

Business Transaction Protocol

An OASIS Committee Specification

CURRENT STATUS : internal committee draft

Version 1.0 [0.9.2.2]

DD Mmm 2002 [12 March 2002 18:00]

<i>Working draft 0.1 (pre-London)</i>	14 June 2001
<i>Working draft 0.2 (London)</i>	18 June 2001
<i>Working draft 0.3a (circulated)</i>	12 July 2001
<i>Working draft 0.3c (circulated)</i>	20 July 2001
<i>Working draft 0.4 (circulated; incorporates PRF material)</i>	25 July 2001
<i>Working draft 0.6 (State tables)</i>	31 August 2001
<i>Working Draft 0.9</i>	24 October 2001
<i>Working Draft 0.9.0.1 – minor editorials issues applied</i>	16 November 2001
<i>Working Draft 0.9.0.2 – issue resolutions balloting to 10 Dec 2001</i>	4 December 2001
<i>Working Draft 0.9.0.3 – possible solution to msging issues</i>	11 December 2001
<i>Working Draft 0.9.0.4 – issue 79 solution, revise msging issues</i>	12 January 2002
<i>Working Draft 0.9.1 – includes all issues agreed 16 Jan 2002, and 82 (deferred)</i>	18 January 2002
<i>Working Draft 0.9.1.1 – format changes and proposed soln 77,78, 17.</i>	27 January 2002
<i>Working Draft 0.9.1.2 – xml changes, new schema, and issue 74</i>	30 January 2002
<i>Working Draft 0.9.1.3 – corrections, issue 30, state table – 81, 104</i>	8 February 2002
<i>Working Draft 0.9.2 – all issues as agreed 13 February 2002</i>	13 February 2002
<i>Working Draft 0.9.2.1 – issues 2, 3, 15, 19, 50, 67, 95</i>	26 February 2002
<i>Working Draft 0.9.2.2 – as accepted 27 Feb 2002+ corrections, issues 29, 60, 97, 99</i>	12 March 2002

Change marks relative to 0.9.2.1 with changes accepted

13 Copyright and related notices

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

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

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

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

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

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

53

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

Acknowledgements

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

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

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

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

Additional contributions to its writing were made by

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

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

In memory of Ed Felt

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

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

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

98 **Typographical and Linguistic Conventions and Style**

99

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

103

104

Cancel

105

Participant

106

Application Message

107

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

109

110 **Coordinator**

111

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

114

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

116

117

BEGIN

118

CONTEXT

119

RESIGN

120

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

122

123

BEGIN/atom

124

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

126

127

BEGIN/atom & CONTEXT

128

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

130

131

ENROL + VOTE

132

133 XML schemata and instances are given in Courier:

134

135

```
<btpe:begin> ... </btpe:begin>
```

136

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

138

139

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

140

```
{
```

141

```
}
```

142

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

146
147
148
149
150

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

150	Contents	
151		
152	Copyright and related notices.....	2
153	Acknowledgements	3
154	Typographical and Linguistic Conventions and Style	4
155	Contents	6
156	Part 1. Purpose and Features of BTP	10
157	Introduction.....	10
158	Development and Maintenance of the Specification.....	11
159	Overview of the Business Transaction Protocol	12
160	Part 2. Normative Specification of BTP	15
161	Actors, Roles and Relationships	15
162	Relationships.....	15
163	Roles involved in the outcome relationships	17
164	Superior.....	17
165	Inferior	18
166	Enroller	19
167	Participant	20
168	Sub-coordinator.....	20
169	Sub-composer	21
170	Roles involved in the control relationships.....	21
171	Decider.....	21
172	Coordinator	22
173	Composer	22
174	Terminator.....	22
175	Initiator.....	23
176	Factory	24
177	Other roles	24
178	Redirector.....	24
179	Status Requestor.....	25
180	Abstract Messages and Associated Contracts	25
181	Addresses.....	26
182	Request/response pairs.....	27
183	Compounding messages	27
184	Extensibility.....	29
185	Messages.....	29
186	Qualifiers	30
187	Messages not restricted to outcome or control relationships.	30
188	CONTEXT.....	31
189	CONTEXT_REPLY	32
190	REQUEST_STATUS	33
191	STATUS	34
192	FAULT.....	35
193	REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES	38
194	Messages used in the outcome relationships	38
195	ENROL	38

196	ENROLLED	39
197	RESIGN	40
198	RESIGNED	41
199	PREPARE	42
200	PREPARED	42
201	CONFIRM	44
202	CONFIRMED	44
203	CANCEL	45
204	CANCELLED	46
205	CONFIRM_ONE_PHASE	47
206	HAZARD	48
207	CONTRADICTION	49
208	SUPERIOR_STATE	50
209	INFERIOR_STATE	51
210	REDIRECT	53
211	Messages used in control relationships	54
212	BEGIN	54
213	BEGUN	55
214	PREPARE_INFERIORS	56
215	CONFIRM_TRANSACTION	58
216	TRANSACTION_CONFIRMED	60
217	CANCEL_TRANSACTION	61
218	CANCEL_INFERIORS	62
219	TRANSACTION_CANCELLED	63
220	REQUEST_INFERIOR_STATUSES	63
221	INFERIOR_STATUSES	65
222	Groups – combinations of related messages	67
223	CONTEXT & application message	67
224	CONTEXT_REPLY & ENROL	68
225	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED	69
226	CONTEXT_REPLY & ENROL & application message (& PREPARED)	69
227	BEGUN & CONTEXT	70
228	BEGIN & CONTEXT	70
229	Standard qualifiers	70
230	Transaction timelimit	70
231	Inferior timeout	71
232	Minimum inferior timeout	72
233	Inferior name	73
234	State Tables	74
235	Explanation of the state tables	74
236	Status queries	74
237	Decision events	74
238	Disruptions – failure events	75
239	Invalid cells and assumptions of the communication mechanism	75
240	Meaning of state table events	76
241	Persistent information	80
242	Failure Recovery	93

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

290	XML Schemas	111
291	XML schema for BTP messages.....	111
292	XML schema for standard qualifiers	124
293	Carrier Protocol Bindings	126
294	Carrier Protocol Binding Proforma.....	126
295	Bindings for request/response carrier protocols	127
296	Request/response exploitation rules.....	128
297	SOAP Binding	129
298	Example scenario using SOAP binding	131
299	SOAP + Attachments Binding.....	133
300	Conformance	135
301	Part 3. Appendices.....	138
302	A. Glossary.....	138
303		
304		

Part 1. Purpose and Features of BTP

Introduction

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

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

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

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

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

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

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

347
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
385

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

385 Overview of the Business Transaction Protocol

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

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

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

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

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

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

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

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

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

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

434

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

446

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

455

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

467

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

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

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

Part 2. Normative Specification of BTP

Actors, Roles and Relationships

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

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

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

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

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

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

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

Relationships

There are two primary relationships in BTP.

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

535

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

538

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

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

561

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

567

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

571

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

575

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

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

590 **Roles involved in the outcome relationships**

591

592 **Superior**

593

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

604

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

607

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

612

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

615

616 A Superior receives

617

618 ENROL

619

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

621

622 A Superior sends

623

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

626
627 A Superior sends

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

635
636 to an enrolled Inferior.

637
638 A Superior receives

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

646
647 from an enrolled Inferior.

648
649 **Inferior**

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

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

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

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

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

673

674 An Inferior receives

675

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

682

683 from its Superior.

684

685 An Inferior sends

686

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

693

694 to its Superior.

695

696

697 **Enroller**

698

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

704

705 An Enroller sends

706

707 ENROL

708

709 to a Superior.

710

711 An Enroller receives

712

713 ENROLLED

714

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

716

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

723 Participant

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

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

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

739 Note – Possible approaches are:

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

752 Sub-coordinator

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

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

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

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

768 **Sub-composer**

769
770 An Inferior which is also a Cohesive Superior.

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

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

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

782
783

784 **Roles involved in the control relationships**

785

786 **Decider**

787

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

794

795 A Decider is instructed to cancel by receiving CANCEL_TRANSACTION.

796

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

800

801 All Deciders receive

802 CONFIRM_TRANSACTION

803 CANCEL_TRANSACTION

804 REQUEST_INFERIOR_STATUSES

805

806 All Deciders send
807 TRANSACTION CONFIRMED ~~COMPLETE~~
808 TRANSACTION CANCELLED ~~COMPLETE~~
809 INFERIOR_STATUSES

810
811

812 Coordinator

813

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

816

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

818

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

823

824 Composer

825

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

828

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

831

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

836

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

841

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

844

845

846 Terminator

847

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

850

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

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

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

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

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

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

881
882 A Terminator receives
883 ~~TRANSACTION_CONFIRMED_COMPLETE~~
884 ~~TRANSACTION_CANCELLED_COMPLETE~~
885 INFERIOR_STATUSES

886 887 Initiator

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

892
893 An Initiator sends

894
895 BEGIN
896 BEGIN & CONTEXT

897
898 to a Factory, and receives in reply

899

900 BEGUN & CONTEXT

901

902 **Factory**

903

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

906

907 Decider, which is either

908 Composer or

909 Coordinator

910 Sub-composer

911 Sub-coordinator

912

913

A Factory receives

914

915 BEGIN

916 BEGIN & CONTEXT

917

918

and replies with

919

920 BEGUN & CONTEXT

921

922

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

925

926

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

929

930

931

932

Other roles

933

934

Redirector

935

936

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

939

940

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

942

943

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

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

949
950 ~~A Redirector may also be used to change the address of other BTP actors.~~
951

952 After receiving a REDIRECT message, the BTP actor **must** use the new address not the old
953 one, unless failure prevents it updating its information.
954

955 **Status Requestor**

956
957 Requests and receives the current status of a transaction tree node – any of an Inferior,
958 Superior or Decider, or the current status of the nodes relationships with its Inferiors, if any.
959 The role of Status Requestor has no responsibilities – it is just a name for where the
960 REQUEST_STATUS and REQUEST_INFERIOR_STATUSES comes from
961 (REQUEST_INFERIOR_STATUSES is also issued by a Terminator to a Decider).
962

963 A Status Requestor sends

964
965 REQUEST_STATUS
966 REQUEST_INFERIOR_STATUSES
967

968 and receives

969
970 STATUS
971 INFERIOR_STATUSES
972

973 in response.

974
975 The receiver of the request can refuse to provide the status information by replying with
976 FAULT(StatusRefused). The information returned in STATUS will always relate to the
977 transaction tree node as a whole (e.g. as an Inferior, even if it is also a Superior).
978

979 **Abstract Messages and Associated Contracts**

980
981 BT Protocol Messages are defined in this section in terms of the abstract information that has
982 to be communicated. These abstract messages will be mapped to concrete messages
983 communicated by a particular carrier protocol (there can be several such mappings defined).
984

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

- 986
- 987 ❑ deliver messages completely and correctly, or not at all (corrupted messages will
988 not be delivered);
 - 989
 - 990 ❑ report some communication failures, but will not necessarily report all (i.e. not all
991 message deliveries are positively acknowledged within the carrier);
 - 992
 - 993 ❑ sometimes deliver successive messages in a different order than they were sent;

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

and

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

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

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

Addresses

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

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

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

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

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

1051
1052 Which of these choices is used is opaque to the entity sending the message – both parts of the
1053 address and the identifier originated at the recipient of this message (and were transmitted as
1054 parameters of earlier messages in the opposite direction).
1055

1056 BTP recovery requires that the state information for a Superior or Inferior is accessible after
1057 failure and that the peer can distinguish between temporary inaccessibility and the permanent
1058 non-existence of the state information. As is explained in “Redirection” below, BTP provides
1059 mechanisms – having a set of BTP addresses for some parameters, and the REDIRECT
1060 message – that make this possible, even if the recovered state information is on a different
1061 address to the original one (as may be the case if case c) above is used).
1062
1063

1064 **Request/response pairs**

1065
1066 Many of the messages combine in pairs as a request and its response. However, in some cases
1067 the response message is sent without a triggering request, or as a possible response to more
1068 than one type of request. To allow for this, the abstract message set treats each message as
1069 standalone; but where a request does expect a reply, a “reply-address” parameter will be
1070 present. For any message with a reply address parameter, in the case of certain errors, a
1071 FAULT message will be sent to the reply address instead of the expected reply.
1072

1073 For messages which are specified as sent between Superior and Inferior, a FAULT message is
1074 sent to the peer.
1075

1076 **Compounding messages**

1077
1078 BTP messages may be sent in combination with each other, or with other (application)
1079 messages. There are two cases:
1080
1081 a) Sending the messages together where the combination has semantic
1082 significance. One message is said to be “related to” the other – the combination
1083 is termed a “group”.
1084 b) Sending of the messages where the combination has no semantic significance,
1085 but is merely a convenience or optimisation. This is termed “bundling” – the
1086 combination is termed a “bundle”.
1087

1088 The form A&B is used to refer to a combination (group) where message B is sent in relation
1089 to A (“relation” is asymmetric). The form A+B is used to refer to A and B bundled together-
1090 the transmission of the bundle "A+B" is semantically identical to the transmission of A
1091 followed by the transmission of B.

1092
1093 Only certain combinations of messages are possible in a group, and the meaning of the
1094 relation is specifically defined for each such combination in the next section. A particular
1095 group is treated as a unit for transmission – it has a single target address. This is usually that
1096 of one of the messages in the group – the specification for the group defines which.

1097
1098 A “bundle” of messages may contain both unrelated messages and groups of related
1099 messages. The only constraint on which messages and groups can be bundled is that all have
1100 the same binding address, but may have different “additional information” values. (Messages
1101 within a related group may have different addresses, where the rules of their relatedness
1102 permit this). Unless constrained by the binding, any messages or groups that are to be sent to
1103 the same binding address may be bundled – the fact that the binding addresses are the same is
1104 a necessary and sufficient condition for the sender to determine that the messages can be
1105 bundled.

1106
1107 A particular and important case of related messages is where a BTP CONTEXT message is
1108 sent related to an application message. In this case, the target of the application message
1109 defines the destination of the CONTEXT message. The receiving implementation may in fact
1110 remove the CONTEXT before delivering the application message to the application (Service)
1111 proper, but from the perspective of the sender, the two are sent to the same place.
1112 The compounding mechanisms, and the multi-part address structures, support the “one-wire”
1113 and “one-shot” communication patterns.

1114
1115 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,
1116 including the associated application messages, pass via the same “endpoints”. These
1117 “endpoints” may in fact be relays, routing messages on to particular actors within their
1118 domain. The onward routing will require some further addressing, but this has to be opaque to
1119 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors
1120 in its domain have the relay’s address as their binding address, and any routing information it
1121 will need in its own domain is placed in the additional information. (This may involve the
1122 relay changing addresses in messages as they pass through it on the way out). On receiving a
1123 message, it determines the within-domain destination from the received additional
1124 information (which is thus rewritten) and forwards the message appropriately. The sender is
1125 unaware of this, and merely sees addresses with the same binding address, which it is
1126 permitted to bundle. The content of the “additional information” is a matter only for the relay
1127 – it could put an entire BTP address in there, or other implementation-defined information.
1128 Note that a quite different one-wire implementation can be constructed where there is no
1129 relaying, but the receiving entity effectively performs all roles, using the received identifiers
1130 to locate the appropriate state.

1131
1132 “One-shot” communication makes it possible to send an application message, receive the
1133 application reply, enrol an Inferior to be responsible for the confirm/cancel of the operations
1134 of those message and inform the Superior that the Inferior is prepared, all in one two-way

1135 exchange across the network (e.g. one request/reply of a carrier protocol).. The application
1136 request is sent with a related CONTEXT message. The application response is sent with a
1137 relation group of CONTEXT_REPLY/related, ENROL/no-rsp-req message and a
1138 PREPARED message. This is possible even if the Superior address is different from the
1139 address of the application element that sends the original message (if the application
1140 exchange is request/reply, there may not even be an identifiable address for the application
1141 element). The target addresses of the ENROL and PREPARED (the Superior address) are not
1142 transmitted; the actor that was originally responsible for adding the CONTEXT to the
1143 outbound application message remembers the Superior address and forwards the ENROL and
1144 PREPARED appropriately.

1145
1146 With “one-shot”, if there are multiple Inferiors created as a result of a single application
1147 message, there is an ENROL and PREPARED message for each sent related to the
1148 CONTEXT_REPLY. If an operation fails, a CANCELLED message is sent instead of a
1149 PREPARED.

1150
1151 If the CONTEXT has “superior-type” of “atom”, then subsequent messages to the same
1152 Service, with the same related CONTEXT/atom, can have their associated operations put
1153 under the control of the same Inferior, and only a CONTEXT_REPLY/completed is sent back
1154 with the response (if the new operations fail, it will be necessary to send back
1155 CONTEXT_REPLY/repudiated, or send CANCELLED). If the “superior type” on the
1156 CONTEXT is “cohesive”, each operation will require separate enrolment.

1157
1158 Whether the “one-shot” mechanism is used is determined by the implementation on the
1159 responding (Inferior) side. This may be subject to configuration and may also be constrained
1160 by the application or by the binding in use.

