

2 Business Transaction Protocol

3 An OASIS Committee Specification

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

5 Version 1.0 [0.9.2.4]

6 DD Mmm 2002 [3 April 2002 16:19]

7

8

9

10

<i>Working Draft 0.9</i>	24 October 2001
<i>Working Draft 0.9.0.1 – minor editorials issues applied</i>	16 November 2001
<i>Working Draft 0.9.0.2 – issue resolutions balloting to 10 Dec 2001</i>	4 December 2001
<i>Working Draft 0.9.0.3 – possible solution to msging issues</i>	11 December 2001
<i>Working Draft 0.9.0.4 – issue 79 solution, revise msging issues</i>	12 January 2002
<i>Working Draft 0.9.1 – includes all issues agreed 16 Jan 2002, and 82 (deferred)</i>	18 January 2002
<i>Working Draft 0.9.1.1 – format changes and proposed soln 77,78, 17.</i>	27 January 2002
<i>Working Draft 0.9.1.2 – xml changes, new schema, and issue 74</i>	30 January 2002
<i>Working Draft 0.9.1.3 – corrections, issue 30, state table – 81, 104</i>	8 February 2002
<i>Working Draft 0.9.2 – all issues as agreed 13 February 2002</i>	13 February 2002
<i>Working Draft 0.9.2.1 – issues 2, 3, 15, 19, 50, 67, 95</i>	26 February 2002
<i>Working Draft 0.9.2.2 – as accepted 27 Feb 2002+ corrections, issues 29, 60, 97, 99</i>	12 March 2002
<i>Working Draft 0.9.2.3 – 0.9.2.2 and issue 106, 96, 98</i>	18 March 2002
<i>Working Draft 0.9.2.4 – as accepted 27 Mar 2002 + 61, 100, 109, inclusion of model</i>	3 April 2002

11

12 [Change marks relative to 0.9.2.3 all accepted](#)

13

13 Copyright and related notices

14
15 Copyright © The Organization for the Advancement of Structured Information Standards
16 (OASIS), 2002. 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

Acknowledgements

The members of the OASIS Business Transactions Technical Committee contributed to the development of this specification. The following were members of the committee for at least part of the time from July 2001 until the agreement of the specification are listed below. Some TC members changed their affiliation to OASIS members, but remained members of the TC:

Mike Abbott	CodeMetamorphosis
Alex Berson	Entrust, Inc
Geoff Brown	Oracle
Doug Bunting	Sun Microsystems
Fred Carter	Sun Microsystems; individual
Alex Ceponkus	Bowstreet Inc.; individual
Pyounguk Cho	Iona
Victor Corrales	Hewlett-Packard Co.
Bill Cox	BEA Systems, Inc.
Sanjay Dalal	BEA Systems, Inc.
Alan Davies	SeeBeyond Inc.
Hatem El-Sebaaly	IPNet
Ed Felt	BEA Systems, Inc.
Tony Fletcher	Choreology Ltd
Bill Flood	Sybase
Peter Furniss	Choreology Ltd
Alastair Green	Choreology Ltd
Mark Hale	Interwoven Inc.
Gordon Hamilton	AppliedTheory, individual
Roddy Herries	Choreology Ltd
Mark Little	Hewlett-Packard Co.
Anne Manes	Systinet
Savas Parastatidis	Hewlett-Packard Co.
Bill Pope	Bowstreet, individual
Mark Potts	Individual, Talking Blocks
Pal Takacsi-Nagy	BEA Systems, Inc.
James Tauber	Bowstreet, individual
Sazi Temel	BEA Systems, Inc.
Steve Viens	individual
Jim Webber	Hewlett-Packard Co.
Steve White	SeeBeyond Inc.

61
62
63
64
65
66
67

The primary authors and editors of the main body of the specification were, in alphabetical order

Alex Ceponkus (alex@ceponkus.org)
Sanjay Dalal (sanjay.dalal@bea.com)

68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91

Tony Fletcher (tony.fletcher@choreology.com)
Peter Furniss (peter.furniss@choreology.com)
Alastair Green (alastair.green@choreology.com)
Bill Pope (zpope@pobox.com)

We thank Pal Takacsi-Nagy of BEA Systems Inc and Bill Pope for their 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.

91 **Typographical and Linguistic Conventions and Style**

92
93 The initial letters of words in terms which are defined (at least in their substantive or
94 infinitive form) in the Glossary are capitalized whenever the term used with that exact
95 meaning, thus:

96
97 Cancel
98 Participant
99 Application Message

100
101 The first occurrence of a word defined in the Glossary is given in bold, thus:

102 **Coordinator**

103
104 Such words may be given in bold in other contexts (for example, in section headings or
105 captions) to emphasize their status as formally defined terms.

106
107 The names of abstract BTP protocol messages are given in upper-case throughout:

108
109 BEGIN
110 CONTEXT
111 RESIGN
112

113
114 The values of elements within a BTP protocol message are indicated thus:

115
116 BEGIN/atom

117
118 BTP protocol messages that are related semantically are joined by an ampersand:

119
120 BEGIN/atom & CONTEXT

121
122 BTP protocol messages that are transmitted together in a compound are joined by a + sign:

123
124 ENROL + VOTE

125
126 XML schemata and instances are given in Courier:

127
128 <ctp:begin> ... </ctp:begin>

129
130 Terms such as **MUST**, **MAY** and so on, which are defined in RFC [TBD number], “[TBD
131 title]” are used with the meanings given in that document but are given in lowercase bold,
132 rather than in upper-case:

133
134 An Inferior **must** send one of RESIGN, PREPARED or CANCELLED to its
135 Superior.

137	Contents	
138		
139	Copyright and related notices.....	2
140	Acknowledgements	3
141	Typographical and Linguistic Conventions and Style	5
142	Contents	6
143	Part 1. Purpose and Features of BTP	11
144	Introduction.....	11
145	Development and Maintenance of the Specification.....	13
146	Structure of this specification.....	14
147	Conceptual Model	15
148	Example Core.....	15
149	Business transactions	16
150	External Effects.....	17
151	Two-phase outcome	18
152	Actors and roles	19
153	Superior:Inferior relationship.....	19
154	Business transaction trees	20
155	Atoms and Cohesions	22
156	Participants, Sub-Coordinator and Sub-Composers	23
157	Business transaction creation	24
158	Business transaction propagation.....	25
159	Creation of Intermediates (Sub-Coordinator and Sub-Composers).....	26
160	“Checking” and context-reply.....	27
161	Message sequence.....	28
162	Control of inferiors	33
163	Evolution of confirm-set	35
164	Confirm-set of intermediates	39
165	Optimisations and variations	41
166	Spontaneous prepared	41
167	One-shot.....	42
168	Resignation	44
169	One-phase confirmation.....	45
170	Autonomous cancel, autonomous confirm and contradictions	45
171	Recovery and failure handling.....	46
172	Types of failure	46
173	Persistent information	47
174	Recovery messages	48
175	Redirection.....	49
176	Terminator:Decider failures and transaction timelimit	50
177	Contradictions and hazard.....	51
178	Relation of BTP to application and carrier protocols	52
179	Other elements	53
180	Identifiers	53
181	Addresses	54

182	Qualifiers	55
183	Part 2. Normative Specification of BTP	58
184	Actors, Roles and Relationships	58
185	Relationships.....	58
186	Roles	60
187	Roles involved in the outcome relationships	61
188	Superior.....	61
189	Inferior	62
190	Enroller	64
191	Participant	64
192	Sub-coordinator.....	65
193	Sub-composer	66
194	Roles involved in the control relationships.....	66
195	Decider.....	66
196	Coordinator	67
197	Composer	67
198	Terminator.....	68
199	Initiator.....	69
200	Factory	69
201	Other roles	70
202	Redirector.....	70
203	Status Requestor.....	70
204	Summary of relationships	71
205	Abstract Messages and Associated Contracts	73
206	Addresses	73
207	Request/response pairs.....	75
208	Compounding messages	75
209	Extensibility	77
210	Messages.....	77
211	Qualifiers	78
212	Messages not restricted to outcome or control relationships.....	78
213	CONTEXT.....	79
214	CONTEXT_REPLY	80
215	REQUEST_STATUS	81
216	STATUS	81
217	FAULT.....	83
218	REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES	86
219	Messages used in the outcome relationships	86
220	ENROL	86
221	ENROLLED	88
222	RESIGN	88
223	RESIGNED.....	89
224	PREPARE	90
225	PREPARED	91
226	CONFIRM	92
227	CONFIRMED.....	93

228	CANCEL	94
229	CANCELLED.....	95
230	CONFIRM_ONE_PHASE	96
231	HAZARD.....	97
232	CONTRADICTION.....	98
233	SUPERIOR_STATE.....	99
234	INFERIOR_STATE.....	101
235	REDIRECT	102
236	Messages used in control relationships.....	103
237	BEGIN	103
238	BEGUN.....	105
239	PREPARE_INFERIORS	106
240	CONFIRM_TRANSACTION	107
241	TRANSACTION_CONFIRMED.....	109
242	CANCEL_TRANSACTION	110
243	CANCEL_INFERIORS	111
244	TRANSACTION_CANCELLED.....	112
245	REQUEST_INFERIOR_STATUSES	113
246	INFERIOR_STATUSES	114
247	Groups – combinations of related messages.....	116
248	CONTEXT & application message.....	116
249	CONTEXT_REPLY & ENROL.....	117
250	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED.....	118
251	CONTEXT_REPLY & ENROL & application message (& PREPARED)	119
252	BEGUN & CONTEXT	120
253	BEGIN & CONTEXT.....	120
254	Standard qualifiers	120
255	Transaction timelimit.....	120
256	Inferior timeout	121
257	Minimum inferior timeout	122
258	Inferior name.....	122
259	State Tables	124
260	Status queries	124
261	Decision events	125
262	Disruptions – failure events	125
263	Invalid cells and assumptions of the communication mechanism	126
264	Meaning of state table events.....	126
265	Persistent information	130
266	Superior state table.....	135
267	Inferior state table	139
268	Persistent information	145
269	XML representation of Message Set.....	148
270	Addresses	148
271	Qualifiers	149
272	Identifiers	149
273	Message References.....	149
274	Messages.....	149

275	CONTEXT.....	149
276	CONTEXT_REPLY	150
277	REQUEST_STATUS	150
278	STATUS	150
279	FAULT.....	151
280	ENROL	152
281	ENROLLED	152
282	RESIGN	153
283	RESIGNED.....	153
284	PREPARE.....	153
285	PREPARED.....	154
286	CONFIRM	154
287	CONFIRMED.....	154
288	CANCEL	155
289	CANCELLED.....	155
290	CONFIRM_ONE_PHASE	155
291	HAZARD.....	156
292	CONTRADICTION.....	156
293	SUPERIOR_STATE.....	156
294	INFERIOR_STATE.....	157
295	REDIRECT	157
296	BEGIN	157
297	BEGUN.....	158
298	PREPARE_INFERIORS	158
299	CONFIRM_TRANSACTION	159
300	TRANSACTION_CONFIRMED.....	159
301	CANCEL_TRANSACTION	159
302	CANCEL_INFERIORS	160
303	TRANSACTION_CANCELLED.....	160
304	REQUEST_INFERIOR_STATUSES	160
305	INFERIOR_STATUSES	161
306	Standard qualifiers.....	161
307	Transaction timelimit.....	161
308	Inferior timeout	161
309	Minimum inferior timeout	161
310	Inferior name.....	162
311	Compounding of Messages.....	162
312	XML Schemas	163
313	XML schema for BTP messages.....	163
314	XML schema for standard qualifiers	176
315	Carrier Protocol Bindings	178
316	Carrier Protocol Binding Proforma.....	178
317	Bindings for request/response carrier protocols	179
318	Request/response exploitation rules.....	180
319	SOAP Binding	181
320	Example scenario using SOAP binding.....	183
321	SOAP + Attachments Binding.....	185

322	Conformance.....	187
323	Part 3. Glossary	191
324		
325		

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~~ numerous 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 is designed to allow coordination of application work between multiple participants owned or controlled by autonomous organizations. BTP uses a two-phase outcome coordination protocol to ensure the overall application achieves a consistent result. ~~create atomic effects (results of computations).~~ BTP also permits the consistent outcome to be defined a priori -- all the work is confirmed or none is -- (an atomic business transaction or atom) or for the composition of such atomic units of work (atoms) into cohesive business transactions (cohesions), which allow application intervention into the selection of the ~~atoms work which will to~~ be confirmed, ~~and of those which will be cancelled~~ (a cohesive business transaction or cohesion).

