4878                             <enumeration value="inaccessible"/>
4879                         </restriction>
4880                     </simpleType>
4881                 </element>
4882                 <element ref="tns:qualifiers" minOccurs="0"/>
4883             </sequence>
4884             <attribute name="id" type="ID"/>
4885         </complexType>
4886     </element>
4887
4888     <element name="redirect">
4889         <complexType>
4890             <sequence>
4891                 <element name="target-additional-information"
4892 type="string"/>
4893                 <element name="superior-identifier" type="tns:identifier"
4894 minOccurs="0"/>
4895                 <element name="inferior-identifier" type="tns:identifier"/>
4896                 <element name="old-address" type="tns:address"
4897 maxOccurs="unbounded"/>

```

```

4898         <element name="new-address" type="tns:address"
4899 maxOccurs="unbounded"/>
4900         <element ref="tns:qualifiers" minOccurs="0"/>
4901     </sequence>
4902     <attribute name="id" type="ID"/>
4903 </complexType>
4904 </element>
4905
4906 <element name="fault">
4907     <complexType>
4908         <sequence>
4909             <element name="target-additional-information"
4910 type="string"/>
4911             <element name="superior-identifier" type="tns:identifier"
4912 minOccurs="0"/>
4913             <element name="inferior-identifier" type="tns:identifier"
4914 minOccurs="0"/>
4915             <element name="fault-type" type="string"/>
4916             <element name="fault-data" type="anyType" minOccurs="0"/>
4917             <element ref="tns:qualifiers" minOccurs="0"/>
4918         </sequence>
4919         <attribute name="id" type="ID"/>
4920     </complexType>
4921 </element>
4922
4923 </schema>
4924

```

4924

4925

4926 **Conformance**

4927

4928

4929

4930

4931

4932

4933

4934

4935

4936

4937

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

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

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

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

4938
4939
4940
4941
4942

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

4943
4944
4945
4946
4947
4948
4949
4950
4951
4952

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)

4952
4953
4954
4955
4956
4957

Part 3. Appendices

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

A. Glossary

Message	A datum which is produced and then consumed.
Sender	The producer of a message.
Receiver	The consumer of a message.
Transmission	The passage of a message from a sender to a receiver.
Endpoint	A sender or receiver.
Address	An identifier for an endpoint.
Carrier Protocol	A protocol which defines how transmissions occur.
Carrier Protocol Address (CPA)	The address of an endpoint for a particular carrier protocol.
Business Transaction Protocol Address (BTPA)	A compound address consisting of a mandatory <i>carrier protocol address</i> and an optional opaque suffix. <div data-bbox="784 1272 1313 1352" style="border: 1px solid black; padding: 2px;"><i>PRF - suffix ? I've used "additional information"</i></div>
Actor	An entity which executes procedures, a software agent.
Application	An actor which uses the Business Transaction Protocol.
Application Message	A message produced by an application and consumed by an application.
Application Endpoint	An endpoint of an application message.

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

Countereffect Contract	<p>The contract which governs the relationship between the effect and the countereffect of a procedure. In the absence of any other overriding contracts the countereffect contract is the promise that</p> <p>“The Countereffect will attempt so far as is possible to reverse or cancel the Effect such that an observer (on completion of the Countereffect) is unaware that the Effect ever occurred, but this attempt cannot be guaranteed to succeed”.</p>
Cancel	Process a countereffect for the current effect of a set of procedures.
Confirm	Ensure that the effect of a set of procedures is completed.
Prepare	Ensure that of a set of procedures is capable of being successfully instructed to cancel or to confirm.
Outcome	A decision to either cancel or confirm.
Participant	A set of procedures which is capable of receiving instructions from a coordinator to prepare, cancel and confirm. A participant must also have a BTPA to which these instructions will be delivered, in the form of BTP messages. A participant is identified by a participant identifier.
Inferior 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.

4958