1161

1162 **Extensibility**

1163

1164 To simplify interoperation between implementations of this edition of BTP with
1165 implementations of future editions, the “must-be-understood” sub-parameter as specified for
1166 Qualifiers may be defined for use with any parameter added to an existing message in a future
1167 revision of this specification. The default for “must-be-understood” shall be “true”, so an
1168 implementation receiving an unrecognised parameter without a “false” value for “must-be-
1169 understood” shall not accept it (the FAULT value “UnrecognisedParameter” is available, but
1170 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If
1171 “must-be-understood” with the value “false” is present as a sub-parameter of a parameter in
1172 any message, a receiving implementation **should** ignore the parameter.

1173

1174 How the sub-parameter is associated with the new parameter is determined by the particular
1175 binding.

1176

1177 No special mechanism is provided to allow for the introduction of completely new messages.

1178

1179 **Messages**

1180

1181
1182
1183
1184
1185

Qualifiers

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

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

1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214

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

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

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

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

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

Messages not restricted to outcome or control relationships.

1215
1216
1217
1218

The messages in this section are used between various roles. CONTEXT message is used in the Initiator:Factory relationship (when it is related to BEGIN or to BEGUN), and related to

1219 an application ‘message’ to propagate the business transaction between parts of the
1220 application. CONTEXT_REPLY is used as the reply to a CONTEXT.REQUEST_STATUS
1221 can be issued to, and STATUS returned by any of Decider, Superior or Inferior. FAULT can
1222 be used on any relationship to indicate an error condition back to the sender of a message.

1223 CONTEXT

1224
1225
1226 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more
1227 application messages. (The means by which this relationship is represented is determined by
1228 the binding and the binding mechanisms of the application protocol.) The “superior-type”
1229 parameter identifies whether the Superior will apply the same decision to all Inferiors
1230 enrolled using the same superior identifier (“superior-type” is “atom”) or whether it may
1231 apply different decisions (“superior-type” is “cohesion”).
1232

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

1233
1234
1235 ~~superior-address-as-superior~~ the address to which ENROL and other
1236 messages from an enrolled Inferior are to be sent. This can be a set of alternative
1237 addresses.

1238
1239 **superior-identifier** identifies the Superior. This shall be globally unambiguous.

1240
1241 **reply-address** the address to which a replying CONTEXT_REPLY is to be sent.
1242 This may be different each time the CONTEXT is transmitted – it refers to the
1243 destination of a replying CONTEXT_REPLY for this particular transmission of
1244 the CONTEXT.

1245
1246 **superior-type** identifies whether the CONTEXT refers to a Cohesion or an
1247 Atom. Default is atom.

1248
1249 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
1250 timelimit” is carried by CONTEXT.

1251
1252 There is no “target-address” parameter for CONTEXT as it is only transmitted in relation to
1253 the application messages, BEGIN and BEGUN.

1254
1255 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the
1256 “superior-type” with the appropriate value.

1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268

CONTEXT_REPLY

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

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

1269
1270
1271
1272
1273
1274
1275
1276
1277

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

superior-identifier the “superior-identifier” from the CONTEXT

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

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.

1278
1279
1280
1281
1282
1283
1284
1285

qualifiers standardised or other qualifiers.

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

1286 If there are no necessary enrolments (e.g. the application messages related to the received
1287 CONTEXT did not require the enrolment of any Inferiors), then
1288 CONTEXT_REPLY/completed is used.

1289
1290 If a CONTEXT_REPLY/repudiated is received, the receiving implementation **must** ensure
1291 that the business transaction will not be confirmed.

1292
1293

1294 REQUEST_STATUS

1295
1296 Sent to an Inferior, Superior or to a Decider to ask it to reply with STATUS. The receiver
1297 may reject the request with a FAULT(StatusRefused).
1298

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

1299

target-address the address to which the REQUEST_STATUS message is sent.
1300 This can be any of "decider-address-as-decider", "inferior-address-as-inferior" or
1301 "superior-address-as-superior".
1302

1303

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

1304

target identifier The identifier for the business transaction, or part of business
1305 transaction whose status is sought. If the target-address is an "decider-address-
1306 as-decider", this parameter shall be the "transaction-identifier" on the BEGUN
1307 message. If the "target-address" is an "inferior-address-as-inferior", this
1308 parameter shall be the "inferior-identifier" on the ENROL message. If the
1309 "target-address" is a an "superior-address-as-superior", this parameter shall be
1310 the "superior-identifier" on the CONTEXT.
1311

1312

qualifiers standardised or other qualifiers.

1313

1314
1315
1316 Types of FAULT possible (sent to "reply-address")
1317

1318

General

1319

Redirect – if the intended target now has a different address

1320

StatusRefused – if the receiver is not prepared to report its status to the
1321 sender of this message

1322

UnknownTransaction – if the target-identifier is unknown

1323

1324

1325 **STATUS**

1326
1327
1328
1329

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

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

1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342

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

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

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

status value	Meaning from Superior	Meaning from Inferior
<i>Created</i>	Not applicable	The Inferior exists (and is addressable) but it has not been enrolled with a Superior
<i>Enrolling</i>	Not applicable	ENROL has been sent, but ENROLLED is awaited
<i>Active</i>	New enrolment of inferiors is possible	The Inferior is enrolled
<i>Resigning</i>	Not applicable	RESIGN has been sent; RESIGNED is awaited
<i>Resigned</i>	Not applicable	RESIGNED has been received
<i>Preparing</i>	Not applicable	PREPARE has been received; PREPARED has not been sent
<i>Prepared</i>	Not applicable	PREPARED has been sent; no outcome has been received or autonomous decision made
<i>Confirming</i>	Confirm decision has been made or CONFIRM has been received	CONFIRM has been received; CONFIRMED/response has not

status value	Meaning from Superior	Meaning from Inferior
	as Inferior but responses from inferiors are pending	bee sent
<i>Confirmed</i>	CONFIRMED/responses have been received from all Inferiors	CONFIRMED/response has been sent
<i>Cancelling</i>	Cancel decision has been made but responses from inferiors are pending	CANCEL has been received or auto-cancel has been decided
<i>Cancelled</i>	CANCELLED has been received from all Inferiors	CANCELLED has been sent
<i>cancel-contradiction</i>	Not applicable	Autonomous cancel decision was made, CONFIRM received; CONTRADICTION has not been received
<i>confirm-contradiction</i>	Not applicable	Autonomous confirm decision was made, CANCEL received; CONTRADICTION has not been received
<i>Hazard</i>	A hazard has been reported from at least one Inferior	A hazard has been discovered; CONTRADICTION has not been received
<i>Contradicted</i>	Not applicable	CONTRADICTION has been received
<i>Unknown</i>	No state information for the target-identifier exists	No state information for the target-identifier exists
<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

1343

1344

qualifiers standardised or other qualifiers.

1345

1346 Types of FAULT possible

1347

General

1348

1349

FAULT

1350

1351

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

1352

1353

1354

Parameter

Type

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

1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372

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

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

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

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

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

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

1374
 1375 *UnknownParameter* A BTP message has been Free text explanation
 1376 received with an unrecognised
 1377 **q** parameter
 1378 **u**
 1379 **Qualifiers** standardised or other qualifiers.
 1380

1381 Note – If the carrier mechanism used for the transmission of BTP messages
 1382 is capable of delivering messages in a different order than they were sent in,
 1383 the “WrongState” FAULT is not sent and should be ignored if received.

1384
 1385 **REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES**

1386
 1387 REQUEST_INFERIOR_STATUSES may be sent to and INFERIOR_STATUSES sent from
 1388 any Decider, Superior or Inferior, asking it to report on the status of its relationships with
 1389 Inferiors (if any). Since Deciders are required to respond to
 1390 REQUEST_INFERIOR_STATUSES with INFERIOR_STATUSES but non-Deciders may
 1391 just issue FAULT(StatusRefused), and INFERIOR_STATUSES is also used as a reply to
 1392 other messages from Terminator to Decider, these messages are described below under the
 1393 messages used in the control relationships.
 1394

1395 **Messages used in the outcome relationships**

1396
 1397 **ENROL**

1398
 1399 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a
 1400 CONTEXT message in relation to an application request.
 1401 The actor issuing ENROL plays the role of Enroller.
 1402

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

1403
 1404 **target-address** the address to which the ENROL is sent. This will be the
 1405 “superior”-address-~~as-superior~~” from the CONTEXT message.

1406
1407 **superior-identifier**. The “superior-identifier” as on the CONTEXT message
1408
1409 **response- requested** true if an ENROLLED response is required, false
1410 otherwise. Default is false.
1411
1412 **reply-address** the address to which a replying ENROLLED is to be sent, if
1413 “response-requested” is true. If this field is absent and “response-requested” is
1414 true, the ENROLLED should be sent to the “inferior-address-as-inferior” (or one
1415 of them, at sender’s option)
1416
1417 inferior-address-as-inferior the address to which PREPARE, CONFIRM,
1418 CANCEL and SUPERIOR_STATE messages for this Inferior are to be sent.
1419
1420 **inferior-identifier** an identifier that identifies this Inferior. This shall be globally
1421 unambiguous..
1422
1423 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
1424 name” may be present.
1425

1426 Types of FAULT possible (sent to “reply-address”)

1427
1428 *General*

1429 *InvalidSuperior* – if “superior-identifier” is unknown

1430 *Redirect – if the Superior now has a different superior-address-as-*
1431 *superior*

1432 *DuplicateInferior* – if inferior with at least one of the set “inferior-
1433 address-as-inferior” the same and the same “inferior-identifier” is already
1434 enrolled

1435 *WrongState* – if it is too late to enrol new Inferiors (generally if the
1436 Superior has already sent a PREPARED message to its superior or
1437 terminator, or if it has already issued CONFIRM to other Inferiors).
1438

1439 The form ENROL/rsp-req refers to an ENROL message with “response-requested” having
1440 the value “true”; ENROL/no-rsp-req refers to an ENROL message with “response-requested”
1441 having the value “false”
1442

1443 ENROL/no-rsp-req is typically sent in relation to CONTEXT_REPLY/related. ENROL/rsp-
1444 req is typically when CONTEXT_REPLY/completed will be used (after the ENROLLED
1445 message has been received.)
1446

1447 **ENROLLED**

1448
1449 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been
1450 successfully enrolled (and will therefore be included in the termination exchanges)
1451

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

1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473

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

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

qualifiers standardised or other qualifiers.

No FAULT messages are issued on receiving ENROLLED.

RESIGN

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

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

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

1474
1475
1476
1477
1478
1479
1480
1481
1482
1483

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

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

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

1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507

1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524

qualifiers standardised or other qualifiers.

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

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

General

InvalidSuperior – if “superior-identifier” is unknown

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

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

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

RESIGNED

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

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

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

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

qualifiers standardised or other qualifiers.

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

Types of FAULT possible (sent to Superior address)

General

WrongState - if RESIGN has not been sent

1525
1526
1527
1528
1529
1530
1531

PREPARE

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

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

1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543

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

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

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

1544
1545
1546

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

1547
1548

Types of FAULT possible (sent to Superior address)

1549
1550
1551

General

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

1552
1553
1554
1555

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

1556
1557

PREPARED

1558
1559
1560
1561
1562
1563
1564

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

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

1565

1566

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

1567

1568

1569

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

1570

1571

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

1572

1573

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.

1574

1575

1576

1577

1578

1579

1580

1581

1582

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

1583

1584

1585

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

1586

1587

1588

1589

1590

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

1591

1592

1593

General

1594

InvalidSuperior – if “superior-identifier” is unknown

1595

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

1596

1597

1598

1599

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

1600

1601

1602
1603
1604
1605
1606
1607

CONFIRM

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

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

1608
1609
1610
1611
1612
1613
1614
1615
1616

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

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

qualifiers standardised or other qualifiers.

1617
1618
1619
1620

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

1621
1622
1623
1624
1625
1626

Types of FAULT possible (sent to Superior address)

General

InvalidInferior – if "inferior-identifier" is unknown

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

1627
1628

CONFIRMED

1629
1630
1631
1632
1633
1634
1635

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

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

Parameter	Type
qualifiers	List of qualifiers
1636	
1637	target-address the address to which the CONFIRMED is sent. This will be the
1638	Superior address as on the CONTEXT message.
1639	
1640	superior-identifier the “superior-identifier” as on the CONTEXT message.
1641	
1642	inferior-identifier the “inferior-identifier” as on the earlier ENROL message.
1643	
1644	
1645	confirm-received “true” if CONFIRMED is sent after receiving a CONFIRM
1646	message; “false” if an autonomous confirm decision has been made and either if
1647	no CONFIRM message has been received or the implementation cannot
1648	determine if CONFIRM has been received (due to loss of state information in a
1649	failure).
1650	
1651	qualifiers standardised or other qualifiers.

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

General

InvalidSuperior – if “superior-identifier” is unknown

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

1661 Note – A CONFIRMED message arriving before a CONFIRM message is
 1662 sent, or after a CANCEL has been sent will occur when the Inferior has
 1663 taken an autonomous decision and is not regarded as occurring in the wrong
 1664 state. (The latter will cause a CONTRADICTION message to be sent.)

1665 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm-
 1666 received” = “false”; CONFIRMED/response refers to a CONFIRMED message
 1667 with “confirm-received” = ”true”.
 1668
 1669

CANCEL

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

Parameter	Type
target-address	BTP address

inferior-identifier Identifier
qualifiers List of qualifiers

1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713

target-address the address to which the CANCEL message is sent. This will be the “~~inferior-address-as-inferior~~” from the ENROL message.

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

qualifiers standardised or other qualifiers.

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

Types of FAULT possible (sent to Superior address)

General

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

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

CANCELLED

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

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

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

Parameter

target-address BTP address
superior-identifier Identifier

inferior-identifier Identifier
 qualifiers List of qualifiers

1714
 1715
 1716
 1717
 1718
 1719
 1720
 1721
 1722
 1723

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

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

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

qualifiers standardised or other qualifiers.

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

1724
 1725
 1726
 1727
 1728
 1729
 1730
 1731
 1732

General

InvalidSuperior – if “superior-identifier” is unknown

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

WrongState – if CONFIRM has been sent

1733
 1734
 1735
 1736

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

1737
 1738
 1739
 1740

CONFIRM_ONE_PHASE

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

1741
 1742
 1743
 1744

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

1745
 1746
 1747

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

1748
 1749 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for
 1750 this Inferior.
 1751
 1752 **report hazard** Defines whether the superior wishes to be informed if a mixed
 1753 condition occurs for the operations associated with the Inferior. If “report-
 1754 hazard” is “true”, the Inferior will reply with HAZARD if a mixed condition
 1755 occurs, or if the Inferior cannot determine that a mixed condition has not
 1756 occurred. If “report-hazard” is false, the Inferior will report only its own decision,
 1757 regardless of whether that decision was correctly and consistently applied.
 1758 Default is false.
 1759
 1760 **qualifiers** standardised or other qualifiers.

1761
 1762 CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom
 1763 PREPARED has been received (subject to the requirement that there is only one enrolled
 1764 Inferior).

1765
 1766 Types of FAULT possible (sent to Superior address)

1767
 1768 *General*
 1769 *InvalidInferior* – if “inferior-identifier” is unknown
 1770 *WrongState* – if a PREPARE has already been sent to this Inferior

1771
 1772 **HAZARD**

1773
 1774 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly
 1775 and consistently cancel or confirm the operations in accord with the decision , or when the
 1776 Inferior is unable to determine that a “mixed” condition has not occurred.

1777
 1778 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there
 1779 is a mixed condition within its associated operations or is unable to determine that there is not
 1780 a mixed condition.

1781
 1782 Note - If the Inferior makes its own autonomous decision then it signals that
 1783 decision with CONFIRMED or CANCELLED and waits to receive a
 1784 confirmatory CONFIRM or CANCEL, or a CONTRADICTION if the
 1785 autonomous decision by the Inferior was the opposite of that made by the
 1786 Superior.

1787

Parameter	Type
target-address	BTP address
superior-identifier	Identifier

inferior-identifier	Identifier
level	mixed/possible
qualifiers	List of qualifiers

1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821

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

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

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

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

qualifiers standardised or other qualifiers.

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

General

InvalidSuperior – if “superior-identifier” is unknown

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

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

CONTRADICTION

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

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

1822

1823 **target-address** the address to which the CONTRADICTION message is sent.
 1824 This will be the "inferior-address-as-inferior" from the ENROL message.
 1825
 1826 **inferior-identifier** The "inferior-identifier" as on the earlier ENROL message for
 1827 this Inferior.
 1828
 1829 **qualifiers** standardised or other qualifiers.

1830 Types of FAULT possible (sent to Superior address)

1831 *General*

1832 *InvalidInferior* – if "inferior-identifier" is unknown

1833 *WrongState* – if neither CONFIRMED or CANCELLED has been sent
 1834 by this Inferior

1835 **SUPERIOR_STATE**

1836 Sent by a Superior as a query to an Inferior when

- 1837
- 1838 1. in the active state
 - 1839 2. there is uncertainty what state the Inferior has reached (due to recovery from
 1840 previous failure or other reason).