~~BTP's ability to coordinate between~~ 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 makes it 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.

369 The BTP is based on a permissive and minimal approach, where constraints on
370 implementation choices are avoided. The protocol also tries to avoid unnecessary
371 dependencies on other standards, with the aim of lowering the hurdle to implementation.
372
373
374

374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412

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

Structure of this specification

This specification document includes, in Part 1, an explanation and description of the conceptual model of BTP, and, in Part 2, a fully normative specification of the protocol.

The use and definition of terms in the model can be regarded as authoritative but should not be taken to restrict implementations or uses of BTP. In case of (unintended) disagreement between the parts, Part 2 takes precedence over Part 1.

Part 1 contains

- Executive Summary
- This document structure description
- Conceptual Model

Part 2 contains the following sections:

- Actors, roles and relationships: defines the model entities used in the specification, their relationships to each other and indicates the correspondence of these to real implementation constructs; this section also lists which messages are sent and received for each role.
- Abstract message set: defines a set of abstract messages that are exchanged between software agents performing the various roles to create, progress and complete the relationships between those roles. For each abstract message the parameters are defined and the associated “contract” is stated – the contract defines the meaning of the message in terms of what the receiver can infer of the sender’s state and the intended effect on the receiver. This section does not itself specify a particular encoding or representation of the messages nor a single mechanism for communicating the messages
- State tables: specifies the state transitions for the Superior and Inferior roles, detailing when particular messages may be sent and when internal decisions may be made that affect the state
- XML representation: defines an XML representation of the message set. Other representations of the message set, or parts of it are possible – these may or may not be suitable for interoperation between heterogeneous implementations.
- Carrier protocol bindings: defines a “carrier binding proforma” that details the information required to specify the mapping to a particular carrier protocol such that independent implementations can interoperate. The proforma requires an identification for the binding, the nature of the addressing information used with the binding, how the messages are represented and encoded and how they are carried (e.g. which carrier protocol messages or fields they are in) and may include other requirements.
- Using the carrier protocol proforma, this section fully specifies bindings to SOAP 1.1, using the XML representation of the abstract message set.
- Conformance definitions: defines combinations of facilities (expressed as roles) that an implementation can declare it supports

456 Part 3 contains a glossary that provides succinct definitions of terms used in the rest of the
457 document.
458

459 Conceptual Model

460 This section introduces the concepts of BTP. Its use and definition of terms can be regarded
461 as authoritative but should not be taken to restrict implementations or uses of BTP. Part 2 of
462 the specification is fully normative and in case of disagreement takes precedence over
463 statements or examples in this section.
464

465 BTP is designed to make minimal assumptions about the implementation structure and the
466 properties of the carrier protocols. This allows BTP to be bound to more than one carrier
467 protocol. BTP implementations built in quite different ways should be able to interoperate if
468 they are bound to the same carrier protocol. This flexibility requires that much of the text is
469 abstract and may be difficult to visualise in the absence of a particular implementation pattern
470 or carrier protocol. To aid understanding some possible implementation examples are
471 presented in the following text.
472

473 Example Core

474 An advanced manufacturing company (*Manufacturer A*) orders the parts and services it
475 needs on-line. It has existing relationships with parts suppliers and providers of services
476 such as shipping and insurance. All of the communications between these organizations
477 is via XML messages. The interactions of these business transactions include:

- 478 1. *Manufacturer A*'s production scheduling system sends an Order message to a
479 *Supplier*.
- 480 2. The *Supplier*'s order processing system sends back an order confirmation with the
481 details of the order.
- 482 3. *Manufacturer A* orders delivery from a *Shipper* for the ordered parts.
- 483 4. The *Shipper* evaluates the request and based on its truck schedule it sends back a
484 positive or negative reply.
- 485 5. Some shipments need to be insured based on their value, where they are shipped
486 from, and method of transportation. *Manufacturer A* sends an Order message to an
487 *Insurer* when this is necessary.
- 488 6. The *Insurer* responds with a bid or a no-bid response.
489

490 Problems have arisen with some of these interactions.

- 491 • *Manufacturer A* had ordered parts from a supplier and contacted shipper M about
492 delivering the goods. Shipper M was busy and agreed to the contract but only for a
493 scheduled delivery the day after the parts were needed. By the time this was
494 addressed it was too late to schedule alternate shipping.
- 495 • There were communications problems with supplier Z that resulted in an order not
496 being confirmed. The shipper arrived to pick up the order and supplier Z knew
497 nothing about it.

498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516

- Goods have been shipped without insurance when company policy dictated that insurance was required.

These problems occur because of the unreliable nature of the Internet and the lack of visibility a company has into the workings and state of an outside organization. By using BTP in support of this supply application, these problems can be ameliorated.

BTP is a protocol, that is, a set of specific messages that get exchanged between computer systems supporting an application, with rules about the meaning and use of the messages. The computer systems will also exchange application-specific messages. Thus, within the example, the Manufacturer's system and the Supplier's system (say), will exchange messages detailing what the goods are, how many, what price and will also exchange BTP messages. The parts of the application in both systems that handle these different sets of messages can be distinguished, as in Figure 1 ~~Figure 1~~. In each BTP-using party there is an **application element** and a **BTP element**. The application elements exchange the order information and cause the associated business functions to be performed. The BTP elements, which send and receive the BTP messages, perform specific roles in the protocol. These BTP elements assist the application in getting the work of the application done. The application element, as understood by this model, may include supporting infrastructure elements, such as containers or interceptors, as well as application-specific code.

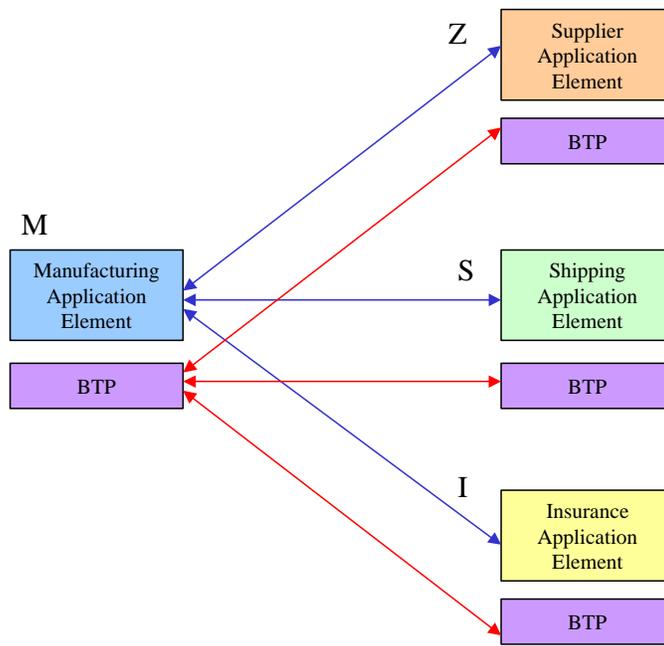


Figure 1 – Manufacturer Example

517
518
519
520
521

Business transactions

522 A **Business Transaction** can be defined as a consistent change in the state of a business
523 relationship between two or more **parties**. A business relationship is any distributed state
524 held by the parties which is subject to contractual constraints agreed by those parties. For
525 example, an master purchasing agreement, which permits the placing of orders for
526 components by known buying organizations allows a buyer and a seller to create and
527 subsequently exchange meaningful information about the creation and processing of an order.
528 Such agreements (and the consequent specification of shared or canonical data formats and of
529 the messages that carry those formats, and their permitted sequences, all of which are needed
530 for an automated implementation of an agreement) stem from business negotiations and are
531 specific to a particular trading or information exchange community (group of potential
532 parties). This definition of a business relationship is deliberately silent on the nature of the
533 “business” transacted between the parties: it might be trading for profit, verification of
534 authorizations for expenditure or loans, consistent publication (replication) of government
535 ordinances to multiple sites, or any other computerized interaction where the parties require
536 high confidence of consistent delivery or processing of data. In each party or site where
537 business relationship state resides an application system must exist which can maintain that
538 state and communicate it as needed to other parties. The Business Transaction Protocol (BTP)
539 assists the application systems of the various parties to bring about consistent and coordinated
540 changes in the relationship as viewed from each party. BTP assumes that for a given business
541 transaction, state changes occur, or are desired, in computer systems controlled by some set
542 of parties, and that these changes are related in some application-defined manner. BTP
543 assumes that the parties involved in a business transaction have distinct and autonomous
544 application systems, which do not require knowledge of each others’ implementation or
545 internal state representations in volatile or persistent storage. Access to such loosely coupled
546 application systems is assumed to occur only through service interfaces.

547
548 Thus the state changes that BTP is concerned with are only those affecting the immediate
549 business relationship. Although these externally visible changes will typically correspond to
550 internal state changes of the parties, use of BTP does not itself imply any constraints or
551 requirements on the internal state.¹

552 External Effects

553
554
555 BTP coordinates the state changes caused by the exchange of application messages. These
556 state changes are part of the contract between BTP-using parties. In the manufacturing
557 example, an interaction between the manufacturer and the supplier might involve the supplier
558 receiving the order (an application message), checking to ensure that it had enough product
559 on hand, reserving the product in the manufacturer’s name and replying. When the
560 manufacturer agrees to the purchase (assuming the shipping and insurance are also reserved),
561 BTP messages are sent to confirm the purchase. In this case, the supplier is offering a **BTP-**
562 **enabled service** – the application element and its supporting BTP elements together offer this
563 service.

¹ Although a Business Transaction is defined as concerning a business relationship, the facilities of BTP make it suitable for other environments where loosely coupled systems require coordination and consistency.

565 In general, to be able to satisfy such contracts a BTP-enabled service must support in some
 566 manner provisional or tentative state changes (the transaction's **provisional effect**) and
 567 completion either through confirmation (**final effect**) or cancellation (**counter-effect**). The
 568 meaning of provisional, final, and counter-effect are specific to the application and to the
 569 implementation of the application. In the example, the reservation of the order is the
 570 provisional effect, the completion of the purchase is the final effect.
 571 Some of the implementation approaches are shown in Table 1 ~~Table 1~~. From the perspective
 572 of BTP and the initiator application, all these are considered equivalent. Outside of BTP the
 573 underlying business relationship (or contract) between the parties can constrain the degree to
 574 which the effects are visible.
 575

576 **Table 1 Some alternatives for provisional, final and counter effects**

<u>provisional effect</u>	<u>final effect</u>	<u>counter effect</u>	<u>Comment</u>
<u>Store intended changes without performing them</u>	<u>Perform the changes</u>	<u>Delete the stored changes, unperformed</u>	<u>Provisional effect may include checking for validity</u>
<u>Perform the changes, making them visible; store information to undo the changes</u>	<u>Delete undo information</u>	<u>Perform undo action</u>	<u>One form of compensation approach</u>
<u>Store original state, prevent outside access, perform changes</u>	<u>Allow access</u>	<u>Restore original state; allow access</u>	<u>a typical database approach</u>

577 These alternatives are not the only ones – they can be combined or varied. The visible state
 578 of the application information prior to confirmation or cancellation may be different from
 579 both the original state and the final state.
 580

581 Especially in the compensation approach, if the changes are cancelled, the counter-effect may
 582 be a precise inversion or removal of provisional changes, or it may be the processing of
 583 operations that in some way compensate for, make good, alleviate or supplement their effect.
 584 There may be side-effects of various kinds from a counter-effected operation – such as
 585 levying of cancellation charges or the record of the operation may be visible, but marked as
 586 cancelled. The possibility of these side-effects is considered to be part of the overarching
 587 contract.
 588

589 **Two-phase outcome**

590 The BTP protocol coordinates the transitions into and out of the event states described above
 591 by sending messages between the transaction parties. This involves a two-phase exchange.
 592 First the application elements exchange messages that determine the characteristics and cause
 593 the transition.
 594

595 the performance of the provisional effect; then a separate message, to the BTP element,
596 asking for the performance of the final or the counter effect.

597
598 In general, the application elements in the systems involved having first communicated the
599 application messages, each system that has to make changes in its own state:

- 600 • determines whether it is able achieve its provisional effect and then ensure it
601 will be able either to cancel (counter-effect) its operation or to confirm (give
602 final effect to) its operation, whichever is subsequently instructed, and
- 603 • reports its ability to confirm-or-cancel (its preparedness) to a central
604 coordinating entity.

605
606 And, after receiving these reports, the coordinating entity:

- 607 • determines which of the systems should be instructed to confirm and which
608 should be instructed to cancel
- 609 • informs each system whether it should confirm or cancel (the “outcome”).by
610 sending a message to its BTP element

611
612 When there is more than one system that has to make changes such a two-phase exchange
613 mediated by a coordinator is required to achieve a consistent outcome for a set of operations.
614 The two-phases of the BTP protocol ensure that either the entire attempted transaction is
615 abandoned or a consistent set of participants is confirmed.

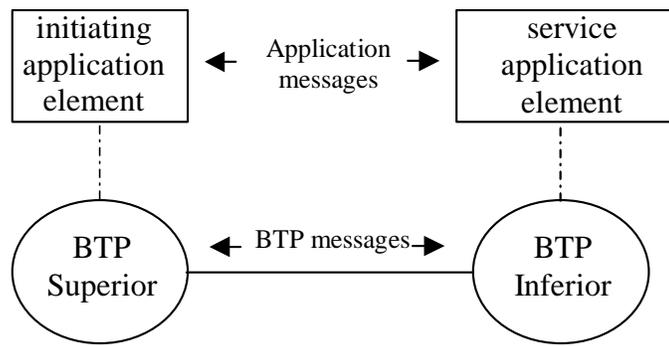
616 Actors and roles

617
618
619 BTP centres on the bilateral relationship between the computer systems of the coordinating
620 entity and those of one of the parties in the overall business transaction. For each bilateral
621 relationship in a business transaction, a software agent within the coordinating entity’s
622 systems plays the BTP role of Superior and a software agent within the systems of the party
623 play the BTP role of Inferior. The concept “role” refers strictly to the participation in a
624 particular relationship in a particular business transaction. The software agent performing a
625 role is termed an Actor. An Actor is distinguished from other Actors by being distinguishably
626 addressable. The same Actor may perform multiple roles in the same business transaction
627 (including the case where a Superior is also an Inferior), and may also perform the same or
628 different roles in multiple business transactions, either concurrently or consecutively.

629 Superior:Inferior relationship

630
631
632 A basic case of a single Superior:Inferior relationship, including the association with
633 application elements, is illustrated in Figure 2~~Figure 2~~. In many cases, including the
634 manufacturer supply example, the application element associated with the superior will
635 directly initiate the application exchanges –as does the manufacturer’s application client to
636 the supplier’s server, for example – but this is not invariably the case. It is possible that the
637 first direct communication between the application elements is from one associated with an

638 [inferior to the one associated with the superior – for example, with an application that](#)
 639 [requested quotes by advertising the identity and location of the Superior along with invitation](#)
 640 [to quote; incoming quotes would be the first direct application message exchanged. In all](#)
 641 [cases the topmost application element in a tree or subtree will be aware of the business](#)
 642 [transaction first. How the identity of the transaction and the address of the BTP Superior are](#)
 643 [communicated to the secondary application element is a matter for the application protocol](#)
 644 [and not strictly part of BTP, although it will commonly be done by associating a BTP](#)
 645 [CONTEXT message with application messages..](#)
 646



647
 648

Figure 2 Basic Superior:Inferior relationship for BTP

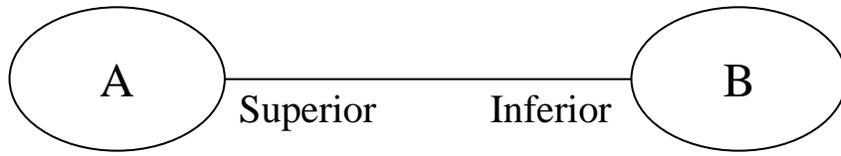
649
 650 [An Inferior is associated with some set of application activities that create effects within the](#)
 651 [party, for a given business transaction. As stated above, commonly, though not invariably,](#)
 652 [this application activity within the party will be a result of some operation invocations from](#)
 653 [elsewhere \(shown as the “initiating application element” in Figure 2Figure 2\), associated with](#)
 654 [the Superior to an application element associated with the Inferior \(shown as “Service](#)
 655 [application element”\). This second application element determines what activities the Inferior](#)
 656 [is responsible for, and then the Inferior is responsible for reporting to the Superior whether](#)
 657 [the associated operations’ provisional effect can be confirmed/cancelled – this is called](#)
 658 [“becoming prepared”, because the Inferior has to remain prepared to receive whichever order](#)
 659 [eventually arrives \(subject to various exceptions and exclusions, detailed below\).](#)

660
 661

Business transaction trees

662
 663
 664
 665
 666
 667
 668
 669
 670

[There are many patterns in which the service provider participants involved in a business](#)
[transaction may be arranged in respect of the two-phase exchange and the determination of](#)
[which are eventually confirmed. The simplest is shown in Figure 3Figure 3involving only](#)
[two parties – one \(B\) making itself subject to the decision of confirm-or-cancel made by the](#)
[other \(A\). This basic bilateral relationship, in which one side makes itself inferior to the other,](#)
[is the building block used in all business transaction patterns. In this simplest case, the](#)
[“coordination” by the superior, A, is just that A can be sure whether the operations at the](#)
[inferior, B were eventually cancelled or confirmed.](#)



671

672

Figure 3 Simple two-party business transaction

673

674

675

676

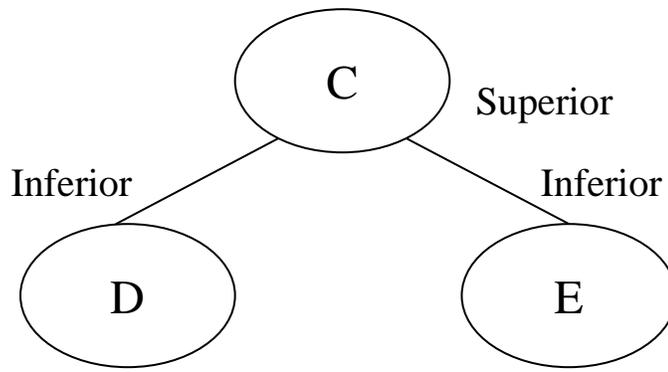
677

678

679

680

In the next simplest case, as in figure Figure 4Figure 4, a bilateral, Superior:Inferior relationship appears twice, with two Inferiors, D and E, both making themselves inferior to a single Superior, C. From the perspective of either D or E, they are in the same position as B in the previous case –they are unaware of and unaffected (directly) by each other. It is only within C that there is any linkage between the confirm-or-cancel outcomes that apply to D and E.



681

682

Figure 4 Business transaction with two inferiors

683

684

685

686

687

688

689

690

The same Superior:Inferior relationship is used in business transaction trees that are both “wider” – with more Inferiors reporting their preparedness to be confirm-or-canceled to a single Superior – and “deeper”. In a “deeper” tree, as in figure Figure 5Figure 5, an entity (G) that is Superior to one or more Inferiors (H, J), is itself Inferior to another entity (F) – it is said to be **interposed** or is an **Intermediate** (either term can be used). In this case, G will collect the information on preparedness of its Inferiors before passing on its own report to its Superior, F, and awaiting the outcome as advised by F.

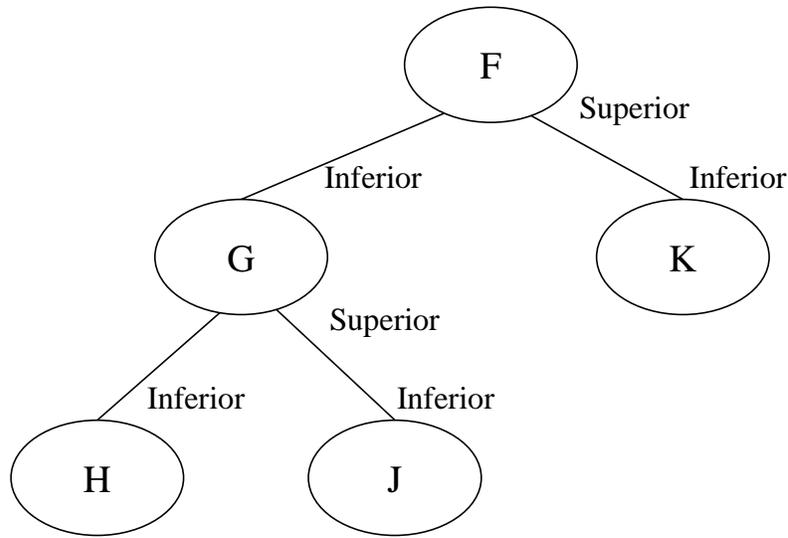


Figure 5 Business transaction with an Intermediate (interposition)

A business transaction tree, made up of these bilateral Superior:Inferior relationships can, in theory, be arbitrarily “wide” or “deep” – there are no fixed limits to how many Inferiors a single Superior can have, or how many levels of intermediates there are between the top-most Superior (that is Inferior to none) and the bottom-most leaf Inferior. The actual creation of the tree depends on the behaviour and requirements of the application. Given the (potentially) inter-organisational nature of business transactions, there may be no overall design or control of the structure of the tree.

Each Inferior has only one Superior. However, a single Superior may (and commonly does) have multiple relationships with Inferiors, and may have such relationships with multiple Inferiors within each party to the transaction, and with Inferiors within multiple parties.

Atoms and Cohesions

As described in the previous section, the Superior receives reports from its Inferiors as to whether they are prepared. It gathers these reports in order to ascertain which Inferiors should be cancelled and which confirmed - those that cannot prepare will have already cancelled themselves. This determined, directly or indirectly, by the application element responsible of the creation and control of the Superior, which determines the nature of the Superior. There are two dimensions of variation in the Superior: is it an Inferior to another Superior; does it treat its own Inferiors atomically or cohesively. The distinction between atomic and cohesive behaviour is whether the Superior will choose or allow some Inferiors to cancel while others confirm – this is not allowed for atomic behaviour, in which all must confirm or all must cancel, but is for cohesive.

The possible cases for a Superior, given these two dimensions of variation, are:

- 720 a) the application element initiated the business transaction (causing the creation of
721 the Superior), and instructed that all Inferiors of the Superior should confirm or
722 all should cancel; the Superior is an **Atom Coordinator**;
723 b) the application element initiated the business transaction, but deferred the choice
724 of which Inferiors should confirm until later, allowing it (the application element)
725 to choose some subset to be confirmed, others to cancel; the Superior is a
726 **Cohesion Composer**;
727 c) the application element was itself involved in an existing business transaction,
728 and the Superior in this relationship is the Inferior in another one; this application
729 element instructed that all Inferiors of this Superior should confirm, but only if
730 confirmation is instructed from above or all should cancel; the Superior is an
731 (atomic) **Sub-coordinator**;
732 d) the application element was itself involved in an existing business transaction,
733 and the Superior in this relationship is the Inferior in another one; this application
734 element deferred the choice of which Inferiors should be candidates to confirm
735 until later, allowing it (the application element) to choose some subset to be
736 confirmed, given that confirmation is instructed from above, others to cancel; the
737 Superior is a (cohesive) **Sub-composer**.
738

739 In the atomic case, the two-phase outcome exchange means a Superior acting as an atomic
740 Coordinator or sub-coordinator will treat any Inferior which cannot prepare to cancel/confirm
741 as having veto power, causing the Superior to instruct all its Inferiors to cancel. A business
742 transaction whose topmost Superior is atomic is an Atomic Business Transaction, or Atom –
743 the superior is the Atom Coordinator.
744

745 In the cohesion case, with the Superior acting as a cohesive Composer or Sub-Composer, the
746 controlling application element will determine the implications of an Inferior’s failure to be
747 prepared to confirm-or-cancel; the application element may cancel some or all other Inferiors,
748 do other application work, which may involve new Inferiors or may just accept the
749 cancellation of that one Inferior and carry on. A business transaction whose topmost Superior
750 is cohesive is a Cohesive Business Transaction, or Cohesion – the Superior is the Cohesion
751 Composer.
752

753 For a cohesion, the set of Inferiors that eventually confirm is called the **confirm-set**. The term
754 is also used to mean the set of Inferiors that have been chosen to (potentially) confirm before
755 the final outcome is decided – if the cohesion is eventually cancelled, then confirm-set
756 Cancels. (See section “Evolution of confirm-set”). The confirm-set of an Atom is all of the
757 Inferiors.
758

759 If the Superior is itself an Inferior, its own action of becoming prepared, and reporting this to
760 its own Superior will depend on the receipt of prepared reports from its Inferiors. If it is
761 atomic (i.e. is a sub-coordinator), it will only become prepared if all Inferiors reported
762 preparedness to it; if it is cohesive (i.e. is a sub-composer), the controlling application
763 element will determine whether the set of Inferiors that have reported as prepared is
764 sufficient.
765

766 If the Superior is not an Inferior, the determination of when, if and, for a Cohesion, what it
767 should confirm depends on the controlling application. This “top-most” Superior has a
768 different relationship to the controlling application to that of an Inferior to its Superior: an
769 Inferior reports that it is prepared to the Superior, which instructs it whether to cancel or to
770 confirm; the top-most Superior is asked by the application element to attempt to confirm, but,
771 dependent on the preparedness of its Inferiors, the top-most Superior makes the final
772 decision. Consequently the top-most Superior is termed the **Decider**; the application element
773 that asks it to confirm is the **Terminator**.