1841 Also sent by the Superior to the Inferior in response to a received INFERIOR_STATE, in
 1842 particular states.

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

1843
 1844
 1845
 1846
 1847 **target-address** the address to which the SUPERIOR_STATE message is sent.
 1848 This will be the "inferior-address-as-inferior" from the ENROL message.
 1849
 1850
 1851 **inferior-identifier** The "inferior-identifier" as on the earlier ENROL message for
 1852 this Inferior.
 1853
 1854
 1855 **status** states the current state of the Superior, in terms of its relation to this
 1856 Inferior only.
 1857
 1858
 1859

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

1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891

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

qualifiers standardised or other qualifiers.

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

A status of unknown shall only be sent if it has been determined for certain that the Superior has no knowledge of the Inferior, or (equivalently) it can be determined that the relationship 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 “response-requested” = “false”. SUPERIOR_STATE/abcd/y refers to a similar message, but with “response-requested” = “true”. The form SUPERIOR_STATE/*y refers to a SUPERIOR_STATE message with “response-requested” = “true” and any value for status.

INFERIOR_STATE

1892 Sent by an Inferior as a query when in the active state to a Superior, when (due recovery from
 1893 previous failure or other reason) there is uncertainty what state the Superior has reached.
 1894
 1895 Also sent by the Inferior to the Superior in response to a received SUPERIOR_STATE, in
 1896 particular states.
 1897

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

1898
 1899 **target-address** the address to which the INFERIOR_STATE is sent. This will
 1900 be the “target-address” as used the original ENROL message.
 1901

1902 **superior-identifier** The “superior-identifier” as used on the ENROL message
 1903

1904 **inferior-identifier** The “inferior-identifier” as on the ENROL message
 1905

1906 **status** states the current state of the Inferior for the atomic business transaction,
 1907 which corresponds to the last message sent to the Superior by (or in the case of
 1908 ENROL for) the Inferior
 1909

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

1910
 1911 **response-requested** “true” if INFERIOR_STATE is sent as a query at the
 1912 Superior’s initiative; “false” if INFERIOR_STATE is sent in reply to a received
 1913 SUPERIOR_STATE or other message. Can only be “true” if “status” is “active”
 1914 or “prepared-received”. Default is “false”
 1915

1916 **qualifiers** standardised or other qualifiers.
 1917

1918 The Superior, on receiving INFERIOR_STATE with “response-requested” = “true”, should
1919 reply in a timely manner by (depending on its state) repeating the previous message it sent or
1920 by sending SUPERIOR_STATE with the appropriate status value.

1921
1922 A status of “unknown” shall only be sent if it has been determined for certain that the Inferior
1923 has no knowledge of a relationship with the Superior. If there could be persistent information
1924 corresponding to the Superior, but it is not accessible from the entity receiving an
1925 SUPERIOR_STATE/*y (or other) message targetted on the Inferior or the entity cannot
1926 determine whether any such persistent information exists, the response shall be
1927 “inaccessible”.

1928
1929 INFERIOR_STATE/unknown is also used as a response to messages, other than
1930 SUPERIOR_STATE/*y that are received when the Inferior is not known (and it is known
1931 there is no state information for it).

1932
1933 A SUPERIOR_STATE/INFERIOR_STATE exchange that determines that one or both sides
1934 are in the active state does not require that the Inferior be cancelled (unlike some other two-
1935 phase commit protocols). The relationship between Superior and Inferior, and related
1936 application elements may be continued, with new application messages carrying the same
1937 CONTEXT. Similarly, if the Inferior is prepared but the Superior is active, there is no
1938 required impact on the progression of the relationship between them.

1939
1940 The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a
1941 value equivalent to “abcd” (for active, unknown and inaccessible) and with “response-
1942 requested” = “false”. INFERIOR_STATE/abcd/y refers to a similar message, but with
1943 “response-requested” = “true”. The form INFERIOR_STATE/*y refers to a
1944 INFERIOR_STATE message with “response-requested” = “true” and any value for status.

1945
1946

1947 REDIRECT

1948
1949 Sent when the address previously given for a Superior or Inferior is no longer valid and the
1950 relevant state information is now accessible with a different address (but the same superior or
1951 “inferior-identifier”).

1952

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

1953

1954 **target-address** the address to which the REDIRECT is sent. This may be the
 1955 “reply-address” from a received message or the address of the opposite side
 1956 (superior/inferior) as given in a CONTEXT or ENROL message
 1957
 1958 **superior-identifier** The “superior-identifier” as on the CONTEXT message and
 1959 used on an ENROL message. (present only if the REDIRECT is sent from the
 1960 Inferior).
 1961
 1962 **inferior-identifier** The “inferior-identifier” as on the ENROL message
 1963
 1964 **old-address** The previous address of the sender of REDIRECT. A match is
 1965 considered to apply if any of the “old-address” values match one that is already
 1966 known.
 1967
 1968 **new-address** The (set of alternatives) “new-address” values to be used for
 1969 messages sent to this entity.
 1970
 1971 **qualifiers** standardised or other qualifiers.
 1972
 1973 If the actor whose address is changed is an Inferior, the “new-address” value
 1974 replaces the “inferior-address-as-inferior” as present in the ENROL.
 1975
 1976 If the actor whose address is changed is a Superior, the “new-address” value
 1977 replaces the Superior address as present in the CONTEXT message (or as present
 1978 in any other mechanism used to establish the Superior:Inferior relationship).
 1979
 1980

1981 **Messages used in control relationships**

1982 **BEGIN**

1983 A request to a Factory to create a new Business Transaction. This may either be a new top-
 1984 level transaction, in which case the Composer or Coordinator will be the Decider, or the new
 1985 Business Transaction may be immediately made the Inferior within an existing Business
 1986 Transaction (thus creating a sub-Composer or sub-Coordinator).
 1987
 1988
 1989

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

1990

1991 **target-address** the address of the entity to which the BEGIN is sent. How this
1992 address is acquired and the nature of the entity are outside the scope of this
1993 specification.
1994

1995 **reply-address** the address to which the replying BEGUN and related
1996 CONTEXT message should be sent.
1997

1998 **transaction-type** identifies whether a new Cohesion or new Atom is to be
1999 created; this value will be the “superior-type” in the new CONTEXT
2000

2001 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
2002 timelimit” may be present on BEGIN, to set the timelimit for the new business
2003 transaction and will be copied to the new CONTEXT. The standard qualifier
2004 “Inferior name” may be present if there is a CONTEXT related to the BEGIN.
2005

2006 A new top-level Business Transaction is created if there is no CONTEXT related to the
2007 BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is
2008 created if the CONTEXT message for the existing Business Transaction is related to the
2009 BEGIN. In this case, the Factory is responsible for enrolling the new Composer or
2010 Coordinator as an Inferior of the Superior identified in that CONTEXT.
2011

2012 Note – This specification does not provide a standardised means to
2013 determine which of the Inferiors of a sub-Composer are in its confirm set.
2014 This is considered part of the application:inferior relationship.

2015 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction-type” having
2016 the corresponding value.
2017

2018 Types of FAULT possible (sent to “reply-address”)
2019

2020
2021 **General**

2022 *Redirect – if the Factory now has a different address*

2023 **WrongState** - only issued if there is a related CONTEXT, and the
2024 Superior identified by the CONTEXT is in the wrong state to enrol new
2025 Inferiors
2026

2027 **BEGUN**

2028
2029 BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT
2030 for the new business transaction.
2031

Parameter	Type
target-address	BTP address

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

2032

2033

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

2034

2035

2036

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

2037

2038

2039

2040

2041

2042

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

2043

2044

2045

2046

2047

2048

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

2049

2050

2051

2052

2053

2054

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

2055

2056

qualifiers standardised or other qualifiers.

2057

2058

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

2059

2060

2061

2062

2063

2064

No FAULT messages are issued on receiving BEGUN.

2065

2066

PREPARE_INFERIORS

2067

2068

Sent from a Terminator to a Decider, but only if it is a Cohesion Composer, to tell it to prepare all or some of its inferiors, by sending PREPARE to any that have not already sent

2069

2070

2071 PREPARED, RESIGN or CANCELLED to the Decider (Composer) on its relationships as
2072 Superior. If the inferiors-list parameter is absent, the request applies to all the inferiors; if the
2073 parameter is present, it applies only to the identified inferiors of the Decider (Composer).
2074

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

2075
2076 **target-address** the address to which the PREPARE_INFERIORS message is
2077 sent. This will be the decider-address from the BEGUN message.
2078

2079 **reply-address** the address of the Terminator sending the
2080 PREPARE_INFERIORS message.
2081

2082 **transaction identifier** identifies the Decider and will be the transaction-identifier
2083 from the BEGUN message.
2084

2085 **inferiors-list** defines which of the Inferiors of this Decider preparation is
2086 requested for, using the “inferior-identifiers” as on the ENROL received by the
2087 Decider (in its role as Superior). If this parameter is absent, the PREPARE
2088 applies to all Inferiors.
2089

2090 **qualifiers** standardised or other qualifiers.
2091
2092

2093 For all Inferiors identified in the inferiors-list parameter (all Inferiors if the parameter is
2094 absent), from which none of PREPARED, CANCELLED or RESIGNED has been received,
2095 the Decider shall issue PREPARE. It will reply to the Terminator, using the “reply-address”
2096 on the PREPARE_INFERIORS message, sending an INFERIOR_STATUSES message
2097 giving the status of the Inferiors identified on the inferiors-list parameter (all of them if the
2098 parameter was absent).
2099

2100 If one or more of the “inferior-identifier”s in the "inferior-list" is unknown (does not
2101 correspond to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. The Decider
2102 shall not send PREPARE to any Inferior.
2103

2104 Types of FAULT possible (sent to Superior address)
2105

2106 *General*

2107 *InvalidDecider* – if Decider address is unknown

2108 *Redirect* – if the Decider- now has a different decider-address-as-decider

2109 **UnknownTransaction** – if the transaction-identifier is unknown
 2110 **InvalidInferior** – if ~~an~~ **one or more** inferior-handles on the inferiors-list is
 2111 unknown
 2112 **WrongState** – if a CONFIRM_TRANSACTION or
 2113 CANCEL_TRANSACTION has already been received by this
 2114 Composer.

2115
 2116 The form PREPARE_INFERIORS/all refers to a PREPARE_INFERIORS message where
 2117 the “inferiors-list” parameter is absent. The form PREPARE_INFERIORS/specific refers to a
 2118 PREPARE_INFERIORS message where the “inferiors-list” parameter is present.

2120
 2121 **CONFIRM_TRANSACTION**

2122
 2123 Sent from a Terminator to a Decider to request confirmation of the business transaction. If the
 2124 business transaction is a Cohesion, the confirm-set is specified by the “inferiors-list”
 2125 parameter.
 2126

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

2127
 2128 **target-address** the address to which the CONFIRM_TRANSACTION message
 2129 is sent. This will be the **“decider-address-as-~~decider~~”** on the BEGUN message.

2130
 2131 **reply-address** the address of the Terminator sending the
 2132 CONFIRM_TRANSACTION message.

2133
 2134 **transaction-identifier** identifies the Decider. This will be the transaction-
 2135 identifier from the BEGUN message.

2136
 2137 **inferiors-list** defines which Inferiors enrolled with the Decider, if it is a
 2138 Cohesion Composer, are to be confirmed, using the “inferior-identifiers” as on
 2139 the ENROL received by the Decider (in its role as Superior). Shall be absent if
 2140 the Decider is an Atom Coordinator.

2141
 2142 **report-hazard** Defines whether the Terminator wishes to be informed of hazard
 2143 events and contradictory decisions within the business transaction. If “report-
 2144 hazard” is “true”, the receiver will wait until responses (CONFIRMED,

2145 CANCELLED or HAZARD) have been received from all of its inferiors,
2146 ensuring that any hazard events are reported. If “report-hazard” is “false”, the
2147 Decider will reply with TRANSACTION CONFIRMED_COMPLETE or
2148 TRANSACTION CANCELLED_COMPLETE as soon as the decision for the
2149 transaction is known.

2150
2151 **qualifiers** standardised or other qualifiers.

2152
2153 If the “inferiors-list” parameter is present, the Inferiors identified shall be the “confirm-set” of
2154 the Cohesion. If the parameter is absent and the business transaction is a Cohesion, the
2155 “confirm-set” shall be all remaining Inferiors. If the business transaction is an Atom, the
2156 “confirm-set” is automatically all the Inferiors.

2157
2158 Any Inferiors from which RESIGN is received are not counted in the confirm-set.

2159
2160 If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED
2161 has not been received, PREPARE shall be issued to that Inferior.

2162

2163 NOTE -- If PREPARE has been sent but PREPARED not yet received from
2164 an Inferior in the confirm-set, it is an implementation option whether and
2165 when to re-send PREPARE. The Superior implementation may choose to re-
2166 send PREPARE if there are indications that the earlier PREPARE was not
2167 delivered.

2168
2169 A confirm decision may be made only if PREPARED has been received from all Inferiors in
2170 the “confirm-set”. The making of the decision shall be persistent (and if it is not possible to
2171 persist the decision, it is not made). If there is only one remaining Inferior in the “confirm
2172 set” and PREPARE has not been sent to it, CONFIRM_ONE_PHASE may be sent to it.

2173
2174
2175 All remaining Inferiors that are not in the confirm set shall be cancelled.

2176
2177 If a confirm decision is made and “report-hazard” was “false”, a
2178 TRANSACTION CONFIRMED_COMPLETE message shall be sent to the “reply-address”.

2179
2180 If a cancel decision is made and “report-hazard” was “false”, a
2181 TRANSACTION CANCELLED_COMPLETE message shall be sent to the “reply-address”.

2182
2183 If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.
2184 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in
2185 the confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent
2186 to the “reply-address”.

2187

2188 If one or more of the "inferior-identifier"s in the "inferior-list" is unknown (does not
2189 correspond to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. The Decider
2190 shall not make a confirm decision and shall not send CONFIRM to any Inferior.

2191
2192 Types of FAULT possible (sent to “reply-address”)

2193
2194 **General**

2195 **InvalidDecider** – if Decider address is unknown

2196 **Redirect** – if the Decider now has a different decider-address-as-decider

2197 **UnknownTransaction** – if the transaction-identifier is unknown

2198 **InvalidInferior** – if ~~an~~ one or more inferior handles in the inferiors-list is
2199 unknown

2200 **WrongState** – if a CANCEL_TRANSACTION has already been
2201 received .

2202
2203 The form CONFIRM_TRANSACTION/all refers to a CONFIRM_TRANSACTION message
2204 where the “inferiors-list” parameter is absent. The form
2205 CONFIRM_TRANSACTION/specific refers to a CONFIRM_TRANSACTION message
2206 where the “inferiors-list” parameter is present.

2207 2208 TRANSACTION_CONFIRMED

2209
2210 A Decider sends TRANSACTION_CONFIRMED to a Terminator in reply to
2211 CONFIRM_TRANSACTION if all of the confirm-set confirms (and, for a Cohesion, all other
2212 Inferiors cancel) without reporting hazards, or if the Decider made a confirm decision and the
2213 CONFIRM_TRANSACTION had a “report-hazards” value of “false”.

2214

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

2215
2216 **target-address** the address to which the TRANSACTION_CONFIRMED is
2217 sent., this will be the “reply-address” from the CONFIRM_TRANSACTION
2218 message.

2219
2220 **transaction-identifier** the “transaction-identifier” as on the BEGUN message
2221 (i.e. the identifier of the Decider as a whole).

2222
2223 **qualifiers** standardised or other qualifiers.

2224

2225 Types of FAULT possible (sent to “decider-address-~~as-decider~~”)

2226

2227 **General**

2228 **InvalidTerminator** – if Terminator address is unknown

2229
2230
2231
2232
2233
2234
2235

UnknownTransaction – if the transaction-identifier is unknown

CANCEL_TRANSACTION

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

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

2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254

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

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

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

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

qualifiers standardised or other qualifiers.

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

Types of FAULT possible (sent to Superior address)

General

InvalidDecider – if Decider address is unknown

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

UnknownTransaction – if the transaction-identifier is unknown

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

2260
2261
2262
2263
2264
2265

2266
2267
2268
2269
2270
2271
2272

CANCEL_INFERIORS

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

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

2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292

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

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

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

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

qualifiers standardised or other qualifiers.

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

2293
2294
2295

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

2296
2297
2298
2299
2300
2301

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

2302 Types of FAULT possible (sent to Superior address)

2303

2304 *General*

2305 *InvalidDecider* – if Decider address is unknown

2306 *Redirect* – if the Decider now has a different decider-address-as-decider

2307 *UnknownTransaction* – if the transaction-identifier is unknown

2308 *InvalidInferior* – if ~~an~~ one or more inferior-handle on the inferiors-list is
2309 unknown

2310 *WrongState* – if a CONFIRM_TRANSACTION or

2311 CANCEL_TRANSACTION has been received by this Composer.

2312

2313

2314

2315 **TRANSACTION_CANCELLED**

2316

2317 A Decider sends TRANSACTION_CANCELLED to a Terminator in reply to
2318 CANCEL_TRANSACTION or in reply to CONFIRM_TRANSACTION if the Decider
2319 decided to cancel. In both cases, TRANSACTION_CANCELLED is used only if all Inferiors
2320 cancelled without reporting hazards or the CANCEL_TRANSACTION or
2321 CONFIRM_TRANSACTION had a “report-hazard” value of “false.”
2322

Parameter

target-address BTP address

transaction-identifier identifier

qualifiers List of qualifiers

2323

2324 **target-address** the address to which the TRANSACTION_CANCELLED is
2325 sent. This will be the “reply-address” from the CANCEL_TRANSACTION or
2326 CONFIRM_TRANSACTION message.