774

775 Participants, Sub-Coordinators and Sub-Composers

776

777 An Inferior may directly be responsible for applying the confirm-or-cancel decision to some
778 application effects, or may in turn be a BTP Superior to which others will enrol. If it only
779 handles application effects it is called a **Participant**, in the latter case it is called a **Sub-**
780 **coordinator** or a **Sub-composer**, depending on whether it is atomic or cohesive with respect
781 to its own future Inferiors. (If an Inferior is both responsible for application effects, and is a
782 BTP Superior, it is not considered a Participant, according to the strict definitions, though
783 informally it may be referred to as such.) The Superior is unaware, via the BTP exchanges,
784 whether the Inferior is a Participant, Sub-coordinator or Sub-composer. This specification
785 does not define messages or interfaces for the creation of Participants or for the application
786 element to tell the Participant what the application effects are or how they are to be confirmed
787 or cancelled as necessary. (Although out-of-scope for this specification, one or more APIs
788 could be standardised.)

789

790

791 Business transaction creation

792

793 This section describes in some detail how a BTP business transaction is created. The
794 interaction diagram in Figure 6 ~~Figure 6~~ also shows this sequence. The messages shown in
795 lower-case italics (between Factory and Coordinator) represent interactions that are not
796 specified in BTP.

797

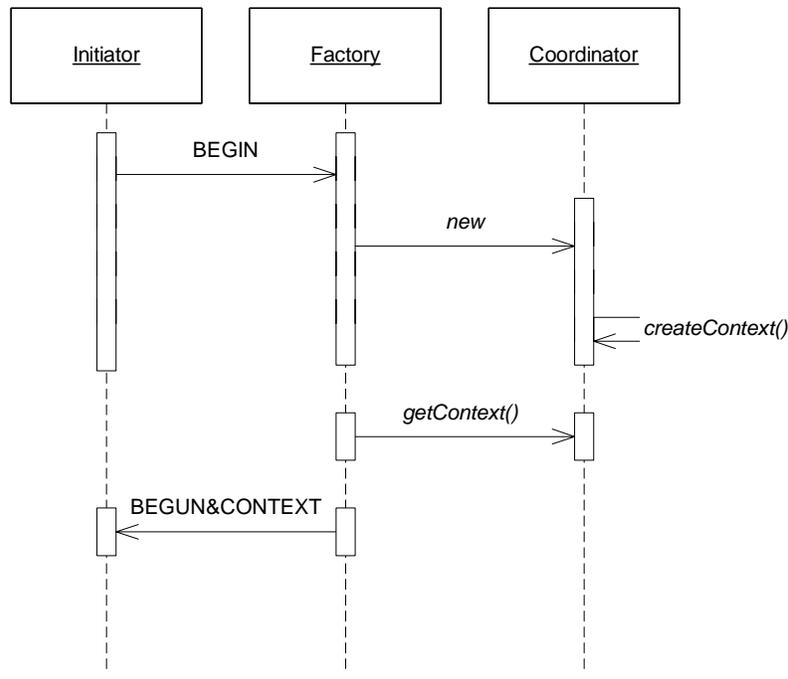


Figure 6 – Creation of a business transaction

798

799

800 A business transaction is started at the initiative of an application element, which causes the
 801 creation of a Coordinator or Composer. Any Inferiors participating in this transaction will
 802 enrol with this Superior. BTP defines abstract messages (BEGIN, BEGUN) to request this
 803 but the equivalent function can also be achieved using proprietary means, especially if the
 804 Factory or Coordinator is an internal component of the initiating application. If the BTP
 805 messages are used, the application element performs the role of Initiator and sends BEGIN to
 806 a Factory. The BEGIN message identifies whether a Coordinator (for an atom) or a Composer
 807 (for a cohesion) is desired. The Factory, after the creation of the new Coordinator or
 808 Composer, replies with related BEGUN and CONTEXT messages. "Related" means they are
 809 sent together in a manner that has semantic significance; how this is represented is
 810 determined by the binding in use. The Coordinator's or Composer's creation is the
 811 establishment of a new instance of a BTP role. It may involve only the assignment of a new
 812 identifier within an existing Actor (which may also be performing the Factory role, for
 813 example). Alternatively a new Actor with a distinct address may be instantiated. These and
 814 other alternatives are implementation choices, and BTP ensures other Actors are unaffected
 815 by the choice made.

816

817 The BEGUN message provides the addressing and identification information needed for a
 818 Terminator to access the new Coordinator or Composer as Decider; the application element
 819 performing the Initiator role may itself act as Terminator, or may pass this information to
 820 some other application element.

821

822 Whether this interoperable BTP Initiator:Factory relationship or some other mechanism is
 823 used to initiate the business transaction, a CONTEXT is made available. This identifies the

824 Coordinator or Composer as a Superior – containing both addressing information and the
825 identification of the relevant state information. The CONTEXT is also marked as to whether
826 or not this Superior will behave atomically with respect to its Inferiors (i.e. is it a Coordinator
827 or Composer).

828

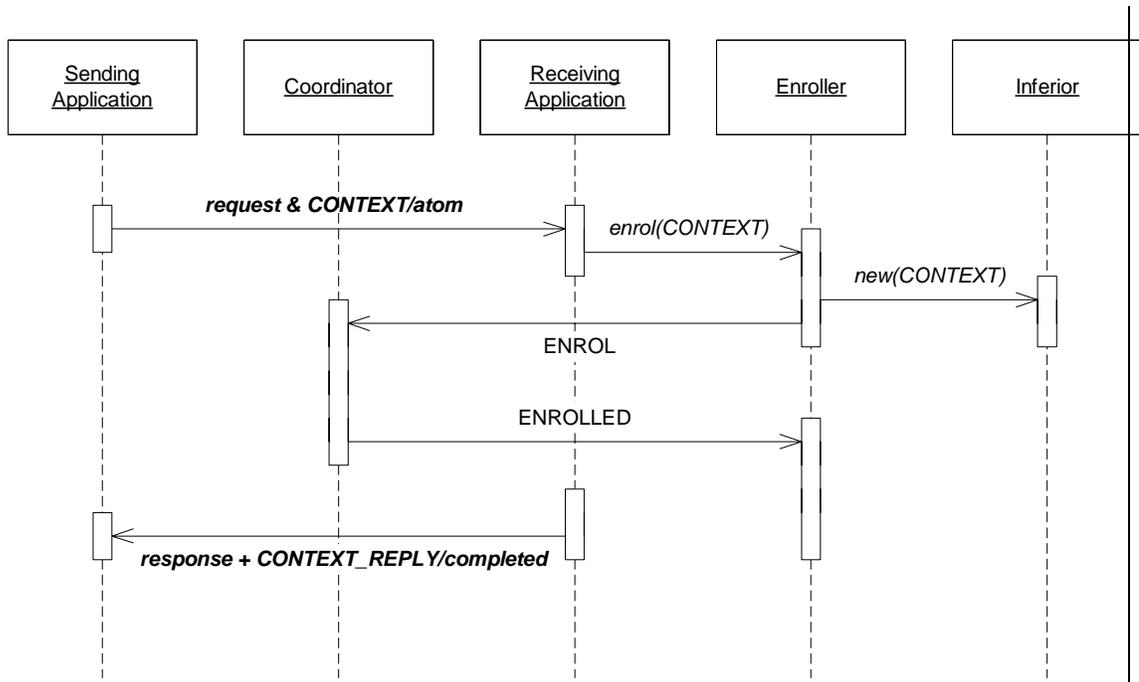
829 Business transaction propagation

830 The propagation of the business transaction from one party to another, to establish the
831 Superior:Inferior relationships involves the transmission of the CONTEXT. This is
832 commonly in association with, or related to, one or more application messages between the
833 parties. In a typical case, an application message is sent from the application element that
834 performed the Initiator role (the “sending application” in Figure 2~~Figure-2~~) to some other
835 element (the receiving application). The CONTEXT is sent with the application message in
836 such a way that the application elements understand that work performed as a result of the
837 application message is to be the subject of a confirm-or-cancel decision of the Superior.² The
838 receiving application element causes the creation of an Inferior (which, as for the Superior
839 may involve just assignment on a new identifier, or instantiation of an new Actor) and
840 ensures the new Inferior is enrolled with the Superior identified in the received CONTEXT,
841 using an ENROL message sent to the Superior using the address in that CONTEXT.

842

843 Figure 7~~Figure-7~~ shows a sequence diagram of the propagation of a business transaction. It is
844 assumed the transaction has already been created, and thus the application element and
845 Coordinator exist. The diagram shows the Enroller as a distinct role, with non-standardised
846 interactions between the application element, the Enroller and the new Inferior The Enroller
847 role may in fact be performed by the application element, by the Inferior or by a distinct
848 entity. At least the Superior-identifier and Superior-address from the CONTEXT has to be
849 passed the Enroller and to the Inferior so they can communicate with the Coordinator (whose
850 identifier and address these are).

² The relationship between the application activity and BTP is subtle, and summarised in this sentence.



851

852

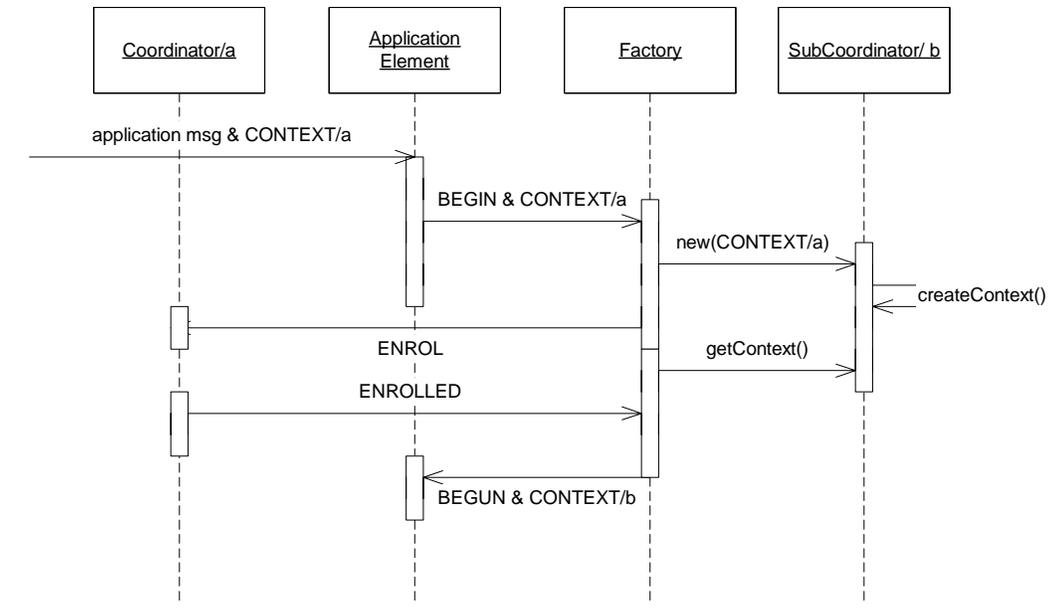
Figure 7 Sequence diagram of propagation

853

Creation of Intermediates (Sub-Coordinators and Sub-Composers)

854 If the new Inferior is to be a Sub-coordinator or Sub-composer, this can be created using a
 855 non-standard mechanism or the Initiator:Factory relationship can be used again. Figure
 856 8Figure 8 shows a sequence diagram, using the latter mechanism. The application element,
 857 having received an application message and a CONTEXT from some Superior – shown as a
 858 Coordinator/a in the diagram - wants to create the new Inferior and acting in the Initiator
 859 role, issues BEGIN to the Factory, but the CONTEXT for the original Superior
 860 (Coordinator/a) is “related” to the BEGIN. The Factory is responsible for enrolling the new
 861 Sub-coordinator or Sub-composer as an Inferior of the Superior identified by the received
 862 CONTEXT. The reply from the Factory is a related BEGUN and CONTEXT – this being the
 863 CONTEXT for the new Sub-coordinator (‘b’) or Sub-composer as a Superior. The Sub-
 864 coordinator/Sub-composer is not a Decider, as its decision is subordinated to the outcome
 865 received from the Superior. For a Sub-coordinator, further control by the application is
 866 primarily a matter of relating the new CONTEXT to appropriate application activity. For a
 867 Sub-composer there is in addition a requirement for the application to determine which of the
 868 Inferiors of the Sub-composer must have reported they are prepared before the Sub-composer
 869 can report that it is itself prepared to its own Superior, and then which of these Inferiors are to
 870 be ordered to confirm if the Sub-composer is ordered to confirm. This specification does not
 871 provide an interface or interoperable message to control this; like the relationship between
 872 application element and Participant, it is left to the implementation or independent
 873 standardisation.

874



876

877

Figure 8 – Creation of a Sub-coordinator

878 The creation of a new Inferior and establishment of a Superior:Inferior relationship does not
 879 always imply that the BTP Actors are under the control of different business parties or
 880 application elements. In particular, an application element may begin a Cohesion, then create
 881 and enrol (atomic) Sub-coordinators as Inferiors of the Composer, then associate a different
 882 Sub-coordinator’s CONTEXT with each of several aspects of the application work,
 883 transmitting that CONTEXT with the application messages for that aspect to the other parties
 884 in the business transaction. Those parties can then create Participants (or other Inferiors) that
 885 are enrolled with the appropriate Sub-coordinator. Later, the application element (as
 886 Terminator, or its equivalent) can choose which of the Cohesion Composers’ Inferiors to
 887 cancel and which to confirm. By interposing its own atomic Sub-coordinator the initiating
 888 application element can indicate to the other parties that some associated set of application
 889 work will be confirmed or cancelled as a unit. This may allow the receiving parties to share
 890 information between application operations and to make one Participant responsible for
 891 applying the outcome to several operations.

892

“Checking” and context-reply

893

894
 895 In BTP, enrolment is at the initiative of an application element that has received or has access
 896 to the CONTEXT which creates an Inferior (BTP uses a “pull” paradigm for enrolment). An
 897 application element in possession of a CONTEXT can choose, perhaps constrained by an
 898 overarching business and application understanding, whether and how many Inferiors to
 899 create and enrol. Consequently, in general, an application element which propagates a
 900 CONTEXT to another (via whatever mechanisms it choose), cannot be sure how many
 901 Inferiors will be enrolled as a result. Without further controls, there would be a possibility
 902 that an application element receiving a CONTEXT might attempt to enrol an Inferior with a
 903 Superior after the Superior had been asked to confirm, or even had completed confirmation.

904 In such a case application work that should have been part of a confirmed atomic business
905 transaction could be cancelled, violating the atomicity in a manner that will not be apparent to
906 the application.

907
908 To avoid this, whenever a CONTEXT is transmitted to another party by or on behalf of the
909 application, the transmission of the CONTEXT itself can be replied to with a
910 CONTEXT_REPLY message – this is required for an Atom, allowed for a Cohesion. An
911 application element that has received a BTP CONTEXT is able, because it knows the
912 Superior’s identification and address in the CONTEXT, to enrol Inferiors (Figure 9Figure-9).³
913 Replying with CONTEXT_REPLY means that the sender (the earlier receiver of a
914 CONTEXT) will not enrol any more Inferiors. Consequently the sender of a CONTEXT can
915 keep track of whether there are any outstanding (un-replied to) CONTEXTs that could be
916 used for an enrolment and can avoid requesting or permitting confirmation until everything is
917 safe. This check is required for an Atom, but is not always essential when the CONTEXT is
918 for a Cohesion. For a Cohesion, it is a matter for the controlling application whether all
919 would-be Inferiors must be enrolled before a confirmation decision can be made; or whether
920 it is acceptable to proceed to confirmation at some point in time with the already enrolled
921 Inferiors (or a subset thereof), accepting the automatic cancellation of any late arrivals.

922
923 CONTEXT_REPLY can also indicate that attempted enrollments failed. This can occur if the
924 Enroller is unable to contact the Superior, but it able to return a CONTEXT_REPLY to
925 where-ever the CONTEXT came from.

926
927

<u>Section explaining becoming prepared ?</u>

928 Message sequence

929 BTP messages are used in relationships between several pairs of roles. These particular pair-
930 wise relationships can be categorised into:

- 931 • Outcome relationships : the Superior:Inferior relationship (i.e. between BTP actors
932 within the transaction tree) and the Enroller:Superior relationship used in establishing it
- 933 • Control relationships : the application:BTP actor relationships that create the nodes of
934 the transaction tree (Initiator:Factory) and drive the completion (Terminator:Decider).

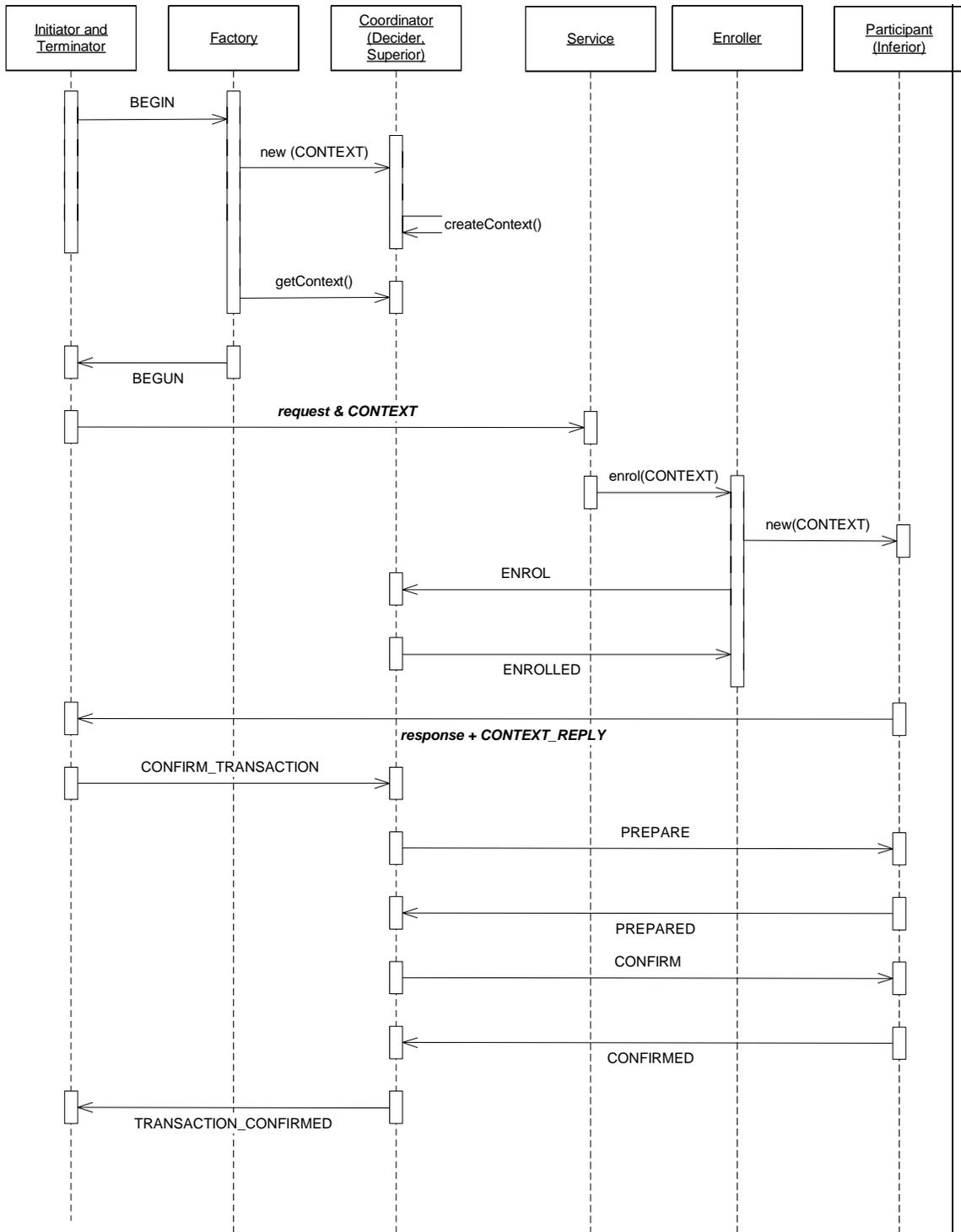
935
936 The outcome relationships and the messages used in them an essential part of BTP. For the
937 control relationships, it would be possible to achieve the same general function using non-
938 standardised messages or API mechanisms. There are other distinguishable relationships
939 between roles defined by BTP that are not standardised in this specification.

940
941 Figure 9Figure-9 shows the message exchange for the conventional progression of a simple
942 transaction to confirmation with a single Superior:Inferior relationship, assuming the standard
943 control relationship. Two application elements using a request/response application message
944 exchange are involved – the first is represented as the Initiator and Terminator, the second as
945 the Service and Enroller. The Decider/Superior is shown as a Coordinator, but with only one
946 Inferior there would be no difference with a cohesion Composer. The Factory:Coordinator

³ The “application element” from the perspective of BTP may include infrastructure software such as containers or interceptors, as well the application-specific code itself.

947 events are non-standardised, but represent interactions that must occur in some form. There
948 are other interactions between the various application groups – Initiator-Terminator and
949 Participant-Enroller-Service that are not shown – in particular the Service:Participant
950 relationship.