2327

2328 **transaction-identifier** the “transaction-identifier” as on the BEGUN message
2329 (i.e. the identifier of the Decider as a whole).

2330

2331 **qualifiers** standardised or other qualifiers.

2332

2333 Types of FAULT possible (sent to “~~decider~~-address-as-decider”)

2334

2335 *General*

2336 *InvalidTerminator* – if Terminator address is unknown

2337 *UnknownTransaction* – if the transaction-identifier is unknown

2338

2339

2340 **REQUEST_INFERIOR_STATUSES**

2341

2342 Sent to a Decider to ask it to report the status of its Inferiors with an INFERIOR_STATUSES
 2343 message. It can also be sent to any actor with an *“superior-address-as-superior”* or *“inferior-*
 2344 *address-as-inferior”*, asking it about the status of that transaction tree nodes Inferiors, if there
 2345 are any. In this latter case, the receiver may reject the request with a FAULT(StatusRefused).
 2346 If it is prepared to reply, but has no Inferiors, it replies with an INFERIOR_STATUSES with
 2347 an empty “status-list” parameter.
 2348

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

2349
 2350 **target-address** the address to which the REQUEST_STATUS message is sent.
 2351 When used to a Decider, this will be the *“decider-address-as-decider”* from the
 2352 BEGUN message. Otherwise it may be an *“superior-address-as-superior”* from a
 2353 CONTEXT or *“inferior-address-as-inferior”* from an ENROL message.
 2354

2355 **reply-address** the address to which the replying INFERIOR_STATUSES is to
 2356 be sent
 2357

2358 **target-identifier** identifies the transaction (or transaction tree node). When the
 2359 message is used to a Decider, this will be the transaction-identifier from the
 2360 BEGUN message. Otherwise it will be the superior-identifier from a CONTEXT
 2361 or an inferior-identifier from an ENROL message.
 2362

2363 **inferiors-list** defines which inferiors enrolled with the target are to be included
 2364 in the INFERIOR_STATUSES, using the “inferior-identifiers” as on the ENROL
 2365 received by the Decider (in its role as Superior). If the list is absent, the status of
 2366 all enrolled Inferiors will be reported.
 2367

2368 **qualifiers** standardised or other qualifiers.
 2369

2370 Types of FAULT possible (sent to reply-address)

2371
 2372 **General**

2373 *Redirect – if the intended target now has a different address*

2374 **StatusRefused** – if the receiver is not prepared to report its status to the
 2375 sender of this message. This “fault-type” shall not be issued when a Decider
 2376 receives REQUEST_STATUSES from the Terminator.

2377 **UnknownTransaction** – if the transaction-identifier is unknown
 2378
 2379

2380 The form REQUEST_INFERIOR_STATUSES/all refers to a REQUEST_STATUS with the
 2381 inferiors-list absent. The form REQUEST_INFERIOR_STATUS/specific refers to a
 2382 REQUEST_INFERIOR_STATUS with the inferiors-list present.

2383

2384 **INFERIOR_STATUSES**

2385

2386 Sent by a Decider to report the status of all or some of its inferiors in response to a
 2387 REQUEST_INFERIOR_STATUSES, PREPARE_INFERIORS, CANCEL_INFERIORS,
 2388 CANCEL_TRANSACTION with “report-hazard” value of “true” and
 2389 CONFIRM_TRANSACTION with “report-hazard” value of “true”. It is also used by any
 2390 actor in response to a received REQUEST_INFERIOR_STATUSES to report the status of
 2391 inferiors, if there are any.

2392

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

2393

2394 **target-address** the address to which the INFERIOR_STATUSES is sent. This
 2395 will be the “reply-address” on the received message

2396

2397 **responders-identifier** the target-identifier used on the
 2398 REQUEST_INFERIOR_STATUSES.

2399

2400 **status-list** contains a number of Status-items, each reporting the status of one of
 2401 the inferiors of the Decider. The fields of a Status-item are

2402

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

2403

2404 The status value reports the current status of the particular inferior, as known to
 2405 the Decider (Composer or Coordinator). Values are:

2406

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

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

2407

2408

2409

2410

2411

2412

2413

2414

2415

2416

2417

2418

2419

2420

2421

2422

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

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

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

General

InvalidTerminator – if Terminator address is unknown

UnknownTransaction – if the transaction-identifier is unknown

2423
2424
2425
2426

2427 **Groups – combinations of related messages**

2428
2429
2430
2431
2432
2433
2434
2435

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.

2436
2437

CONTEXT & application message

2438
2439
2440
2441
2442
2443
2444

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.

2445
2446
2447
2448
2449

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).

2450
2451
2452
2453
2454

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).

2455
2456
2457
2458

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.

2459
2460
2461
2462

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.

2463
2464
2465
2466
2467

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.

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
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514

CONTEXT_REPLY & ENROL

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

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

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

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

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

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

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

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

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

2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525

CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED

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

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

2526
2527

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

2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539

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

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

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.

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).

2544
2545
2546
2547
2548
2549
2550
2551
2552

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

CONTEXT_REPLY & ENROL & application message (& PREPARED)

2554
2555
2556
2557
2558
2559

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

Meaning: the relation between the BTP messages is as for the preceding groups, The transmission of the application message (and application effects implied by its

2560 transmission) has been associated with the Inferior identified by the ENROL and will be
2561 subject to the outcome delivered to that Inferior.

2562
2563 **target-address:** the “target-address” of the group is the “target-address” of the
2564 CONTEXT_REPLY which shall also be the “target-address” of the application message.
2565 The ENROL and PREPARED messages do not contain their “target-address” parameters.

2566
2567 The processing of ENROL and PREPARED messages is the same as for the previous
2568 groups.

2569
2570 This group can be used when participation in business transaction (normally a cohesion),
2571 is initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with
2572 some associated application semantic, performs some work for the transaction and sends
2573 an application message with a related ENROL. The CONTEXT_REPLY allows the
2574 addressing of the application (and the CONTEXT_REPLY) to be distinct from that of the
2575 Superior.

2576
2577 The actor receiving the group may associate the “inferior-identifier” received on the
2578 ENROL with the application message in a manner that is visible to the application
2579 receiving the message (e.g. for subsequent use in Terminator:Decider exchanges).

2580

2581 **BEGUN & CONTEXT**

2582

2583 **Meaning:** the CONTEXT is that for the new business transaction, containing the
2584 Superior address.

2585

2586 **target-address:** the “target-address” is that of the BEGUN message – this will be the
2587 “reply-address” of the earlier BEGIN message.

2588

2589 **BEGIN & CONTEXT**

2590

2591 **Meaning:** the new business transaction is to be an Inferior (sub-coordinator or sub-
2592 composer) of the Superior identified by the CONTEXT. The Factory (receiver of the
2593 BEGIN) will perform the enrolment.

2594

2595 **target-address:** the “target-address” is that of the BEGIN – this will be the address of the
2596 Factory.

2597

2598 **Standard qualifiers**

2599

2600 The following qualifiers are expected to be of general use to many applications and
2601 environments. The URI “urn:oasis:names:tc:BTP:1.0:qualifiers” is used in the
2602 Qualifier group value for the qualifiers defined here.

2603

2604

2605 **Transaction timelimit**

2606

2607 The transaction **timelimit** allows the Superior (or an application element initiating the
2608 business transaction) to indicate the expected length of the active phase, and thus give an
2609 indication to the Inferior of when it would be appropriate to initiate cancellation if the active
2610 phase appears to continue too long. The time limit ends (the clock stops) when the Inferior
2611 decides to be prepared and issues **PREPARED** to the Superior.
2612

2613 It should be noted that the expiry of the time limit does not change the permissible actions of
2614 the Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is
2615 **permitted** to initiate cancellation for internal reasons. The **timelimit** gives an indication to the
2616 entity of when it will be useful to exercise this right.
2617

2618 The qualifier is propagated on a **CONTEXT** message.
2619

2620 The “Qualifier name” shall be “`transaction-timelimit`” .
2621

2622 The “Content” shall contain the following field:
2623

Content field	Type
Timelimit	Integer

2624
2625 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
2626 time of transmission of the containing **CONTEXT**, of the active phase of the business
2627 transaction.
2628

2629 **Inferior timeout**

2630
2631 This qualifier allows an Inferior to limit the duration of its “promise”, when sending
2632 **PREPARED**, that it will maintain the ability to confirm or cancel the effects of all associated
2633 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or
2634 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and
2635 can apply the decision indicated in the qualifier.
2636

2637 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply
2638 a confirm or cancel decision before the **CONFIRM** or **CANCEL** is received and before this
2639 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,
2640 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
2641 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
2642 expiry of this timeout, is liable to cause contradictory decisions across the business
2643 transaction. BTP ensures that at least the occurrence of such a contradiction will be
2644 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic
2645 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
2646 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
2647 change in the probability that it will.
2648

2649 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
2650 intended decision, only that is at liberty to do so. An implementation may choose to only

2651 apply the decision if there is contention for the underlying resource, for example.
2652 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for
2653 the business transaction are made before these timeouts expire (and allow a margin of error
2654 for network latency etc.).

2655
2656 The qualifier may be present on a PREPARED message. If the PREPARED message has the
2657 “default-is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall
2658 have the value “cancel”.

2659
2660 The “Qualifier name” shall be “inferior-timeout”.

2661
2662 The “Content” shall contain the following fields:
2663

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

2664
2665 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
2666 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
2667 effects of the associated operations, as ordered by the receiving Superior.
2668

2669 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an
2670 autonomous decision is made.

2671
2672 **Minimum inferior timeout**

2673
2674 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the
2675 Inferior. If a Superior knows that the decision for the business transaction will not be
2676 determined for some period, it can require that Inferiors do not send PREPARED messages
2677 with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to
2678 send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with
2679 CANCELLED.

2680
2681 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If
2682 present on more than one, and with different values of the MinimumTimeout field, the value
2683 on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall
2684 prevail over either of the others.

2685
2686 The “Qualifier name” shall be “minimum-inferior-timeout”.

2687
2688 The “Content” shall contain the following field:
2689

Content field	Type
MinimumTimeout	Integer

2690

2691 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be
2692 acceptable in the Inferior timeout qualifier on an answering PREPARED message.

2693
2694 **Inferior name**

2695
2696 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on
2697 INFERIOR_STATUSES and thus allow the Terminator to determine which Inferior (of the
2698 Composer or Coordinator) is related to which application work. This is in addition to the
2699 “inferior-identifier” field. The name can be human-readable and can also be used in fault
2700 tracing, debugging and auditing.

2701
2702 The name is never used by the BTP actors themselves to identify each other or to direct
2703 messages. (The BTP actors use the addresses and the identifiers in the message parameters
2704 for those purposes.)

2705
2706 This specification makes no requirement that the names are unambiguous within any scope
2707 (unlike the globally unambiguous “inferior-identifier” on ENROLLED and BEGUN). Other
2708 specifications, including those defining use of BTP with a particular application may place
2709 requirements on the use and form of the names. (This may include reference to information
2710 passed in application messages or in other, non-standardised, qualifiers.)

2711
2712 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item
2713 in INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if
2714 present, the same qualifier value **should** be included in the consequent ENROL. If
2715 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an
2716 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.

2717
2718 The “Qualifier -name” shall be “inferior-name”

2719
2720 The “Content” shall contain the following fields:

2721

Content field	Type
inferior-name	String

2722
2723 **Inferior name** the name assigned to the enrolling Inferior.
2724

2725

State Tables

2726

Explanation of the state tables

2727

2728

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).

2731

2732

2733

2734

2735

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.

2736

2737

2738

2739

2740

2741

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.

2742

2743

2744

Status queries

2745

2746

2747

2748

2749

2750

2751

2752

2753

2754

2755

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.

2756

2757

2758

2759

2760

2761

2762

2763

2764

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

2765

Decision events

2766

2767

2768

2769

2770

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

2771 business transaction and on features of the implementation (e.g. making of a persistent record
2772 of the decision means that the information will survive at least some failures that otherwise
2773 lose state information, but the level of survival depends on the purpose of the
2774 implementation). [Table 2](#) [Table 2](#) [Table 2](#) and [Table 3](#) [Table 3](#) [Table 3](#) define the decision
2775 events.

2776
2777 In some cases, an implementation may not need to make an active change to have a persistent
2778 record of a decision, provided that the implementation will restore itself to the appropriate
2779 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,
2780 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via
2781 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a
2782 record of “decide to cancel autonomously”, provided it always updated such a record as part
2783 of the “apply ordered confirmation” decision event.

2784
2785 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
2786 PREPARE indicates that the application exchange (to associate operations with the Inferior)
2787 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
2788 state corresponding to an incomplete application exchange. However, implementations are
2789 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
2790 that experiences failure after sending PREPARE may, on recovery, have no information
2791 about the transaction, in which case it is considered to be in the completed state (Z), which
2792 will imply the cancellation of the Inferior and its associated operations.

2793
2794 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its
2795 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a
2796 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
2797 persistent information (which would combine both superior and inferior information, pointing
2798 both up and down the tree).

2799
2800

2801 **Disruptions – failure events**

2802
2803 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or
2804 may) cause a change of state. The disruption events in the state tables model different extents
2805 of loss of state information. An implementation is not required to exhibit all the possible
2806 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
2807 possible disruption.

2808
2809 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,
2810 which involves possible interruption of service and loss of messages in transit, but no change
2811 of state (either because no state information was lost, or because recovery from persistent
2812 information restores the implementation to the same state). The “disruption 0” event would
2813 typically be an appropriate abstraction for a communication failure.

2814
2815
2816

Invalid cells and assumptions of the communication mechanism

2817 The empty cells in state table represent events that cannot happen. For events corresponding
2818 to sending a message or any of the decision events, this prohibition is absolute – e.g. a
2819 conformant implementation in the Superior active state “B1” will not send CONFIRM. For
2820 events corresponding to receiving a message, the interpretation depends on the properties of
2821 the underlying communications mechanism.

2822
2823 For all communication mechanisms, it is assumed that

- 2824 a) the two directions of the Superior:Inferior communication are not synchronised –
2825 that is messages travelling in opposite directions can cross each other to any
2826 degree; any number of messages may be in transit in either direction; and
- 2827 b) messages may be lost arbitrarily

2828
2829 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered
2830 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a
2831 state where the corresponding cell is empty indicates that the far-side has sent a message out
2832 of order – a FAULT message with the “fault-type” “WrongState” can be returned.

2833
2834 If the communication mechanisms cannot guarantee ordered delivery, then messages received
2835 where the corresponding cell is empty should be ignored. Assuming the far-side is
2836 conformant, these messages can assumed to be “stale” and have been overtaken by messages
2837 sent later but already delivered. (If the far-side is non-conformant, there is a problem
2838 anyway).

2839 2840 **Meaning of state table events**

2841 The tables in this section define the events (rows) in the state tables. [Table 1](#) ~~Table 1~~ ~~Table 1~~
2842 defines the events corresponding to sending or receiving BTP messages and the disruption
2843 events. [Table 2](#) ~~Table 2~~ ~~Table 2~~ describes the decision events for an Inferior, [Table 3](#) ~~Table~~
2844 ~~3~~ ~~Table 3~~ those for a Superior.

2845
2846
2847 The decision events for a Superior, defined in [Table 3](#) ~~Table 3~~ ~~Table 3~~ cannot be specified
2848 without reference to other Inferiors to which it is Superior and to its relation with the
2849 application or other entity that (acting ultimately on behalf of the application) drives it.

2850
2851 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and
2852 which are to be treated as an atomic decision unit with (and thus including) the Inferior on
2853 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior-
2854 type” of “atom”, this will be all Inferiors established with same Superior address and
2855 “superior-identifier” except those from which RESIGN has been received. If the CONTEXT
2856 had “superior-type” of “cohesion”, the “remaining Inferiors” excludes any that it has been
2857 determined will be cancelled, as well as any that have resigned – in other words it includes
2858 only those for which a confirm decision is still possible or has been made. The determination
2859 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some
2860 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this
2861 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.

2862

2863 In order to ensure that the confirmation decision **is** delivered to all remaining Inferiors,
 2864 despite failures, the Superior must persistently record which these Inferiors are (i.e. their
 2865 addresses and identifiers). It must also either record that the decision is confirm, or ensure
 2866 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it
 2867 will be told about it. This latter would apply if the Superior were also BTP Inferior to another
 2868 entity which persisted a confirm decision (or recursively deferred it still higher). However,
 2869 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
 2870 behaviour of asking another entity to make (and persist) the confirm decision is termed
 2871 "offering confirmation" - the Superior offers the possible confirmation of itself, and its
 2872 remaining Inferiors to some other entity. If that entity (or something higher up) then does
 2873 make and persist a confirm decision, the Superior is "instructed to confirm" (which is
 2874 equivalent BTP CONFIRM).

2875
 2876 The application, or an entity acting indirectly on behalf of the application, may request a
 2877 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
 2878 more operations associated with the Inferior. Following a request to prepare all remaining
 2879 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
 2880 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
 2881 application.)

2882
 2883 The application, or an entity acting indirectly on behalf of the application, may also request
 2884 confirmation. This means the Superior is to attempt to make and persist a confirm decision
 2885 itself, rather than offer confirmation.

2886
 2887
 2888

Table 1 : send, receive and disruption events

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

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

2889

2890

Table 2 : Decision events for Inferior

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

Event name	Meaning
detect problem	<ul style="list-style-type: none"> • For at least some of the associated operations, EITHER <ul style="list-style-type: none"> ○ they cannot be consistently cancelled or consistently confirmed; OR ○ it cannot be determined whether they will be cancelled or confirmed • AND, information about this is not persistent
detect and record problem	<ul style="list-style-type: none"> • As for the first condition of “detect problem” • information recording this has been persisted (to the degree considered appropriate), or the detection itself is persistent. (i.e. will be re-detected on recovery)

2891

2892

Table 3: Decision events for a Superior

Event name	Meaning
decide to confirm one-phase	<ul style="list-style-type: none"> • All associated application messages to be sent to the service have been sent; • There are no other remaining Inferiors • If an atom, all enrolments that would create other Inferiors have completed (no outstanding CONTEXT_REPLYS) • The Superior has been requested to confirm
decide to prepare	<ul style="list-style-type: none"> • All associated application messages to be sent to the service have been sent; • The Superior has been requested to prepare this Inferior
decide to confirm	<ul style="list-style-type: none"> • Either <ul style="list-style-type: none"> ○ PREPARED or PREPARED/cancel has been received from all other remaining Inferiors; AND ○ Superior has been requested to confirm; AND ○ persistent information records the confirm decision and identifies all remaining Inferiors; • Or <ul style="list-style-type: none"> ○ persistent information records an offer of confirmation and has been instructed to confirm
decide to cancel	<ul style="list-style-type: none"> • Superior has not offered confirmation; OR • Superior has offered confirmation and has been instructed to cancel; OR

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

2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921

Persistent information

Persisted information (especially prepared information at an Inferior, confirm information at a Superior) may include qualifications of the state carried in Qualifiers of the corresponding message (e.g. inferior timeouts in prepared information). It may also include application-specific information (especially in Inferiors) to allow the future confirmation or cancellation of the associated operations. In some cases it will also include information allowing an application message sent with a BTP message (e.g. PREPARED) to be repeated.

The “effective” removal of persistent information allows for the possibility that the information is retained (perhaps for audit and tracing purposes) but some change to the persistent information (as a whole) means that if there is a failure after such change, on recovery, the persistent information does not cause the endpoint to return the state it would have recovered to before the change.

In all cases, the degree to which information described as “persistent” will survive failure is a configuration and implementation option. An implementation **should** describe the level of failure that it is capable of surviving. For applications manipulating information that is itself volatile (e.g. network configurations), there is no requirement to make the BTP state information more persistent than the application information.

The degree of persistence of the recording of a hazard (problem) at an Inferior and recording of a detected contradiction at a Superior may be different from that applying to the persistent prepared and confirm information. Implementations and configuration may choose to pass hazard and contradiction information via management mechanisms rather than through BTP. Such passing of information to a management mechanism could be treated as “record problem” or “record contradiction”.

Table 4 : Superior states

State	summary
I1	CONTEXT created
A1	ENROLing
B1	ENROLLED (active)
C1	resigning
D1	PREPARE sent
E1	PREPARED received
E2	PREPARED/cancel received
F1	CONFIRM sent
F2	completed after confirm
G1	cancel decided
G2	CANCEL sent
G3	cancelling, RESIGN received
G4	both cancelled
H1	inferior autonomously confirmed
J1	Inferior autonomously cancelled
K1	confirmed, contradiction detected
L1	cancelled, contradiction detected
P1	hazard reported
P2	hazard reported in null state
P3	hazard reported after confirm decision
P4	hazard reported after cancel decision
Q1	contradiction detected in null state
R1	Contradiction or hazard recorded
R2	completed after contradiction or hazard recorded
S1	one-phase confirm decided
Y1	completed queried
Z	completed and unknown

Table 5 : Inferior states

State	summary
i1	aware of CONTEXT
a1	enrolling
b1	enrolled
c1	resigning
d1	preparing
e1	prepared
e2	prepared,default to cancel
f1	confirming
f2	confirming after default cancel
g1	CANCEL received in prepared state
g2	CANCEL received in prepared/cancel state
h1	Autonomously confirmed
h2	autonomously confirmed, superior confirmed
j1	autonomously cancelled
j2	autonomously cancelled, superior cancelled
k1	autonomously cancelled, contradicted
k2	autonomously cancelled, CONTRADICTION received
l1	autonomously confirmed, contradicted
l2	autonomously confirmed, CONTRADICTION received
m1	confirmation applied
n1	cancelling
p1	hazard detected, not recorded
p2	hazard detected in prepared state, not recorded
q1	hazard recorded
s1	CONFIRM_ONE_PHASE received after prepared state
s2	CONFIRM_ONE_PHASE received
s3	CONFIRM_ONE_PHASE received, confirming
s4	CONFIRM_ONE_PHASE received, cancelling
s5	CONFIRM_ONE_PHASE received, hazard detected
s6	CONFIRM_ONE_PHASE received, hazard recorded
x1	completed, presuming abort
x2	completed, presuming abort after prepared/cancel

State	summary
y1	completed, queried
y2	completed, default cancel, a message received
z	completed
z1	completed with default cancel

2925

2926

2927

2928

2929

2930

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

Table 6: Superior state table – normal forward progression

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

Table 7: Superior state table – cancellation and contradiction

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

Table 8: Superior state table – hazard and request confirm

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

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

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

2937

2938

Table 10: Inferior state table – normal forward progression

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

2939

2940

Table 11: Inferior state table – cancellation and contradiction

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

2941

2942

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

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

Table 13: Inferior state table – request confirm states

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

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

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

2947

2948

2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993

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.

2994 After recovery from node failure, the implementation behaves much as if a communication
2995 failure had occurred.

2996

2997 **Persistent information**

2998

2999 BTP requires that some decision events are persisted – that information recording an
3000 Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s
3001 autonomous decision survive failure. Making the first two decisions persistent ensures that a
3002 consistent decision can be reached for the business transaction and that it is delivered to all
3003 involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to
3004 ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the
3005 contradiction will be reported to the Superior, despite failures.

3006

3007 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
3008 active state (unlike many transaction protocols, where a communication or endpoint failure in
3009 active state would invariably cause rollback of the transaction). Recovery in the active state
3010 may require that the application exchange is resynchronised as well – BTP does not directly
3011 support this, but does allow continuation of the business transaction as such. In the state
3012 tables, from some states, there are several levels of disruption, distinguished by which state
3013 the implementation transits to – this represents the survival of different extents of state
3014 information over failure and recovery. The different levels of disruption describe legitimate
3015 states for the endpoint to be in after it has recovered – **they do not require that all**
3016 **implementations are able to exhibit the appropriate partial loss of state information.**

3017

3018 The absence of a destination state for the disruption events means that such a transition is not
3019 legitimate – thus, for example, an Inferior that has decided to be prepared will always recover
3020 to the same state, by virtue of the information persisted in the “decide to be prepared” event.

3020

3021 Apart from the (optional) recovery in active state, BTP follows the well-known presume-
3022 abort model – it is only required that information be persisted when decisions are made (and
3023 not, e.g. on enrolment). This means that on recovery, one side may have persistent
3024 information but the other does not. This occurs when an Inferior has decided to be prepared
3025 but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the
3026 Superior did confirm, and the Inferior applied the confirm, removed its persistent information
3027 but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it
3028 still had the persistent information when the failure occurred).

3029

3030 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to
3031 re-establish communication with the Superior, to apply a confirm decision and to apply a
3032 cancel decision. It will thus need to include

3033

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

3034

Superior address (as on CONTEXT)

3035

“superior-identifier” (as on CONTEXT)

3036

default-is-cancel value (as on PREPARED)

3037

3038 The information needed to apply confirm/cancel decisions will depend on the application and
3039 the associated operations. It may also normally be necessary to persist any qualifiers that

3040 were sent with the PREPARED message or application messages sent with the PREPARED,
3041 since the PREPARED message will be repeated if a failure occurs.

3042
3043 A Superior must record corresponding information to allow it to re-establish communication
3044 with the Inferior:

3045 Inferior address (as on ENROL)
3046 “inferior-identifier” (as on ENROL)

3047
3048 A Superior that is the Decider for the business transaction need only persist this information
3049 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A
3050 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as
3051 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as
3052 Superior (to this Inferior) as part of the persistent information of its decision to be prepared
3053 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and
3054 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If
3055 CONFIRM is received, the persistent information may be changed to show the confirm
3056 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.
3057 If the persistent information is left unchanged and there is a node failure, on recovery the
3058 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision
3059 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

3060
3061 After failure, an implementation may not be able to restore an endpoint to the appropriate
3062 state immediately – in particular, the necessary persistent information may be inaccessible,
3063 although the implementation can respond to received BTP messages. In such a case, a
3064 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a “response-
3065 requested” value “false”) with SUPERIOR_STATE/inaccessible and an Inferior to any BTP
3066 message except SUPERIOR_STATE/* with “INFERIOR_STATE/inaccessible. Receipt of
3067 the *_STATE/inaccessible messages has no effect on the endpoint state.

3068 **Redirection**

3069
3070
3071 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
3072 that there are cases where one side will attempt to re-establish communication when there is
3073 no persistent information for the relationship at the far-end. In such cases, it is important the
3074 side that is attempting recovery can distinguish between unsuccessful attempts to connect to
3075 the holder of the persistent information and when the information no longer exists. If the peer
3076 information does not exist, this side can draw conclusions and complete appropriately; if they
3077 merely fail to get through they are stuck in attempting recovery.

3078
3079 Two mechanisms are provided to make it possible that even when one side of a
3080 Superior:Inferior relationship has completed, that a message can eventually get through to
3081 something that can definitively report the status, distinguishing this case from a temporary
3082 inability to access the state of a continuing transaction element. The mechanisms are:

- 3083 o Address fields which provide a “callback address” can be a set of addresses,
3084 which are alternatives one of which is chosen as the “target-address” for the
3085 future message. If the sender of that message finds the address does not work,
3086 it can try a different alternative.

3087 o The REDIRECT message can be used to inform the peer that an address
3088 previously given is no longer valid and to supply a replacement address (or
3089 set of addresses). REDIRECT can be issued either as a response to receipt of
3090 a message or spontaneously.

3091
3092 The two mechanisms can be used in combination, with one or more of the original set of
3093 addresses just being a redirector, which does not itself ever have direct access to the state
3094 information for the transaction, but will respond to any message with an appropriate
3095 REDIRECT.

3096
3097 An alternative implementation approach is to have a single addressable entity that uses the
3098 same address for all transactions, distinguishing them by identifier, and which always
3099 recovers to use the same address. Such an implementation would not need to supply
3100 “backup” addresses (and would only use REDIRECT if it was being permanently migrated).

3101 3102 **Terminator:Decider failures**

3103
3104 BTP does not provide facilities or impose requirements on the recovery of
3105 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
3106 may survive failures (by retaining knowledge of the Decider’s address and identifier), but this
3107 is an implementation option. Although a Decider (if it decides to confirm) will persist
3108 information about the confirm decision, it is not required, after failure, to remain accessible
3109 using the inferior address it offered to the Terminator. Any such recovery is an
3110 implementation option.

3111
3112 A Decider’s address (as returned on BEGUN) may be a set of addresses, allowing a failed
3113 Decider to be recovered at a different address.

3114
3115 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
3116 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for
3117 a CONFIRM_TRANSACTION that will never arrive, the standard qualifier “Transaction
3118 timelimit” can be used (by the Initiator) to inform the Decider when it can assume the
3119 Terminator will not issue CONFIRM_TRANSACTION and so it (the Decider) should initiate
3120 cancellation.

3121 3122 **XML representation of Message Set**

3123
3124 This section describes the syntax for BTP messages in XML. These XML messages represent
3125 a midpoint between the abstract messages and what actually gets sent on the wire.

3126
3127 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)
3128 [3121](#)

3129
3130 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:[1.0:core.xml](#)

3131
3132 In addition to an XML schema, this specification uses an informal syntax to describe the
3133 structure of the BTP messages. The syntax appears as an XML instance, but the values

3134 contain data types instead of values. The following symbols are appended to some of the
3135 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
3136 these symbols corresponds to "one and only one."

3137

3138 Addresses

3139

3140 As described in the “Abstract Message and Associated Contracts – Addresses” section, a BTP
3141 address comprises three parts, and for a “target-address” only the “additional information”
3142 field is inside the BTP messages. For all BTP messages whose abstract form includes a
3143 “target-address” parameter, the corresponding XML representation includes a “target-
3144 additional-information” element. This element may be omitted if it would be empty.

3145

3146 For other addresses, all three fields are represent, as in:

3147

```
3148 <ctp:some-address>  
3149   <ctp:binding-name>...carrier binding URI...</ctp:binding-name>  
3150   <ctp:binding-address>...carrier specific  
3151   address...</ctp:binding-address>  
3152   <ctp:additional-information>...optional additional addressing  
3153   information...</ctp:additional-information> ?  
3154 </ctp:some-address>
```

3155

3156

3157 A "published" address can be a set of <some-address>, which are alternatives which can be
3158 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to
3159 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice
3160 of which address to use (depending on which binding is preferable.) In the case where
3161 multiple addresses are used for redundancy, a priority attribute can be specified to help the
3162 receiver choose among the addresses- the address with the highest priority should be used,
3163 other things being equal. The priority is used as a hint and does not enforce any behaviour in
3164 the receiver of the message. Default priority is a value of 1.

3165

3166 Qualifiers

3167 The “Qualifier name” is used as the element name, within the namespace of the “Qualifier
3168 group”.

3169

3170 Examples:

3171

```
3172 <ctpq:inferior-timeout  
3173   xmlns:ctpq="urn:oasis:names:tc:BTP:1.0:qualifiers"  
3174   xmlns:ctp="urn:oasis:names:tc:BTP:1.0:corexml"  
3175   ctp:must-be-understood="false"  
3176   ctp:to-be-propagated="false">1800</ctpq:inferior-timeout>
```

3176

3177

```
3178 <auth:username  
3179   xmlns:auth="http://www.example.com/ns/auth"  
3180   xmlns:ctp="urn:oasis:names:tc:BTP:1.0:corexml"  
3181   ctp:must-be-understood="true"  
3182   ctp:to-be-propagated="true">jtauber</auth:username>
```

3182

3183 Attributes must-be-understood **has default value “true”** and to-be-propagated has default
3184 value “false”.

3185

3186 Identifiers

3187

3188 Identifiers shall be URIs "

3189

3190

Note – Identifiers need to be globally unambiguous. Apart from their
3191 generation, the only operation the BTP implementations have to perform on
3192 identifiers is to match them.

3193

3194 Message References

3195 Each BTP message has an optional id attribute to give it a unique identifier. An application
3196 can make use of those identifiers, but no processing is enforced.