951
952 The message sequence is shown is the “conventional” sequence, with all messages explicitly
953 present and sent separately. There are several variations and optimisations possible – these
954 are discussed below.
955



956
957
958

Figure 9 A conventional message sequence for a simple transaction

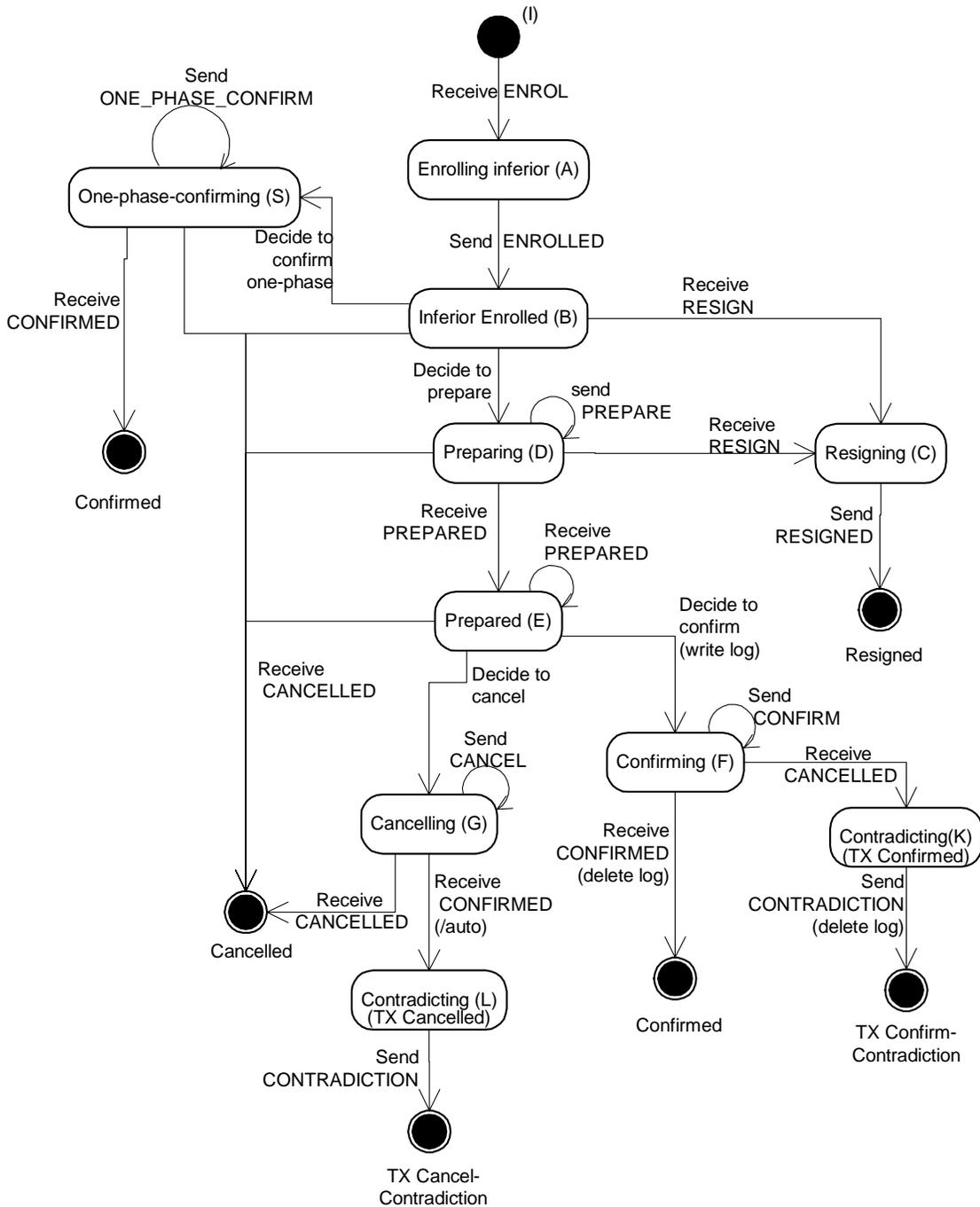
959 Note that CONTEXT has a “related” (&) relationship to BEGUN and to the application
960 request (although in the latter case the meaning of this is defined by the application, not by
961 BTP. The response + CONTEXT_REPLY has no semantic significance, and could be sent
962 separately; provided the CONTEXT_REPLY is not sent until the ENROLLED has returned.

963
964 The progression of a single instance of the central outcome (Superior:Inferior) relationship
965 can also be presented as a set of state transitions. The normative part of the specification
966 includes state tables for the Superior side of such a relationship and for the Inferior. Since a
967 single Superior (Coordinator, Composer, Sub-coordinator, Sub-composer) can have multiple
968 Inferiors, each Superior will have multiple instances of the “Superior state”. How these link
969 together is discussed below in the section “Evolution of confirm-set”, but the state transitions
970 for the individual Superior:Inferior relationships include “decision events” which constrain
971 the behaviour of the business transaction tree node as a whole, and thus define the semantics
972 of the BTP messages.

973
974 The normative state tables distinguish some states that differ only in which messages can be
975 received and thus allow for a level of error checking. The progress of the outcome
976 relationship can be followed without dropping to such a detailed level, and the state diagrams
977 shown here aggregate some of the states that are distinguished in the state tables. The single
978 letters in parentheses in the diagrams correspond to the state names used in the tables. For
979 simplicity, the state diagrams do not include the events leading to the sending of a HAZARD
980 message – the detection and recording of a “problem” – meaning that the Inferior is unable to
981 cleanly confirm or cleanly cancel the operations it is responsible for. As is specified in the
982 state tables, such a problem can be detected in most states, and reported with a HAZARD
983 message.

984
985 It should be noted that, with some exceptions, the transmission of a message **from** a Superior
986 or Inferior does not cause a state change at that side. State changes are normally caused either
987 by the receipt of a message from the peer, or by a “decision event” – which may be an
988 internal change, including a change in the persistent information for the transactions, or may
989 be the receipt of a message on another relationship (e.g. as when a Sub-coordinator receives
990 CANCEL from its Superior, which is a decision event as perceived on the relationships to its
991 Inferiors). It would be normal for an implementation on entering a new state to send the
992 message it can now send (there will be only one). It may repeat this message at any interval –
993 in practice only if there is reason to believe (due to lower-layer errors, timeout or known
994 recovery events) that messages may have got lost.

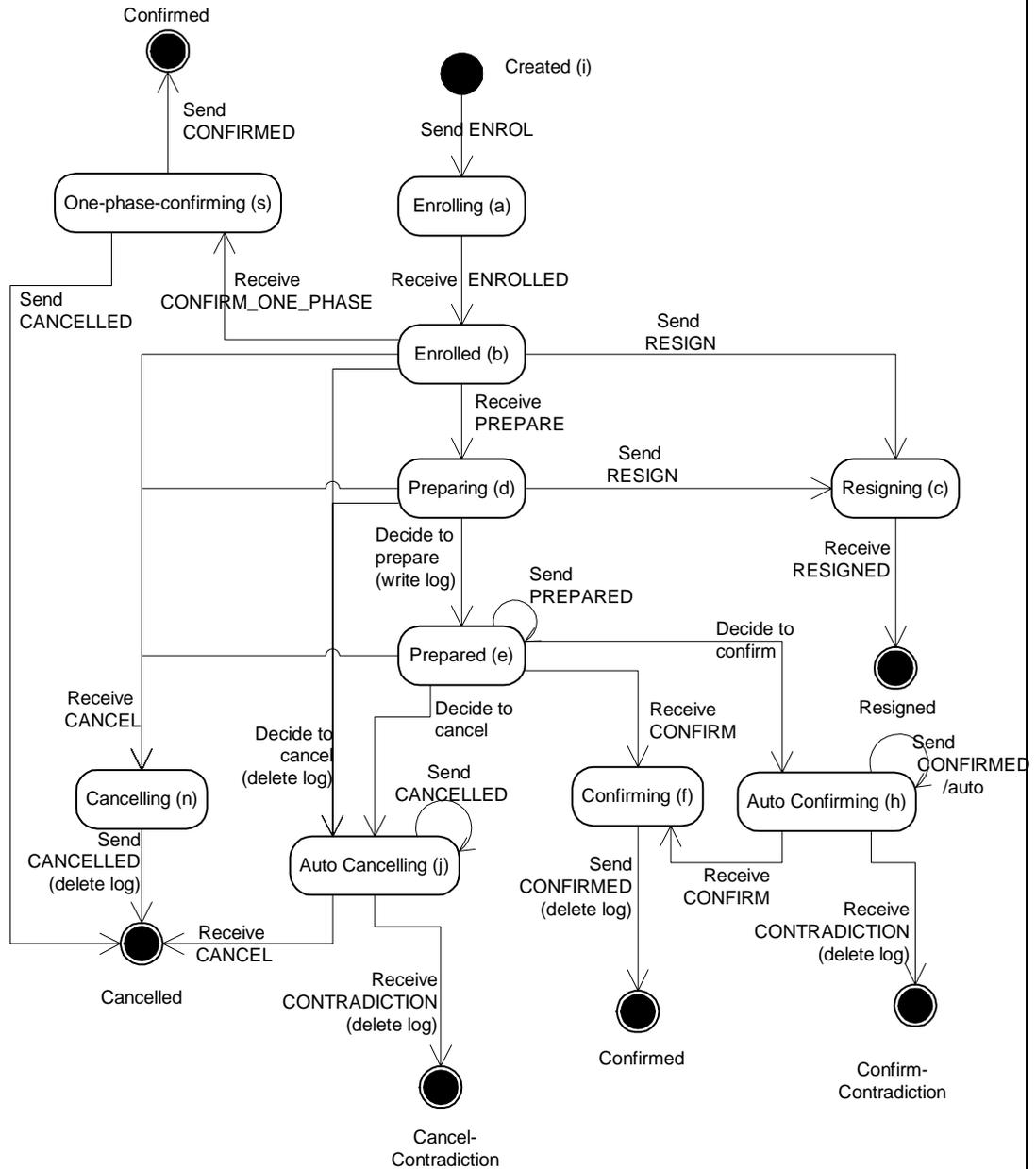
995



996

997

Figure 10 State diagram for Superior side of a Superior:Inferior relationship



998

999

Figure 11 State diagram for Inferior side of Superior:Inferior relationship

1000

Control of inferiors

1001

1002

1003

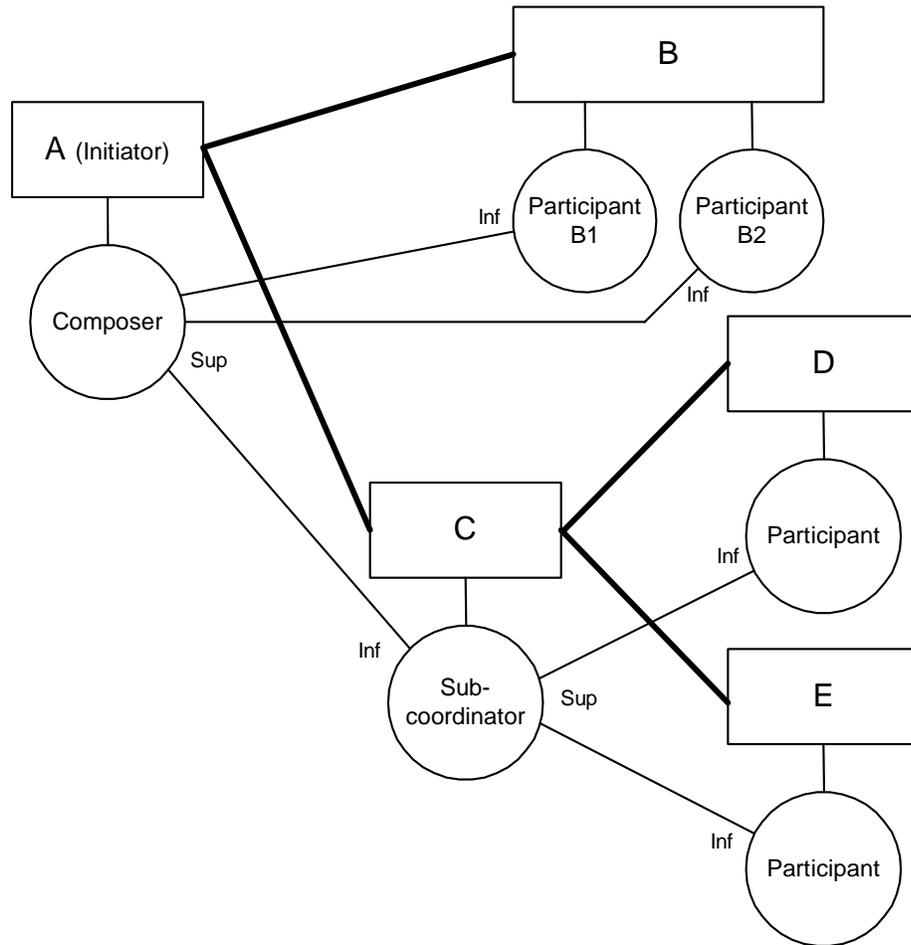
1004

1005

In the case as shown in Figure 12, where the CONTEXT has been propagated from one application element (A) to others (B, C, and from C to D,E), the determination of whether to create and enrol Inferiors is, in general, up to the receiving application element – this is an aspect of the fundamental autonomy of the parties involved in a business transaction. This

1006
1007
1008

autonomy may be constrained in particular situations, by inter-party agreement or where the application elements are in fact under common control.



1009

1010

Figure 12 Transaction tree showing various application:Participant relationships

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

The relationship between the application messages and either the propagated CONTEXT or the ENROL message(s) sent to the Superior is strictly part of the application protocol (or the application-with-BTP combination protocol). However defined, this allows the Superior-side application element to be aware of what application work will be confirmed or cancelled under the control of an Inferior. However, from the perspective of the Superior, and the application element controlling it, the Inferior is opaque – it is not in general possible for the Superior or its controlling application element to determine whether is a Sub-composer or Sub-coordinator (i.e. has Inferiors of its own) or is a Participant, with no further BTP relationships. Thus, if the Inferior is a Sub-composer or Sub-coordinator, the Superior has no visibility or control of its “grand-children” – the Inferiors of its Inferior (thus, in Figure 12, the Composer at A is unaware of D and E)

1024 The opacity of an Inferior does not however apply to the control exercised by the
1025 immediately controlling application element. An application element, acting as Terminator to
1026 a Decider (i.e. to a Composer or Coordinator), can be aware of and distinguish the different
1027 Inferiors enrolled with that Decider (i.e. Inferiors enrolled with the Decider in its role as
1028 Superior). (E.g.in Figure 12~~Figure 12~~, application element A knows of the Inferiors at C, B1
1029 and B2) This is especially the case for a Cohesion Composer, where the Terminator will be
1030 able to control which of the enrolled Inferiors of the Composer are eventually confirmed –
1031 more exactly, the application will have control of the confirm-set for the Cohesion. For an
1032 Atom Coordinator, visibility of the Inferiors is useful but less important, since no selection
1033 can be made among which will be in the confirm-set – for an Atom, all Inferiors are ipso
1034 facto members of the confirm-set.

1035
1036 For this control of the Inferiors to be useful, the Terminator application element will need to
1037 be able to associate particular parts of the application work with each Inferior. This can be
1038 achieved by various means. Taking the case of an application element controlling a Cohesion
1039 Composer:

- 1040
1041 a) The application element can create an Atom Sub-coordinator as an immediate
1042 Inferior of the Cohesion Composer and propagate the Sub-coordinator’s CONTEXT
1043 associate with application messages concerned with the particular part of the
1044 application work; any Inferiors (however many there may be) enrolled with Sub-
1045 coordinator can be assumed to be responsible for (some of) that part of the
1046 application, and the Terminator application element can just deal with the immediate
1047 Inferior of the Composer that it created.
1048 b) The application element can propagate the Composer’s own CONTEXT, and the
1049 receiving application element can create its own Inferior which will be responsible
1050 for some part of the application, and send ENROL to the Composer (as Superior).
1051 Application messages concerned with that part of the application are associated with
1052 the ENROL, and the Terminator application element can thus determine what the
1053 Inferior is responsible for.

1054
1055 In both cases, the means by which the application message and the BTP CONTEXT or
1056 ENROL are associated is ultimately application-specific. At the abstract message level, BTP
1057 defines the concept of transmitting “related” BTP and application messages – particular
1058 bindings to carrier protocols can specify interoperable ways to represent this relatedness. BTP
1059 messages, including CONTEXT and ENROL, can also carry “qualifiers” – extension fields
1060 that are not core parts of BTP or are not defined by BTP at all. The standard qualifier
1061 “inferior-name” or application-specific qualifiers can be used to associate application
1062 information and the BTP message, allowing a Terminator to determine which parts of the
1063 application work are associated with each Inferior.

1064
1065 These considerations about control of the Inferiors of a Decider also apply to the control of
1066 the Inferiors of a Sub-composer (and, again of less importance, a Sub-coordinator).

1067 Evolution of confirm-set

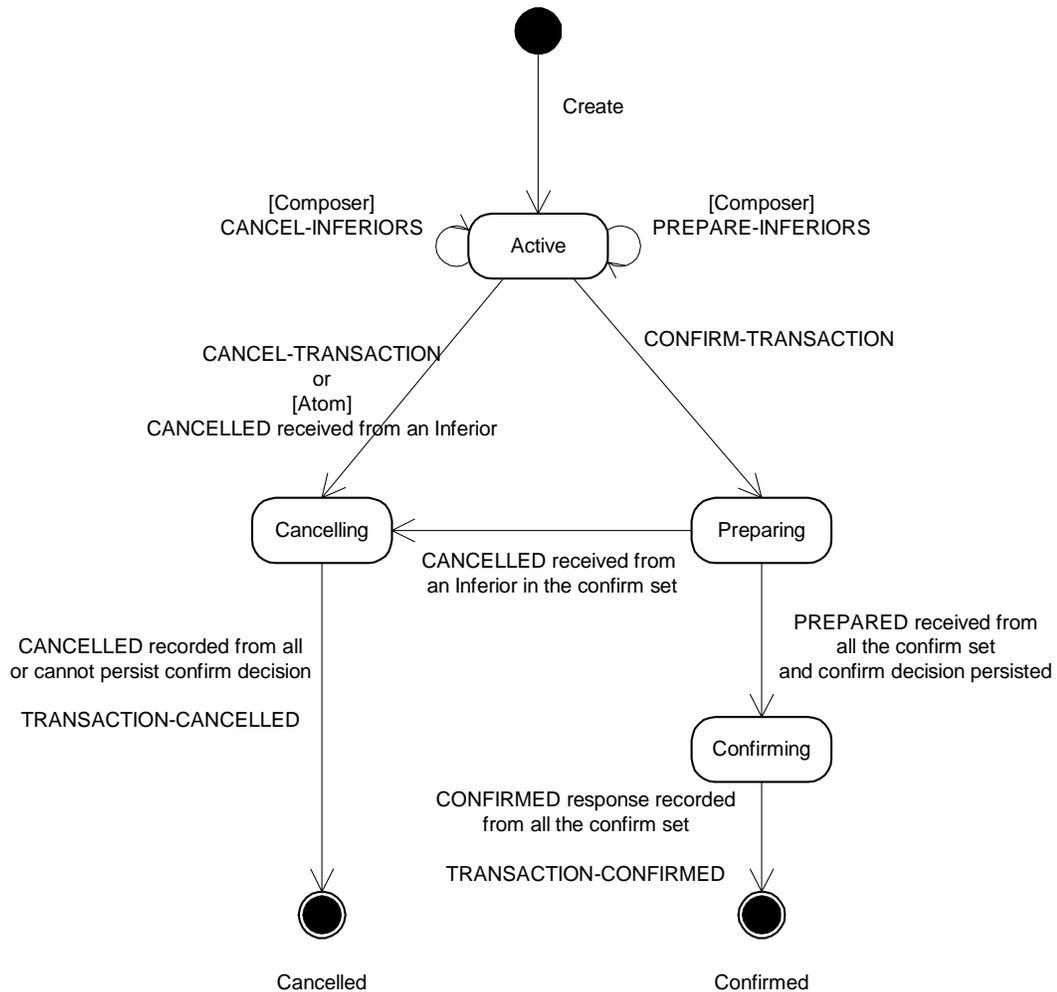
1068
1069

1070 As mentioned above, the set of Inferiors of a Cohesion that will eventually confirm is called
1071 the Confirm-set. The determination of the Confirm-set is made by the controlling application,
1072 but is affected by events from the Inferiors themselves. If the standard control relationship is
1073 used, the control of the Cohesion Composer is expressed by the Terminator:Decider
1074 exchanges, and the progressive determination of the confirm-set (its evolution) is effectively
1075 the event sequence for the Terminator:Decider relationship.

1076
1077 An Atom also has a confirm-set, but this always includes all the Inferiors and so does not
1078 evolve in the same way as Cohesion's. With some exceptions, the Terminator:Decider
1079 relationship is the same for Atom Coordinators as for Cohesion Composers; this section deals
1080 with both, noting the exceptions.

1081
1082 The event sequence for a Composer or Coordinator is summarised in the state diagram in
1083 Figure 13~~Figure 13~~. The step-by-step description refers to "Composer", but should be read as
1084 referring to Coordinators as well, unless stated otherwise.

1085
1086 Initially, the Composer is created (by the Factory, using BEGIN with no related CONTEXT),
1087 and has no Inferiors. The Composer is now in the active state.



1088

1089

Figure 13 State diagram for a Composer or Coordinator (i.e. Decider)

1090

While in the active state, the following may occur, in any order and with any repetition or overlapping:

1093

- Inferiors are enrolled – ENROL is received by the Composer – adding to the set of Inferiors of the Composer.

1094

1095

1096

- Inferiors may resign - RESIGN is received from an Inferior (see section Resignation below). The Inferior is immediately removed from the set of Inferiors, as if it had never been enrolled (a RESIGNED message may be sent to the Inferior, but it no longer “counts” in any of the Composer-wide considerations here.

1097

1098

1099

1100

1101

- CANCELLED may be received from an Inferior; there is no required immediate effect, but if this is a Coordinator the Atom will certainly cancel eventually (and an implementation may choose to initiate cancellation immediately).

1102

1103

1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149

- PREPARED may be received; there is no immediate effect
- The Terminator may issue PREPARE INFERIORS to the Composer (as Decider) for some subset of the Inferiors; PREPARE is sent to each and any of the Inferiors in the subset, excluding any from RESIGN, CANCELLED or PREPARED has been received; the sending of PREPARE will induce the Inferiors to reply with PREPARED, CANCELLED or RESIGN; when replies have been received from all, the Composer (as Decider) replies to the Terminator with INFERIOR STATUSES, reporting the replies received (which may in fact have been received before the PREPARE INFERIORS). PREPARE INFERIORS is not issued to Atom Coordinators.
- The Terminator may issue CANCEL INFERIORS to the Composer (as Decider) for some subset of the Inferiors; CANCEL is sent to each and any of the Inferiors in the subset, excluding any from RESIGN or CANCELLED has been received; the sending of CANCEL will normally induce the Inferiors to reply with CANCELLED – there are some exception cases; when replies have been received from all, the Composer (as Decider) replies to the Terminator with INFERIOR STATUSES, reporting the replies received. CANCEL INFERIORS is not issued to Atom Coordinators. CANCEL INFERIORS may be issued for an Inferior regardless of whether PREPARED has been received from it.
- The Terminator may issue REQUEST INFERIOR STATUSES to the Composer (as Decider) for all or some subset of the Inferiors; the Composer immediately replies with INFERIOR STATUSES, reporting the current state of the Inferiors as known to the Superior.

Eventually, the Terminator issues one of the completion messages – CANCEL TRANSACTION or CONFIRM TRANSACTION. These messages have a flag that determines whether the Terminator wishes to be informed of contradictory and heuristic decisions or hazards within the transaction – this affects when the reply from the Composer (as Decider) is sent to the Terminator. (See section “Autonomous cancel, autonomous confirm and contradictions” for details on contradictory and heuristic cases).

If the message is CANCEL TRANSACTION, CANCEL is sent to all Inferiors that it has not already been sent to, and from which neither RESIGN or CANCELLED have been received. If the Terminator indicates it does not want to be informed of contradictions, the Composer will immediately reply with TRANSACTION CANCELLED. Otherwise, if and when CANCELLED or RESIGN has been received from all Inferiors, the Composer replies to the Terminator with TRANSACTION CANCELLED; but if HAZARD or CONFIRMED is received from any Inferior, the reply is INFERIOR STATUSES, identifying which Inferior(s) had problems.