3197

3198 Messages

3199

3200 CONTEXT

3201

```
<btp:context id?>  
  <btp:superior-address> +  
    ...address...  
  </btp:superior-address>  
  <btp:superior-identifier>...URI...</btp:superior-identifier>  
  <btp:reply-address> ?  
    ...address...  
  </btp:reply-address>  
  <btp:superior-type>cohesion|atom</btp:superior-type>  
  <btp:qualifiers> ?  
    ...qualifiers...  
  </btp:qualifiers>
```

3214 </btp:context>

3215

3216 CONTEXT_REPLY

3217

```
<btp:context-reply id?>  
  <btp:target-additional-information> ?  
    ...additional address information...  
  </btp:target-additional-information>  
  
  <btp:superior-identifier>...URI...</btp:superior-identifier>  
  <btp:completion-  
3225 status>completed|related|repudiated</btp:completion-status>  
  <btp:qualifiers> ?  
    ...qualifiers...  
  </btp:qualifiers>  
  </btp:context-reply>
```

3230

3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281

REQUEST_STATUS

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

STATUS

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

FAULT

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

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

- 3282
- 3283 o communication-failure
- 3284 o duplicate-inferior
- 3285 o general
- 3286 o invalid-decider
- 3287 o invalid-inferior
- 3288 o invalid-superior
- 3289 o status-refused
- 3290 o invalid-terminator
- 3291 o unknown-parameter
- 3292 o unknown-transaction
- 3293 o unsupported-qualifier
- 3294 o wrong-state

3295

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

3299

Fault data can take on various forms:

3300

Free text:

3301

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

3302

Identifier:

3303

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

3304

3305

3306

Inferior Identity:

3307

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

3308

3309

3310

ENROL

3311

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

3312

3313

3314

3315

3316

3317

3318

3319

3320

3321

3322

3323

3324

3325

3326

3327

3328

3329

```
3330 </btp:reply-address>
3331 <btp:inferior-address> +
3332 ...address...
3333 </btp:inferior-address>
3334 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3335 <btp:qualifiers> ?
3336 ...qualifiers...
3337 </btp:qualifiers>
3338 </btp:enrol>
```

3339
3340

ENROLLED

```
3341
3342
3343 <btp:enrolled id?>
3344 <btp:target-additional-information> ?
3345 ...additional address information...
3346 </btp:target-additional-information>
3347 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3348 <btp:qualifiers> ?
3349 ...qualifiers...
3350 </btp:qualifiers>
3351 </btp:enrolled>
```

3352
3353

RESIGN

```
3354
3355
3356 <btp:resign id?>
3357 <btp:target-additional-information> ?
3358 ...additional address information...
3359 </btp:target-additional-information>
3360 <btp:superior-identifier>...URI...</btp:superior-identifier>
3361 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3362 <btp:response-requested>true|false</btp:response-requested>
3363 <btp:qualifiers> ?
3364 ...qualifiers...
3365 </btp:qualifiers>
3366 </btp:resign>
```

3367
3368

RESIGNED

```
3369
3370
3371 <btp:resigned id?>
3372 <btp:target-additional-information> ?
3373 ...additional address information...
3374 </btp:target-additional-information>
3375 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3376 <btp:qualifiers> ?
3377 ...qualifiers...
3378 </btp:qualifiers>
3379 </btp:resigned>
```

3380

3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431

PREPARE

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

PREPARED

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

CONFIRM

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

CONFIRMED

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

```
3432 <ctp:qualifiers> ?
3433   ...qualifiers...
3434 </ctp:qualifiers>
3435 </ctp:confirmed>
```

CANCEL

```
3439
3440 <ctp:cancel id?>
3441   <ctp:target-additional-information> ?
3442     ...additional address information...
3443   </ctp:target-additional-information>
3444   <ctp:inferior-identifier>...URI...</ctp:inferior-identifier>
3445   <ctp:reply-address> ?
3446     ...address...
3447   </ctp:reply-address>
3448   <ctp:qualifiers> ?
3449     ...qualifiers...
3450   </ctp:qualifiers>
3451 </ctp:cancel>
```

CANCELLED

```
3452
3453
3454
3455
3456 <ctp:cancelled id?>
3457   <ctp:target-additional-information> ?
3458     ...additional address information...
3459   </ctp:target-additional-information>
3460   <ctp:superior-identifier>...URI...</ctp:superior-identifier>
3461
3462   <ctp:inferior-identifier>...URI...</ctp:inferior-identifier> ?
3463   <ctp:qualifiers> ?
3464     ...qualifiers...
3465   </ctp:qualifiers>
3466 </ctp:cancelled>
```

CONFIRM_ONE_PHASE

```
3467
3468
3469
3470
3471 <ctp:confirm-one-phase id?>
3472   <ctp:target-additional-information> ?
3473     ...additional address information...
3474   </ctp:target-additional-information>
3475   <ctp:inferior-identifier>...URI...</ctp:inferior-identifier>
3476   <ctp:report-hazard>true|false</ctp:report-hazard>
3477   <ctp:qualifiers> ?
3478     ...qualifiers...
3479   </ctp:qualifiers>
3480 </ctp:confirm-one-phase>
```

3481

3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532

HAZARD

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

CONTRADICTION

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

SUPERIOR_STATE

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

INFERIOR_STATE

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



```
3533 <btpr:superior-identifier>...URI...</btpr:superior-identifier>
3534
3535 <btpr:inferior-identifier>...URI...</btpr:inferior-identifier>
3536 <btpr:status>active|inaccessible|unknown</btpr:status>
3537 <btpr:response-requested>>true|false</btpr:response-requested>
3538 <btpr:qualifiers> ?
3539 ...qualifiers...
3540 </btpr:qualifiers>
3541 </btpr:inferior-state>
```

3542
3543

REDIRECT

```
3544
3545
3546 <btpr:redirect id?>
3547 <btpr:target-additional-information> ?
3548 ...additional address information...
3549 </btpr:target-additional-information>
3550 <btpr:superior-identifier>...URI...</btpr:superior-identifier> ?
3551 <btpr:inferior-identifier>...URI...</btpr:inferior-identifier>
3552 <btpr:old-address> +
3553 ...address...
3554 </btpr:old-address>
3555 <btpr:new-address> +
3556 ...address...
3557 </btpr:new-address>
3558 <btpr:qualifiers> ?
3559 ...qualifiers...
3560 </btpr:qualifiers>
3561 </btpr:redirect>
```

3562
3563

BEGIN

```
3564
3565 <btpr:begin id?>
3566 <btpr:target-additional-information> ?
3567 ...additional address information...
3568 </btpr:target-additional-information>
3569 <btpr:reply-address> ?
3570 ...address...
3571 </btpr:reply-address>
3572 <btpr:transaction-type>cohesion|atom</btpr:transaction-type>
3573 <btpr:qualifiers> ?
3574 ...qualifiers...
3575 </btpr:qualifiers>
3576 </btpr:begin>
```

3577
3578

BEGUN

```
3579
3580
3581 <btpr:begun id?>
3582 <btpr:target-additional-information> ?
3583 ...additional address information...
```

```
3584 </btp:target-additional-information>
3585 <btp:decider-address> *
3586   ...address...
3587 </btp:decider-address>
3588 <btp:inferior-address> *
3589   ...address...
3590 </btp:inferior-address>
3591 <btp:transaction-identifier>...URI...</btp:transaction-
3592 identifier>
3593 <btp:qualifiers> ?
3594   ...qualifiers...
3595 </btp:qualifiers>
3596 </btp:begin>
```

3597
3598

PREPARE_INFERIORS

```
3600
3601 <btp:prepare-inferiors id?>
3602   <btp:target-additional-information> ?
3603     ...additional address information...
3604 </btp:target-additional-information>
3605 <btp:reply-address> ?
3606   ...address...
3607 </btp:reply-address>
3608 <btp:transaction-identifier>...URI...</btp:transaction-
3609 identifier>
3610 <btp:inferiors-list> ?
3611   <btp:inferior-handle>...URI...</btp:inferior-handle> +
3612 </btp:inferiors-list>
3613 <btp:qualifiers> ?
3614   ...qualifiers...
3615 </btp:qualifiers>
3616 </btp:prepare-inferiors>
```

3617
3618

CONFIRM_TRANSACTION

```
3620
3621 <btp:confirm-transaction id?>
3622   <btp:target-additional-information> ?
3623     ...additional address information...
3624 </btp:target-additional-information>
3625 <btp:reply-address> ?
3626   ...address...
3627 </btp:reply-address>
3628 <btp:transaction-identifier>...URI...</btp:transaction-
3629 identifier>
3630 <btp:inferiors-list> ?
3631   <btp:inferior-handle>...URI...</btp:inferior-handle> +
3632 </btp:inferiors-list>
3633 <btp:report-hazard>true|false</btp:report-hazard>
3634 <btp:qualifiers> ?
3635   ...qualifiers...
```

```
3636     </btp:qualifiers>
3637 </btp:confirm_transaction>
```

3638
3639

TRANSACTION_CONFIRMED

```
3641
3642 <btp:transaction-confirmed id?>
3643   <btp:target-additional-information> ?
3644     ...additional address information...
3645   </btp:target-additional-information>
3646
3647   <btp:transaction-identifier>...URI...</btp:transaction-
3648   identifier>
3649   <btp:qualifiers> ?
3650     ...qualifiers...
3651   </btp:qualifiers>
3652 </btp:transaction-confirmed>
```

3653
3654

CANCEL_TRANSACTION

```
3655
3656 <btp:cancel-transaction id?>
3657   <btp:target-additional-information> ?
3658     ...additional address information...
3659   </btp:target-additional-information>
3660   <btp:reply-address> ?
3661     ...address...
3662   </btp:reply-address>
3663   <btp:transaction-identifier>...URI...</btp:transaction-
3664   identifier>
3665   <btp:report-hazard>true|false</btp:report-hazard>
3666   <btp:qualifiers> ?
3667     ...qualifiers...
3668   </btp:qualifiers>
3669 </btp:cancel-transaction>
```

3670
3671

CANCEL_INFERIORS

```
3672
3673 <btp:cancel-inferiors id?>
3674   <btp:target-additional-information> ?
3675     ...additional address information...
3676   </btp:target-additional-information>
3677   <btp:reply-address> ?
3678     ...address...
3679   </btp:reply-address>
3680   <btp:transaction-identifier>...URI...</btp:transaction-
3681   identifier> ?
3682   <btp:inferiors-list>
3683     <btp:inferior-handle>...URI...</btp:inferior-handle> +
3684   </btp:inferiors-list>
3685   <btp:qualifiers> ?
```

3686

```
3687     ...qualifiers...
3688     </btp:qualifiers>
3689 </btp:cancel-inferiors>
```

3690
3691

TRANSACTION_CANCELLED

```
3692
3693
3694 <btp:transaction-cancelled id?>
3695   <btp:target-additional-information> ?
3696     ...additional address information...
3697   </btp:target-additional-information>
3698
3699   <btp:transaction-identifier>...URI...</btp:transaction-
3700 identifier>
3701   <btp:qualifiers> ?
3702     ...qualifiers...
3703   </btp:qualifiers>
3704 </btp:transaction-cancelled>
```

3705
3706

REQUEST_INFERIOR_STATUSES

```
3707
3708
3709 <btp:request-inferior-statuses id?>
3710   <btp:target-additional-information> ?
3711     ...additional address information...
3712   </btp:target-additional-information>
3713   <btp:reply-address> ?
3714     ...address...
3715   </btp:reply-address>
3716   <btp:target-identifier>...URI...</btp:target-identifier>
3717   <btp:inferiors-list> ?
3718     <btp:inferior-handle>...URI...</btp:inferior-handle> +
3719   </btp:inferiors-list>
3720   <btp:qualifiers> ?
3721     ...qualifiers...
3722   </btp:qualifiers>
3723 </btp:request-inferior-statuses>
```

3724
3725

INFERIOR_STATUSES

```
3726
3727
3728 <btp:inferior-statuses id?>
3729   <btp:target-additional-information> ?
3730     ...additional address information...
3731   </btp:target-additional-information>
3732
3733   <btp:responders-identifier>...URI...</btp:responders-identifier>
3734   <btp:status-list>
3735     <btp:status-item> +
3736       <btp:inferior-handle>...URI...</btp:inferior-handle>
3737       <btp:status>active|resigned|preparing|prepared|
```

```

3738         autonomously-confirmed|autonomously-cancelled|
3739         confirming|confirmed|cancelling|cancelled|
3740         cancel-contradiction|confirm-contradiction|
3741         hazard|invalid</btp:status>
3742     <btp:qualifiers> ?
3743         ...qualifiers...
3744     </btp:qualifiers>
3745 </btp:status-item>
3746 </btp:status-list>
3747 <btp:qualifiers> ?
3748     ...qualifiers...
3749 </btp:qualifiers>
3750 </btp:inferior-statuses>
3751

```

3752 **Standard qualifiers**

3753 The informal syntax for these messages assumes the namespace prefix “btpq” is associated
3754 with the URI “urn:oasis:names:tc:BTP:[1.0:qualifiers](#)”.

3756 **Transaction timelimit**

```

3757
3758 <btpq:transaction-timelimit>
3759     <btpq:timelimit>
3760         ...time in seconds...
3761     </btpq:timelimit>
3762 </btpq:transaction-timelimit>
3763

```

3764 **Inferior timeout**

```

3765 <btpq:inferior-timeout>
3766     <btpq:timeout>
3767         ...time in seconds...
3768     </btpq:timeout>
3769     <btpq:intended-decision>confirm|cancel</btpq:intended-decision>
3770 </btpq:inferior-timeout>
3771

```

3772 **Minimum inferior timeout**

```

3773 <btpq:minimum-inferior-timeout>
3774     <btpq:minimum-timeout>
3775         ...time in seconds...
3776     </btpq:minimum-timeout>
3777 </btpq:minimum-inferior-timeout>
3778

```

3779 **Inferior name**

```

3780 <btpq:inferior-name>
3781     <btpq:inferior-name>
3782         ...string...
3783     </btpq:inferior-name>
3784 </btpq:inferior-name>
3785

```

3786 **Compounding of Messages**

3787

3788 Relating BTP to one another, in a “group” is represented by containing them within the
3789 btp:related-group element, with the related messages as child elements. The processing for
3790 the group is defined in the section “Groups – combinations of related messages”. For example

3791
3792
3793
3794
3795
3796
3797
3798
3799

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

3800 If the rules for the group state that the “target-address” of the abstract message is omitted, the
3801 corresponding target-address-information element shall be absent in the message in the
3802 related-group. The carrier protocol binding specifies how a relation between application and
3803 BTP messages is represented.

3804
3805
3806
3807
3808

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):

3809
3810
3811
3812
3813
3814
3815

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

3815

3816 XML Schemas

3817

3818 XML schema for BTP messages

3819

```
3820 <?xml version="1.0"?>
```

```
3821 <schema
```

```
3822   xmlns="http://www.w3.org/2001/XMLSchema"
```

```
3823   targetNamespace="urn:oasis:names:tc:BTP:1.0:corexml"
```

```
3824   xmlns:btp="urn:oasis:names:tc:BTP:1.0:corexml"
```

```
3825   elementFormDefault="qualified">
```

3826

3827

```
3828   <!-- Qualifiers -->
```

3829

```
3830   <complexType name="qualifier-type">
```

```
3831     <simpleContent>
```

```
3832       <extension base="string">
```

```
3833         <attribute name="must-be-understood" type="boolean"/>
```

```
3834         <attribute name="to-be-propagated" type="boolean"/>
```

```
3835       </extension>
```

```
3836     </simpleContent>
```

```
3837   </complexType>
```

3838

```
3839   <element name="qualifier" type="btp:qualifier-type" abstract="true"/>
```

3840

```
3841   <element name="qualifiers">
```

```
3842     <complexType>
```

```
3843       <sequence>
```

```
3844         <element ref="btp:qualifier" maxOccurs="unbounded"/>
```

```
3845       </sequence>
```

```
3846     </complexType>
```

```
3847   </element>
```

3848

```
3849   <!-- example qualifier:
```

```
3850     <element name="some-qualifer" type="btp:qualifier-type"
```

```
3851 substitutionGroup="btp:qualifier"/>
```

```
3852   -->
```

3853

3854

```
3855   <!-- Message set data types -->
```

3856

```
3857   <simpleType name="identifier">
```

```
3858     <restriction base="anyURI" />
```

```
3859   </simpleType>
```

3860

```
3861   <simpleType name="additional-information">
```

```
3862     <restriction base="string" />
```

```
3863   </simpleType>
```

3864

```
3865   <complexType name="address">
```

```
3866     <sequence>
```

```

3867         <element name="binding-name" type="anyURI"/>
3868         <element name="binding-address" type="string"/>
3869         <element name="additional-information" type="btp:additional-
3870 information" minOccurs="0" />
3871     </sequence>
3872 </complexType>
3873
3874     <simpleType name="superior-type">
3875         <restriction base="string">
3876             <enumeration value="cohesion"/>
3877             <enumeration value="atom"/>
3878         </restriction>
3879     </simpleType>
3880
3881     <simpleType name="transaction-type">
3882         <restriction base="string">
3883             <enumeration value="cohesion"/>
3884             <enumeration value="atom"/>
3885         </restriction>
3886     </simpleType>
3887
3888
3889     <!-- Compounding -->
3890
3891     <element name="messages">
3892         <complexType>
3893             <sequence>
3894                 <element ref="btp:message" minOccurs="0"
3895 maxOccurs="unbounded"/>
3896             </sequence>
3897         </complexType>
3898     </element>
3899
3900     <element name="related-group" substitutionGroup="btp:message">
3901         <complexType>
3902             <sequence>
3903                 <element ref="btp:message" minOccurs="0"
3904 maxOccurs="unbounded"/>
3905             </sequence>
3906         </complexType>
3907     </element>
3908
3909
3910     <!-- Message set -->
3911
3912     <element name="message" abstract="true" />
3913
3914     <element name="context" substitutionGroup="btp:message">
3915         <complexType>
3916             <sequence>
3917                 <element name="superior-address" type="btp:address"
3918 maxOccurs="unbounded"/>
3919                 <element name="superior-identifier" type="btp:identifier"/>

```



```

3920         <element name="reply-address" type="btp:address"
3921 minOccurs="0"/>
3922         <element name="superior-type" type="btp:superior-type"/>
3923         <element ref="btp:qualifiers" minOccurs="0"/>
3924     </sequence>
3925     <attribute name="id" type="ID" use="optional"/>
3926 </complexType>
3927 </element>
3928
3929 <element name="context-reply" substitutionGroup="btp:message">
3930 <complexType>
3931 <sequence>
3932     <element name="target-additional-information"
3933 type="btp:additional-information" minOccurs="0"/>
3934     <element name="superior-identifier" type="btp:identifier"/>
3935     <element name="completion-status">
3936 <simpleType>
3937 <restriction base="string">
3938     <enumeration value="completed"/>
3939     <enumeration value="related"/>
3940     <enumeration value="repudiated"/>
3941 </restriction>
3942 </simpleType>
3943 </element>
3944     <element ref="btp:qualifiers" minOccurs="0"/>
3945 </sequence>
3946 <attribute name="id" type="ID"/>
3947 </complexType>
3948 </element>
3949
3950 <element name="request-status" substitutionGroup="btp:message">
3951 <complexType>
3952 <sequence>
3953     <element name="target-additional-information"
3954 type="btp:additional-information" minOccurs="0"/>
3955     <element name="reply-address" type="btp:address"
3956 minOccurs="0"/>
3957     <element name="target-identifier" type="btp:identifier"/>
3958     <element ref="btp:qualifiers" minOccurs="0"/>
3959 </sequence>
3960 <attribute name="id" type="ID"/>
3961 </complexType>
3962 </element>
3963
3964 <element name="status" substitutionGroup="btp:message">
3965 <complexType>
3966 <sequence>
3967     <element name="target-additional-information"
3968 type="btp:additional-information" minOccurs="0"/>
3969     <element name="responders-identifier"
3970 type="btp:identifier"/>
3971     <element name="status-value">
3972 <simpleType>

```

```

3973         <restriction base="string">
3974             <enumeration value="created" />
3975             <enumeration value="enrolling" />
3976             <enumeration value="active" />
3977             <enumeration value="resigning" />
3978             <enumeration value="resigned" />
3979             <enumeration value="preparing" />
3980             <enumeration value="prepared" />
3981             <enumeration value="confirming" />
3982             <enumeration value="confirmed" />
3983             <enumeration value="cancelling" />
3984             <enumeration value="cancelled" />
3985             <enumeration value="cancel-contradiction" />
3986             <enumeration value="confirm-contradiction" />
3987             <enumeration value="hazard" />
3988             <enumeration value="contradicted" />
3989             <enumeration value="unknown" />
3990             <enumeration value="inaccessible" />
3991         </restriction>
3992     </simpleType>
3993 </element>
3994     <element ref="btp:qualifiers" minOccurs="0" />
3995 </sequence>
3996     <attribute name="id" type="ID" />
3997 </complexType>
3998 </element>
3999
4000 <element name="fault" substitutionGroup="btp:message">
4001     <complexType>
4002         <sequence>
4003             <element name="target-additional-information"
4004 type="btp:additional-information" minOccurs="0" />
4005             <element name="superior-identifier" type="btp:identifier"
4006 minOccurs="0" />
4007             <element name="inferior-identifier" type="btp:identifier"
4008 minOccurs="0" />
4009             <element name="fault-type">
4010                 <simpleType>
4011                     <restriction base="string">
4012                         <enumeration value="communication-failure" />
4013                         <enumeration value="duplicate-inferior" />
4014                         <enumeration value="general" />
4015                         <enumeration value="invalid-decider" />
4016                         <enumeration value="invalid-inferior" />
4017                         <enumeration value="invalid-superior" />
4018                         <enumeration value="status-refused" />
4019                         <enumeration value="invalid-terminator" />
4020                         <enumeration value="unknown-parameter" />
4021                         <enumeration value="unknown-transaction" />
4022                         <enumeration value="unsupported-qualifier" />
4023                         <enumeration value="wrong-state" />
4024                     </restriction>
4025                 </simpleType>

```

```

4026         </element>
4027         <element name="fault-data" type="anyType" minOccurs="0"/>
4028         <element ref="btp:qualifiers" minOccurs="0"/>
4029     </sequence>
4030     <attribute name="id" type="ID"/>
4031 </complexType>
4032 </element>
4033
4034 <element name="enrol" substitutionGroup="btp:message">
4035     <complexType>
4036         <sequence>
4037             <element name="target-additional-information"
4038 type="btp:additional-information" minOccurs="0"/>
4039             <element name="superior-identifier" type="btp:identifier"/>
4040             <element name="response-requested" type="boolean"/>
4041             <element name="reply-address" type="btp:address"
4042 minOccurs="0"/>
4043             <element name="inferior-address" type="btp:address"
4044 minOccurs="1" maxOccurs="unbounded"/>
4045             <element name="inferior-identifier" type="btp:identifier"/>
4046             <element ref="btp:qualifiers" minOccurs="0"/>
4047         </sequence>
4048         <attribute name="id" type="ID"/>
4049     </complexType>
4050 </element>
4051
4052
4053 <element name="enrolled" substitutionGroup="btp:message">
4054     <complexType>
4055         <sequence>
4056             <element name="target-additional-information"
4057 type="btp:additional-information" minOccurs="0"/>
4058             <element name="inferior-identifier" type="btp:identifier"/>
4059             <element ref="btp:qualifiers" minOccurs="0"/>
4060         </sequence>
4061         <attribute name="id" type="ID"/>
4062     </complexType>
4063 </element>
4064
4065 <element name="resign" substitutionGroup="btp:message">
4066     <complexType>
4067         <sequence>
4068             <element name="target-additional-information"
4069 type="btp:additional-information" minOccurs="0"/>
4070             <element name="superior-identifier" type="btp:identifier"/>
4071             <element name="inferior-identifier" type="btp:identifier"/>
4072             <element name="response-requested" type="boolean"/>
4073             <element ref="btp:qualifiers" minOccurs="0"/>
4074         </sequence>
4075         <attribute name="id" type="ID"/>
4076     </complexType>
4077 </element>
4078

```

```

4079     <element name="resigned" substitutionGroup="btp:message">
4080         <complexType>
4081             <sequence>
4082                 <element name="target-additional-information"
4083 type="btp:additional-information" minOccurs="0"/>
4084                 <element name="inferior-identifier" type="btp:identifier"/>
4085                 <element ref="btp:qualifiers" minOccurs="0"/>
4086             </sequence>
4087             <attribute name="id" type="ID"/>
4088         </complexType>
4089     </element>
4090
4091     <element name="prepare" substitutionGroup="btp:message">
4092         <complexType>
4093             <sequence>
4094                 <element name="target-additional-information"
4095 type="btp:additional-information" minOccurs="0"/>
4096                 <element name="inferior-identifier" type="btp:identifier"/>
4097                 <element ref="btp:qualifiers" minOccurs="0"/>
4098             </sequence>
4099             <attribute name="id" type="ID"/>
4100         </complexType>
4101     </element>
4102
4103     <element name="prepared" substitutionGroup="btp:message">
4104         <complexType>
4105             <sequence>
4106                 <element name="target-additional-information"
4107 type="btp:additional-information" minOccurs="0"/>
4108                 <element name="superior-identifier" type="btp:identifier"/>
4109                 <element name="inferior-identifier" type="btp:identifier"/>
4110                 <element name="default-is-cancel" type="boolean"/>
4111                 <element ref="btp:qualifiers" minOccurs="0"/>
4112             </sequence>
4113             <attribute name="id" type="ID"/>
4114         </complexType>
4115     </element>
4116
4117     <element name="confirm" substitutionGroup="btp:message">
4118         <complexType>
4119             <sequence>
4120                 <element name="target-additional-information"
4121 type="btp:additional-information" minOccurs="0"/>
4122                 <element name="inferior-identifier" type="btp:identifier"/>
4123                 <element ref="btp:qualifiers" minOccurs="0"/>
4124             </sequence>
4125             <attribute name="id" type="ID"/>
4126         </complexType>
4127     </element>
4128
4129     <element name="confirmed" substitutionGroup="btp:message">
4130         <complexType>
4131             <sequence>

```

```

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

```

```

4185     <complexType>
4186         <sequence>
4187             <element name="target-additional-information"
4188 type="btp:additional-information" minOccurs="0"/>
4189             <element name="superior-identifier" type="btp:identifier"/>
4190             <element name="inferior-identifier" type="btp:identifier"/>
4191             <element name="level">
4192                 <simpleType>
4193                     <restriction base="string">
4194                         <enumeration value="mixed"/>
4195                         <enumeration value="possible"/>
4196                     </restriction>
4197                 </simpleType>
4198             </element>
4199             <element ref="btp:qualifiers" minOccurs="0"/>
4200         </sequence>
4201         <attribute name="id" type="ID"/>
4202     </complexType>
4203 </element>
4204
4205     <element name="contradiction" substitutionGroup="btp:message">
4206         <complexType>
4207             <sequence>
4208                 <element name="target-additional-information"
4209 type="btp:additional-information" minOccurs="0"/>
4210                 <element name="inferior-identifier" type="btp:identifier"/>
4211                 <element ref="btp:qualifiers" minOccurs="0"/>
4212             </sequence>
4213             <attribute name="id" type="ID"/>
4214         </complexType>
4215     </element>
4216
4217     <element name="superior-state" substitutionGroup="btp:message">
4218         <complexType>
4219             <sequence>
4220                 <element name="target-additional-information"
4221 type="btp:additional-information" minOccurs="0"/>
4222                 <element name="inferior-identifier" type="btp:identifier"/>
4223                 <element name="status">
4224                     <simpleType>
4225                         <restriction base="string">
4226                             <enumeration value="active"/>
4227                             <enumeration value="prepared-received"/>
4228                             <enumeration value="inaccessible"/>
4229                             <enumeration value="unknown"/>
4230                         </restriction>
4231                     </simpleType>
4232                 </element>
4233                 <element name="response-requested" type="boolean"/>
4234                 <element ref="btp:qualifiers" minOccurs="0"/>
4235             </sequence>
4236             <attribute name="id" type="ID"/>
4237         </complexType>

```

```

4238     </element>
4239
4240     <element name="inferior-state" substitutionGroup="btp:message">
4241         <complexType>
4242             <sequence>
4243                 <element name="target-additional-information"
4244 type="btp:additional-information" minOccurs="0"/>
4245                 <element name="superior-identifier" type="btp:identifier"/>
4246                 <element name="inferior-identifier" type="btp:identifier"/>
4247                 <element name="status">
4248                     <simpleType>
4249                         <restriction base="string">
4250                             <enumeration value="active"/>
4251                             <enumeration value="inaccessible"/>
4252                             <enumeration value="unknown"/>
4253                         </restriction>
4254                     </simpleType>
4255                 </element>
4256                 <element name="response-requested" type="boolean"/>
4257                 <element ref="btp:qualifiers" minOccurs="0"/>
4258             </sequence>
4259             <attribute name="id" type="ID"/>
4260         </complexType>
4261     </element>
4262
4263     <element name="redirect" substitutionGroup="btp:message">
4264         <complexType>
4265             <sequence>
4266                 <element name="target-additional-information"
4267 type="btp:additional-information" minOccurs="0"/>
4268                 <element name="superior-identifier" type="btp:identifier"
4269 minOccurs="0"/>
4270                 <element name="inferior-identifier" type="btp:identifier"
4271 />
4272                 <element name="old-address" type="btp:address"
4273 maxOccurs="unbounded"/>
4274                 <element name="new-address" type="btp:address"
4275 maxOccurs="unbounded"/>
4276                 <element ref="btp:qualifiers" minOccurs="0"/>
4277             </sequence>
4278             <attribute name="id" type="ID"/>
4279         </complexType>
4280     </element>
4281
4282
4283     <element name="begin" substitutionGroup="btp:message">
4284         <complexType>
4285             <sequence>
4286                 <element name="target-additional-information"
4287 type="btp:additional-information" minOccurs="0"/>
4288                 <element name="reply-address" type="btp:address"
4289 minOccurs="0"/>
4290                 <element name="transaction-type" type="btp:superior-type"/>

```

```

4291         <element ref="btp:qualifiers" minOccurs="0"/>
4292     </sequence>
4293     <attribute name="id" type="ID"/>
4294 </complexType>
4295 </element>
4296
4297     <element name="begun" substitutionGroup="btp:message">
4298         <complexType>
4299             <sequence>
4300                 <element name="target-additional-information"
4301 type="btp:additional-information" minOccurs="0"/>
4302                 <element name="decider-address" type="btp:address"
4303 minOccurs="0" maxOccurs="unbounded"/>
4304                 <element name="transaction-identifier"
4305 type="btp:identifier" minOccurs="0"/>
4306                 <element name="inferior-handle" type="btp:identifier"
4307 minOccurs="0"/>
4308                 <element name="inferior-address" type="btp:address"
4309 minOccurs="0" maxOccurs="unbounded"/>
4310                 <element ref="btp:qualifiers" minOccurs="0"/>
4311             </sequence>
4312             <attribute name="id" type="ID"/>
4313         </complexType>
4314     </element>
4315
4316     <element name="prepare-inferiors" substitutionGroup="btp:message">
4317         <complexType>
4318             <sequence>
4319                 <element name="target-additional-information"
4320 type="btp:additional-information" minOccurs="0"/>
4321                 <element name="reply-address" type="btp:address"
4322 minOccurs="0"/>
4323                 <element name="transaction-identifier"
4324 type="btp:identifier"/>
4325                 <element name="inferiors-list" minOccurs="0">
4326                     <complexType>
4327                         <sequence>
4328                             <element name="inferior-handle"
4329 type="btp:identifier" maxOccurs="unbounded"/>
4330                         </sequence>
4331                     </complexType>
4332                 </element>
4333                 <element ref="btp:qualifiers" minOccurs="0"/>
4334             </sequence>
4335             <attribute name="id" type="ID"/>
4336         </complexType>
4337     </element>
4338
4339     <element name="confirm-transaction" substitutionGroup="btp:message">
4340         <complexType>
4341             <sequence>
4342                 <element name="target-additional-information"
4343 type="btp:additional-information" minOccurs="0"/>

```



```

4344         <element name="reply-address" type="btp:address"
4345 minOccurs="0"/>
4346         <element name="transaction-identifier"
4347 type="btp:identifier"/>
4348         <element name="inferiors-list" minOccurs="0">
4349             <complexType>
4350                 <sequence>
4351                     <element name="inferior-handle"
4352 type="btp:identifier" maxOccurs="unbounded"/>
4353                 </sequence>
4354             </complexType>
4355         </element>
4356         <element name="report-hazard" type="boolean"/>
4357         <element ref="btp:qualifiers" minOccurs="0"/>
4358     </sequence>
4359     <attribute name="id" type="ID"/>
4360 </complexType>
4361 </element>
4362
4363     <element name="transaction-confirmed" substitutionGroup="btp:message">
4364         <complexType>
4365             <sequence>
4366                 <element name="target-additional-information"
4367 type="btp:additional-information" minOccurs="0"/>
4368                 <element name="transaction-identifier"
4369 type="btp:identifier"/>
4370                 <element ref="btp:qualifiers" minOccurs="0"/>
4371             </sequence>
4372             <attribute name="id" type="ID"/>
4373         </complexType>
4374     </element>
4375
4376     <element name="cancel-transaction" substitutionGroup="btp:message">
4377         <complexType>
4378             <sequence>
4379                 <element name="target-additional-information"
4380 type="btp:additional-information" minOccurs="0"/>
4381                 <element name="reply-address" type="btp:address"
4382 minOccurs="0"/>
4383                 <element name="transaction-identifier"
4384 type="btp:identifier"/>
4385                 <element name="report-hazard" type="boolean"/>
4386                 <element ref="btp:qualifiers" minOccurs="0"/>
4387             </sequence>
4388             <attribute name="id" type="ID"/>
4389         </complexType>
4390     </element>
4391
4392     <element name="cancel-inferiors" substitutionGroup="btp:message">
4393         <complexType>
4394             <sequence>
4395                 <element name="target-additional-information"
4396 type="btp:additional-information" minOccurs="0"/>

```

```

4397         <element name="reply-address" type="btp:address"
4398 minOccurs="0"/>
4399         <element name="transaction-identifier"
4400 type="btp:identifier" minOccurs="0"/>
4401         <element name="inferiors-list">
4402             <complexType>
4403                 <sequence>
4404                     <element name="inferior-handle"
4405 type="btp:identifier" maxOccurs="unbounded"/>
4406                 </sequence>
4407             </complexType>
4408         </element>
4409         <element ref="btp:qualifiers" minOccurs="0"/>
4410     </sequence>
4411     <attribute name="id" type="ID"/>
4412 </complexType>
4413 </element>
4414
4415     <element name="transaction-cancelled" substitutionGroup="btp:message">
4416         <complexType>
4417             <sequence>
4418                 <element name="target-additional-information"
4419 type="btp:additional-information" minOccurs="0"/>
4420                 <element name="transaction-identifier"
4421 type="btp:identifier"/>
4422                 <element ref="btp:qualifiers" minOccurs="0"/>
4423             </sequence>
4424             <attribute name="id" type="ID"/>
4425         </complexType>
4426     </element>
4427
4428     <element name="request-inferior-statuses"
4429 substitutionGroup="btp:message">
4430         <complexType>
4431             <sequence>
4432                 <element name="target-additional-information"
4433 type="btp:additional-information" minOccurs="0"/>
4434                 <element name="reply-address" type="btp:address"
4435 minOccurs="0"/>
4436                 <element name="target-identifier" type="btp:identifier"/>
4437                 <element name="inferiors-list" minOccurs="0">
4438                     <complexType>
4439                         <sequence>
4440                             <element name="inferior-handle"
4441 type="btp:identifier" maxOccurs="unbounded"/>
4442                         </sequence>
4443                     </complexType>
4444                 </element>
4445                 <element ref="btp:qualifiers" minOccurs="0"/>
4446             </sequence>
4447             <attribute name="id" type="ID"/>
4448         </complexType>
4449     </element>

```

```

4450
4451     <element name="inferior-statuses" substitutionGroup="btp:message">
4452         <complexType>
4453             <sequence>
4454                 <element name="target-additional-information"
4455 type="btp:additional-information" minOccurs="0"/>
4456                 <element name="responders-identifier"
4457 type="btp:identifier"/>
4458                 <element name="status-list">
4459                     <complexType>
4460                         <sequence>
4461                             <element name="status-item" maxOccurs="unbounded">
4462                                 <complexType>
4463                                     <sequence>
4464                                         <element name="inferior-handle"
4465 type="btp:identifier"/>
4466                                         <element name="status">
4467                                             <simpleType>
4468                                                 <restriction base="string">
4469                                                     <enumeration value="active"/>
4470                                                     <enumeration value="resigned"/>
4471                                                     <enumeration value="preparing"/>
4472                                                     <enumeration value="prepared"/>
4473                                                     <enumeration value="autonomously-confirmed"/>
4474                                                     <enumeration value="autonomously-cancelled"/>
4475                                                     <enumeration value="confirming"/>
4476                                                     <enumeration value="confirmed"/>
4477                                                     <enumeration value="cancelling"/>
4478                                                     <enumeration value="cancelled"/>
4479                                                     <enumeration value="cancel-contradiction"/>
4480                                                     <enumeration value="confirm-contradiction"/>
4481                                                     <enumeration value="hazard"/>
4482                                                     <enumeration value="invalid"/>
4483                                                 </restriction>
4484                                             </simpleType>
4485                                         </element>
4486                                         <element ref="btp:qualifiers" minOccurs="0"/>
4487                                     </sequence>
4488                                 </complexType>
4489                             </element>
4490                         </sequence>
4491                     </complexType>
4492                 </element>
4493                 <element ref="btp:qualifiers" minOccurs="0"/>
4494             </sequence>
4495             <attribute name="id" type="ID"/>
4496         </complexType>
4497     </element>
4498
4499
4500 </schema>
4501

```

XML schema for standard qualifiers

4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553

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

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

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

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

```
4554         <element name="minimum-timeout"  
4555 type="nonNegativeInteger"/>  
4556         </sequence>  
4557     </extension>  
4558 </complexContent>  
4559 </complexType>  
4560 </element>  
4561  
4562 <element name="inferior-name" substitutionGroup="btp:qualifier">  
4563     <complexType>  
4564         <complexContent>  
4565             <extension base="btp:qualifier-type">  
4566                 <sequence>  
4567                     <element name="inferior-name" type="string"/>  
4568                 </sequence>  
4569             </extension>  
4570         </complexContent>  
4571     </complexType>  
4572 </element>  
4573  
4574 </schema>  
4575
```

4575
4576

4577 **Carrier Protocol Bindings**

4578

4579 The notion of bindings is introduced to act as the glue between the BTP messages and an
4580 underlying transport. A binding specification must define various particulars of how the BTP
4581 messages are carried and some aspects of how the related application messages are carried.
4582 This document specifies two bindings: a SOAP binding and a SOAP + Attachments binding.
4583 However, other bindings could be specified by the Oasis BTP technical committee or by a
4584 third party. For example, in the future a binding might exist to put a BTP message directly on
4585 top of HTTP without the use of SOAP, or a closed community could define their own
4586 binding. To ensure that such specifications are complete, the Binding Proforma defines the
4587 information that must be included in a binding specification.
4588

4589 **Carrier Protocol Binding Proforma**

4590

4591 A BTP carrier binding specification should provide the following information:
4592

4593 **Binding name:** A name for the binding, as used in the “binding name” field of BTP
4594 addresses (and available for declaring the capabilities of an implementation). Binding
4595 specified in this document, and future revisions of this document have binding names that are
4596 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).
4597 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in
4598 this document use numbers to identify the version of the binding, not the version(s) of the
4599 carrier protocol.

4600

4601 **Binding address format:** This section states the format of the “binding address” field of a
4602 BTP address for this binding. For many bindings, this will be a URL of some kind; for other
4603 bindings it may be some other form
4604

4605

4606 **BTP message representation:** This section will define how BTP messages are represented.
4607 For many bindings, the BTP message syntax will be as specified in the XML schema defined
4608 in this document, and the normal string encoding of that XML will be used.

4609

4610 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that
4611 are not related to application messages are sent in either direction between Superior and
4612 Inferior. (i.e. those messages sent directly between BTP actors). This mapping need not be
4613 symmetric (i.e. Superior to Inferior may differ to some degree to Inferior to Superior). The
4614 mapping may define particular rules for particular BTP messages, or messages with particular
4615 parameter values (e.g. the FAULT message with “fault-type” “CommunicationFailure” will
4616 typically not be sent as a BTP message). The mapping states any constraints or requirements
4617 on which BTP may or must be bundled together by compounding.

4618

4619 **Mapping for BTP messages related to application messages:** This section will define how
4620 BTP messages that are related to application messages are sent. A binding specification may
defer details of this to a particular application (e.g. a mapping specification could just say

4621 “the CONTEXT may be carried as a parameter of an application invocation”). Alternatively,
4622 the binding may specify a general method that represents the relationship between application
4623 and BTP messages.

4624
4625 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly
4626 but are treated as implicit in [carrier-protocol mechanisms](#), application messages or other BTP
4627 messages. This may depend on particular parameter values of the BTP messages or the
4628 application messages.

4629
4630 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
4631 protocol and of BTP shall be defined. This may include definition of which carrier protocol
4632 exceptions are equivalent to a FAULT/communication-failure message.

4633
4634 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.
4635 If BTP addresses with different bindings are be considered to match (for purposes of
4636 identifying the peer Superior/Inferior and redirection), this should be specified here.

4637
4638 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are
4639 imposed by use of this binding should be listed. This would include limitations on which
4640 messages can be sent, which event sequences are supported and restrictions on parameter
4641 values. Such limitations may reduce the usefulness of an implementation, but may be
4642 appropriate in certain environments.

4643
4644 **Other:** Other features of the binding, especially any that will potentially affect interoperation
4645 should be specified here. This may include restrictions or requirements on the use or support
4646 of optional carrier parameters or mechanisms.

4647

4648 **Bindings for request/response carrier protocols**

4649

4650 BTP does not generally follow request/response pattern. In particular, on the outcome
4651 relationship either side may initiate a message – this is an essential part of the presume-abort
4652 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
4653 messages, especially in the control relationship, that do have a request/response pattern.
4654 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The
4655 specification of a binding specification to a request/response carrier protocol needs to state
4656 what rules apply – which messages can be carried by requests, which by responses. The
4657 simplest rule is to send all BTP messages on requests, and let the carrier responses travel back
4658 empty. This would be inefficient in use of network resources, and possibly inconvenient
4659 when used for the BTP request/response pairs.

4660

4661 This section defines a set of rules that allow more efficient use of the carrier, while allowing
4662 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the
4663 carrier response. These rules are specified in this section to enable binding specifications to
4664 reference them, without requiring each binding specification to repeat similar information.

4665

4666 A binding to a request/response carrier is not required to use these rules. It may define other
4667 rules.

4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714

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 “~~superior-address-as-superior~~” and “~~inferior-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.

- 4715 e) A BTP actor that receives a carrier protocol request carrying BTP messages that
4716 do have a “reply-address”, or which initiate processing that produces BTP
4717 messages whose target binding address matches the origin of the request, **may**
4718 freely choose whether to use the carrier protocol response for the replies, or to
4719 send back an “empty carrier protocol response”, and send the BTP replies in a
4720 separately initiated carrier protocol request. The characteristics of an “empty
4721 carrier protocol response” shall be stated in the particular binding specification.
4722
- 4723 f) A BTP actor that sends BTP messages on a carrier protocol request **must** be able
4724 to accept returning BTP messages on the corresponding carrier protocol response
4725 and, if the actor has offered an address on which it will receive carrier requests,
4726 must be able to accept “replying” BTP messages on a separate carrier protocol
4727 request.
4728

4729 SOAP Binding

4730 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
4731 specification, using the SOAP literal messaging style conventions. If no application message
4732 is sent at the same time, the BTP messages are contained within the SOAP Body element. If
4733 application messages are sent, the BTP messages are contained in the SOAP Header element.
4734

4735 **Binding name:** soap-http-1
4736

4737 **Binding address format:** shall be a URL, of type HTTP.
4738

4739 **BTP message representation:** The string representation of the XML, as specified in the
4740 XML schema defined in this document shall be used. The BTP XML messages are embedded
4741 in the SOAP message without the use of any specific encoding rules (literal style SOAP
4742 message); hence the encodingStyle attribute need not be set or can be set to an empty string.
4743

4744 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be
4745 used.
4746

4747 BTP messages sent on an HTTP request or HTTP response which is not carrying an
4748 application message, the messages are contained in a single btp:messages element which is
4749 the immediate child element of the SOAP Body element.
4750

4751 An “empty carrier protocol response” sent after receiving an HTTP request containing a
4752 btp:messages element in the SOAP Body and the implementation BTP actor chooses just to
4753 reply at the lower level (and when the request/response exploitation rules allow an empty
4754 carrier protocol response), shall be any of:
4755

- 4756 a) an empty HTTP response
- 4757 b) an HTTP response containing an empty SOAP Envelope
- 4758 c) an HTTP response containing a SOAP Envelope containing a single, empty
4759 btp:messages element.
4760

4761 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they
4762 have no effect on the BTP sequence (other than indicating that the earlier sending did not
4763 cause a communication failure.)
4764

4765

4766

4767

4768

4769

4770

4771

4772

4773

4774

4775

4776

4777

4778

4779

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

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

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

4780

4781

4782

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

4783

4784

4785

4786

4787

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

4788

4789

4790

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

4791

4792

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

4793

4794

4795

4796

4797

4798

4799

4800

4801

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

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

4802 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4803 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
4804 http-1” if the binding address and additional information fields match.

4805
4806 **Limitations on BTP use:** None

4807
4808 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
4809 attribute. The SOAPAction HTTP header is left to be application specific when there are
4810 application messages in the SOAP Body, as an already existing web service that is being
4811 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
4812 header shall be omitted when the SOAP message carries only BTP messages in the SOAP
4813 Body.

4814
4815 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP
4816 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to
4817 determine whether any enrolments are necessary and replies with CONTEXT_REPLY as
4818 appropriate. The sender of the CONTEXT (and related application message) can use this to
4819 ensure that the application work is performed as part of the business transaction, assuming the
4820 receiver’s SOAP implementation supports the mustUnderstand attribute. If mustUnderstand if
4821 false, a receiver can ignore the CONTEXT (if BTP is not supported there), and no
4822 CONTEXT_REPLY will be returned. It is a local option on the sender (client) side whether
4823 the absence of a CONTEXT_REPLY is assumed to be equivalent to aCONTEXT_REPLY/ok
4824 (and the business transaction allowed to proceed to confirmation).

4825
4826 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to
4827 enforce these requirements.

4828 **Example scenario using SOAP binding**

4829
4830 The example below shows an application request with CONTEXT message sent from
4831 client.example.com (which includes the Superior) to services.example.com (Service).

```
4832  
4833  
4834 <soap:Envelope  
4835   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
4836   soap:encodingStyle="">  
4837  
4838   <soap:Header>  
4839  
4840     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core.xml">|  
4841       <btp:context superior-type="atom">  
4842         <btp:superior-address>  
4843           <btp:binding>soap-http-1</btp:binding>  
4844           <btp:binding-  
4845 address>http://client.example.com/soaphandler</btp:binding-  
4846 address>  
4847           <btp:additional-information>btpengine</btp:additional-  
4848 information>  
4849           </btp:superior-address>
```

```

4850         <btp:superior-
4851 identifier>http://example.com/1001</btp:superior-identifier>
4852         <btp:qualifiers>
4853             <btpq:transaction-timelimit
4854 xmlns:btpq="urn:oasis:names:tc:BTP:1.0:qualifiers"><btpq:timelimit
4855 >1800</btpq:timelimit></btpq:transaction-timelimit>
4856             </btp:qualifiers>
4857         </btp:context>
4858     </btp:messages>
4859
4860 </soap:Header>
4861
4862 <soap:Body>
4863
4864     <ns1:orderGoods
4865 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4866         <custID>ABC8329045</custID>
4867         <itemID>224352</itemID>
4868         <quantity>5</quantity>
4869     </ns1:orderGoods>
4870
4871 </soap:Body>
4872
4873 </soap:Envelope>
4874
4875

```

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

```

4883 <soap:Envelope
4884     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
4885     soap:encodingStyle="">
4886
4887     <soap:Header>
4888     </soap:Header>
4889
4890     <soap:Body>
4891
4892         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core" >|
4893             <btp:related-group>
4894                 <btp:context-reply>
4895                     <btp:target-additional-information>btpengine</btp:target-
4896 additional-information>
4897                     <btp:superior-
4898 identifier>http://example.com/1001</btp:superior-identifier>
4899                     <completion-status>related</completion-status>
4900                 </btp:context-reply>
4901

```

```
4902         <btpe:enrol response-requested="false">
4903             <btpe:target-additional-
4904 information>btpeengine</btpe:target-additional-information>
4905             <btpe:superior-
4906 identifier>http://example.com/1001</btpe:superior-identifier>
4907             <btpe:inferior-address>
4908                 <btpe:binding>soap-http-1</btpe:binding>
4909                 <btpe:binding-address>
4910                     http://services.example.com/soaphandler
4911                 </btpe:binding-address>
4912             </btpe:inferior-address>
4913             <btpe:inferior-identifier>
4914                 http://example.com/AAAB
4915             </btpe:inferior-identifier>
4916         </btpe:enrol>
4917     </btpe:related-group>
4918 </btpe:messages>
4919 </btpe:Body>
4920 </btpe:Envelope>
```

4927 SOAP + Attachments Binding

4928 This binding describes how BTP messages will be carried using SOAP as in the [SOAP](#)
4929 [Messages with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-
4930 http-1. The two bindings only differ when application messages are sent.

4931 **Binding name:** soap-attachments-http-1

4932 **Binding address format:** as for soap-http-1

4933 **BTP message representation:** As for soap-http-1

4934 **Mapping for BTP messages (unrelated):** As for “soap-http-1”, except the SOAP Envelope
4935 containing the SOAP Body containing the BTP messages shall be in a MIME body part, as
4936 specified in [SOAP Messages with Attachments](#) specification. If an application message is
4937 being sent at the same time, the mapping for related messages for this binding shall be used,
4938 as if the BTP messages were related to the application message(s).

4939 **Mapping for BTP messages related to application messages:** MIME packaging shall be
4940 used. One of the MIME multipart/related parts shall contain a SOAP Envelope, whose SOAP
4941 Headers element shall contain precisely one btpe:messages element, containing any BTP
4942 messages. Any BTP CONTEXT in the btpe:messages is considered to be related to the
4943 application message(s) in the SOAP Body, and to also any of the MIME parts referenced
4944 from the SOAP Body (using the “href” attribute).

4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
5001

Implicit messages: As for soap-http-1.

Faults: As for soap-http-1.

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

Limitations on BTP use: None

Other: As for soap-http-1

Example using SOAP + Attachments binding

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

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

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

  <soap:Header>

    <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core+xml" >|
      <btp:context superior-type="atom">
        <btp:superior-address>
          <btp:binding>soap-http-1</btp:binding>
          <btp:binding-address>
            http://client.example.com/soaphandler
          </btp:binding-address>
          </btp:superior-address>
          <btp:superior-
            identifier>http://example.com/1001</btp:superior-identifier>
          </btp:context>
        </btp:messages>

      </soap:Header>

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

  </soap:Envelope>
```

5002
5003
5004
5005
5006
5007
5008
5009
5010
5011
5012
5013
5014
5015
5016
5017

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

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

--MIME_boundary--
```

5018 **Conformance**

5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029

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
--------------------	----------------------

5030
5031
5032
5033
5034

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

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

5035
5036
5037
5038
5039

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

5040 sends ENROL with the same inferior address and with the “reply-address” absent (because
5041 the Inferior in all transactions are dealt with by the same addressable entity), must not assume
5042 that the same is true of received ENROLs)
5043
5044

5044 Part 3. Appendices

5045

5046 *The glossary is the subject of issue 4*

5047

5048 **A. Glossary**

5049

Message	A datum which is produced and then consumed.
Sender	The producer of a message.
Receiver	The consumer of a message.
Transmission	The passage of a message from a sender to a receiver.
Endpoint	A sender or receiver.
Address	An identifier for an endpoint.
Peer	The other party in a two-party relationship, as in Superior to Inferior, or Sender to Receiver
Carrier Protocol	A protocol which defines how transmissions occur.
Carrier Protocol Address (CPA)	The address of an endpoint for a particular carrier protocol.
Business Transaction Protocol Address (BTPA)	A compound address consisting of a mandatory <i>carrier protocol address</i> and an optional opaque suffix. <i>PRF - suffix ? I've used "additional information"</i>
Actor	An entity which executes procedures, a software agent.
Application	An actor which uses the Business Transaction Protocol.
Application Message	A message produced by an application and consumed by an application.

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

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

Coordinator

An actor which decides the outcome of a single atom, and has a lifetime which is coincident with that of the atom. A coordinator can issue instructions to a participant to prepare, cancel and confirm. These instructions take the form of BTP messages. A coordinator is identified by its atom's atom identifier. A coordinator must also have a BTPA to which participants can send BTP messages.

~~superior-address-as-superior~~

The address used to communicate with an actor playing the role of an Superior

~~Composer-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.

~~Inferior-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 ~~superior-address-as-superior~~ of a given Superior.

Identity-as-Inferior

The combination of inferior-identifier and ~~inferior-address-as-inferior~~ of a given Inferior.

5050