If the completion message is CONFIRM TRANSACTION, the inferiors-list parameter of the message defines the confirm-set. If the parameter is absent (which it must be for an atom

1150 Coordinator), then all Inferiors (excluding only those that have resigned) are the confirm-set;
1151 otherwise the confirm-set is only the Inferiors identified in the inferiors-list parameter (less
1152 any from which RESIGN has been received). The processing to arrive at the confirm decision
1153 is:

- 1154 • If at the point of receiving CONFIRM TRANSACTION or at any point before
1155 making the confirm decision (see below), CANCELLED is received, then the
1156 transaction is cancelled and processing continues as if CANCEL TRANSACTION
1157 had been received.
- 1158 • If there any Inferiors **not** in the confirm-set from which neither CANCELLED or
1159 RESIGN has been received, CANCEL is sent to them (this cannot happen for Atom
1160 Coordinators)
- 1161 • If initially or later, there is exactly one Inferior in the confirm-set, and either
1162 PREPARE has not been sent to it, or PREPARED has been received from it, then at
1163 implementation or configuration option, CONFIRM ONE PHASE can be sent to
1164 that Inferior. This delegates the confirm decision to the Inferior
- 1165 • If at any point, RESIGN is received from an Inferior, it is immediately removed from
1166 the confirm-set (this may trigger the decision making)
- 1167 • If there are any Inferiors in the confirm-set from which none of PREPARED,
1168 CANCELLED has been received and to which PREPARE has not yet been sent,
1169 PREPARE is sent to that Inferior
- 1170 • If initially or later, PREPARED has been received from all Inferiors in the confirm-
1171 set, the Composer *makes the confirm decision*; it persists (or attempts to persist)
1172 information identifying the Inferiors in the confirm-set; if this fails, the transaction is
1173 cancelled and processing continues as if CANCEL TRANSACTION had been
1174 received; if the information is persisted, the confirm decision has been made.

1175
1176 When the confirm decision is made, CONFIRM is sent to all the Inferiors in the confirm-set.
1177 And, if on the CONFIRM TRANSACTION the Terminator indicated it did not wish to be
1178 informed of contradictions, TRANSACTION CONFIRMED is sent to the Terminator.
1179 If the Terminator indicated it wanted to be informed of contradictions, the Composer replies
1180 to it with TRANSACTION CONFIRMED if and when CONFIRMED has been received
1181 from all the Inferiors in the confirm-set and CANCELLED or RESIGN has been received
1182 from any other Inferiors. If other replies (CANCELLED from a confirm-set Inferior,
1183 CONFIRMED from other Inferiors, HAZARD from any) are received, the reply to the
1184 Terminator is INFERIOR STATUSES, identifying which Inferior(s) had problems.

1185 1186 Confirm-set of intermediates

1187
1188 An Intermediate, that is a Superior that is also an Inferior, also has a confirm-set, but this is
1189 controlled rather differently to the top-most Superior (Decider) described above.

1190
1191 As an Inferior, the interface between the application and BTP elements is not fully defined in
1192 this specification. However, within the standard control relationship, issuing BEGIN with a
1193 related CONTEXT to a Factory will cause the creation of a Sub-coordinator or Sub-composer
1194 (depending on whether the BEGIN parameter asked for atomic or cohesive behaviour).
1195 Initially, of course, the new Intermediate has no Inferiors – however, unlike a Participant (in

1196 the strict sense of the term), it has a “superior-address” to which ENROL can be sent to enrol
1197 Inferiors. This address is a field of the new CONTEXT.

1198
1199 The behaviour of the Intermediate towards its Inferiors, during the active phase, is basically
1200 the same as for the Decider:

- 1201 • ENROL messages can be received, adding a new Inferior
- 1202
- 1203 • Inferiors may resign - RESIGN is received from an Inferior. The Inferior is
1204 immediately removed from the set of Inferiors
- 1205
- 1206 • CANCELLED may be received from an Inferior
- 1207
- 1208 • PREPARED may be received from an Inferior
- 1209

1210 In some circumstances, receipt of an incoming message allows an Intermediate to
1211 determine that a state change for the whole transaction node takes place. The
1212 Intermediate is able to send messages to its Superior at its own initiative (whereas a
1213 Decider can only respond to a received message from the Terminator), so the receipt of
1214 a message from an Inferior can trigger the sending of messages. This is especially the
1215 case if the Intermediate knows (from application knowledge, perhaps involving
1216 received or sent CONTEXT_REPLY messages) that there will be no further
1217 enrolments. In particular:

- 1218
- 1219 • If CANCELLED is received from an Inferior, and this is a Sub-coordinator, the Sub-
1220 coordinator can itself cancel - CANCEL is sent to other Inferiors, and CANCELLED
1221 to the Superior
- 1222 • If RESIGN is received from the only Inferior and there will be no other enrolments,
1223 the Intermediate can itself resign, sending RESIGN to the Superior
- 1224 • If PREPARED is received from the Superior, it is known there will be no other
1225 enrolments and this is a Sub-coordinator, the Sub-coordinator can become prepared
1226 (assuming successful persistence of the appropriate information) and send
1227 PREPARED to the Superior.
- 1228

1229 For a Sub-composer, application logic will invariably be involved in determining what effect
1230 a CANCELLED and PREPARED from an Inferior have – though in a real implementation,
1231 this logic may be delegated to the BTP-support software.

1232
1233 The Intermediate may initiate cancellation or the two-phase outcome exchange, either as a
1234 result of receiving the corresponding message (CANCEL, PREPARE) from the Superior, or
1235 triggered by its own controlling application element. For a Sub-composer, this may be partial
1236 - a Sub-composer might be instructed by the application element to cancel some Inferiors and
1237 send PREPARE to others. Receipt of PREPARE from the Superior will often have a similar
1238 effect to a Decider receiving CONFIRM_TRANSACTION – PREPARE is propagated to all
1239 Inferiors that have not indicated they are PREPARED. However, exactly what happens on
1240 receiving PREPARE will depend on the application – receipt of the PREPARE may be
1241 visible to the application element and cause it to initiate further application activity (perhaps

1242 causing enrolment of new Inferiors) before it is determined whether to propagate PREPARE,
1243 and with a Sub-composer, some of the Inferiors may be instructed to cancel instead.

1244
1245 Assuming the Intermediate does not cancel as a whole (in which case CANCEL would be
1246 sent to all Inferiors), the Intermediate will at some point attempt to become prepared. If it is a
1247 Sub-coordinator, this will require that PREPARED has been received from all Inferiors. For a
1248 Sub-composer, application logic will determine from which Inferiors PREPARED is
1249 required, with the others being cancelled. In either case, the Intermediate will persist the
1250 information about the Inferiors that are to be in the confirm-set and about the Superior, if this
1251 persisting is successful, send PREPARED to its own Superior.

1252
1253 If CANCEL is subsequently received from the Superior, this is propagated to all the Inferiors
1254 and the persistent information removed (or effectively removed as far as recovery is
1255 concerned). It is not important which order this is done in, since the recovery sequence will
1256 ensure that a cancel outcome is eventually delivered anyway.

1257
1258 If CONFIRM is received from the Superior (which can only be after sending PREPARED to
1259 the Superior), this is likewise propagated to the Inferiors. For a Sub-coordinator, CONFIRM
1260 is invariably sent to all Inferiors. However, for a Sub-composer it is possible further
1261 application logic intervenes and some of the Inferiors are rejected from the confirm-set at this
1262 late stage. (This can only occur when the application work, as defined by the contract to the
1263 Superior, can be performed by some sub-set of the Inferiors.) The Intermediate may, but is
1264 not required to, change the persistent information to reflect the confirm outcome (though a
1265 Sub-composer that selects only some Inferiors probably will need to re-write the information
1266 to ensure the correct subset are confirmed despite possible failures). If the information is not
1267 changed, then, on recovery, the Intermediate will find itself to be in a prepared state and will
1268 interrogate the Superior to re-determine the outcome. If the information is changed, a
1269 recovered Intermediate can immediately continue with ordering confirmation to its Inferiors.

1270
1271 If CONFIRM ONE PHASE is received from the Superior, either before or after the
1272 Intermediate has become PREPARED, the effect is very similar to a Decider receiving
1273 CONFIRM TRANSACTION. If there is only one Inferior, the CONFIRM ONE PHASE
1274 may be propagated to that Inferior. Otherwise, the Intermediate behaves as a Decider, making
1275 a confirm decision if it can.

1276
1277 If one or more Inferiors make contradictory autonomous decisions, or HAZARD is received
1278 from an Inferior, the Intermediate may report this to the Superior using HAZARD. However,
1279 BTP does not require this. Since the Superior may be owned and controlled by a different
1280 organisation, there may be business reasons not to report such problems.

1281 Optimisations and variations

1282 1283 Spontaneous prepared

1284
1285 As described above, before a Superior can order confirmation to an Inferior, the Inferior must
1286 become “prepared”, meaning that it is ready to confirm or to cancel as it so ordered and send
1287 the PREPARED message as a report of this. In the conventional message sequence, as shown

1288 above, the Inferior attempts to become prepared when it receives a PREPARE message from
1289 the Superior. The PREPARE in turn is sent by the Superior when it receives an appropriate
1290 request from its controlling application (or from its own Superior, if there is one). The
1291 application controlling the Superior will request the sending of PREPARE when it
1292 determines that no further application work associated with this Inferior (or, perhaps with the
1293 whole business transaction) will occur.

1294
1295 However, for some applications, the application element controlling the Inferior will know
1296 that the application work for which the Inferior will be responsible is complete before a
1297 PREPARE is sent from the Superior. In fact, because the application element has autonomy
1298 in determining how application work is to be allocated to Inferiors, it is possible for the
1299 Inferior-side application element to know the work is complete for a particular Inferior
1300 when Superior-side application element will be sending more message to the Inferior-side.
1301 (The future work will, probably, require the enrollment of additional Inferiors.)

1302
1303 BTP consequently allows the application element controlling an Inferior to cause the Inferior
1304 to become prepared, and to send PREPARED to the Superior without PREPARE having been
1305 received from the Superior. From the perspective of the BTP Superior the Inferior sends
1306 PREPARED spontaneously. Apart from this, a spontaneous PREPARED message is the same
1307 as, and has the same effect and implications as one induced by a PREPARE message.

1308

One-shot

1309

1310
1311 In the “conventional” message sequence shown above and assuming the Initiator, Terminator
1312 and Coordinator on the one side, and “Service”, Enroller and Participant on the other are
1313 located within their respective parties, there are eight messages passed in one direction or the
1314 other between the two parties. There are four round-trip exchanges: the application request
1315 and response exchange, the ENROL/ENROLLED exchange (going in the opposite direction
1316 and overlapped with the application exchange), then PREPARE/PREPARED and the
1317 CONFIRM/CONFIRMED. However, if the application exchange is a single
1318 request/response, it is possible to reduce these eight to two round-trips– the first of which
1319 merges the first three of the conventional sequence. The fundamental two-phase nature of
1320 BTP (or any coordination mechanism) means there have to be at least two round trips – one
1321 before the confirm-or-cancel decision is made at the Superior, one after. This merging of the
1322 exchanges is termed “one-shot”, as it requires only one exchange to take the relationship from
1323 non-existent to waiting for the confirm-or-cancel decision.

1324

1325 Figure 14~~Figure 14~~ shows a typical “one-shot” message sequence. The diagram distinguishes
1326 an additional aspect of the application elements, labelled “context-handler”. This is not a role
1327 in the BTP model, but is used only to distinguish a set of responsibilities and actions. In a real
1328 implementation these might be performed by the user application itself, or might be
1329 performed by the BTP-supporting infrastructure on the path between the application
1330 elements. (Figure 9~~Figure 9~~ could be redrawn to show the context-handlers, but to no
1331 particular benefit) As in the conventional case, the CONTEXT is sent related to the
1332 application request (the creation of the CONTEXT by the Factory is not shown and is the

1333 same as the conventional case). The “context-handler” is aware of the sending of the
1334 CONTEXT.

1335

1336 On the responder (service side), however, when the application element creates the Inferior,
1337 the ENROL is not sent immediately, but retained. The application performs the “provisional
1338 effect” implied by the received message and the Inferior becomes prepared and issues a
1339 PREPARED message, which is also retained. When the application response is available, it is
1340 sent with the retained messages and the CONTEXT_REPLY (which indicates that the related
1341 ENROL will complete the enrolments implied by the earlier transmission of the CONTEXT.

1342

1343 When this group of messages is received by the context-handler on the client side, the
1344 contained ENROL and PREPARED messages are forwarded to the Superior (whose address
1345 was on the original CONTEXT and so is known to the context-handler). An ENROLLED
1346 message is sent back to the context-handler, assuring it that the enrolment was successful and
1347 the application can progress. If enrollment fails and the business transaction is atomic,
1348 confirmation must be prevented – this responsibility falls on the context-handler and the
1349 client application, since the failure of the enrolment implies that Superior itself is
1350 inaccessible. If enrolment fails and the business transaction is a cohesion, the appropriate
1351 response is a matter for the application.

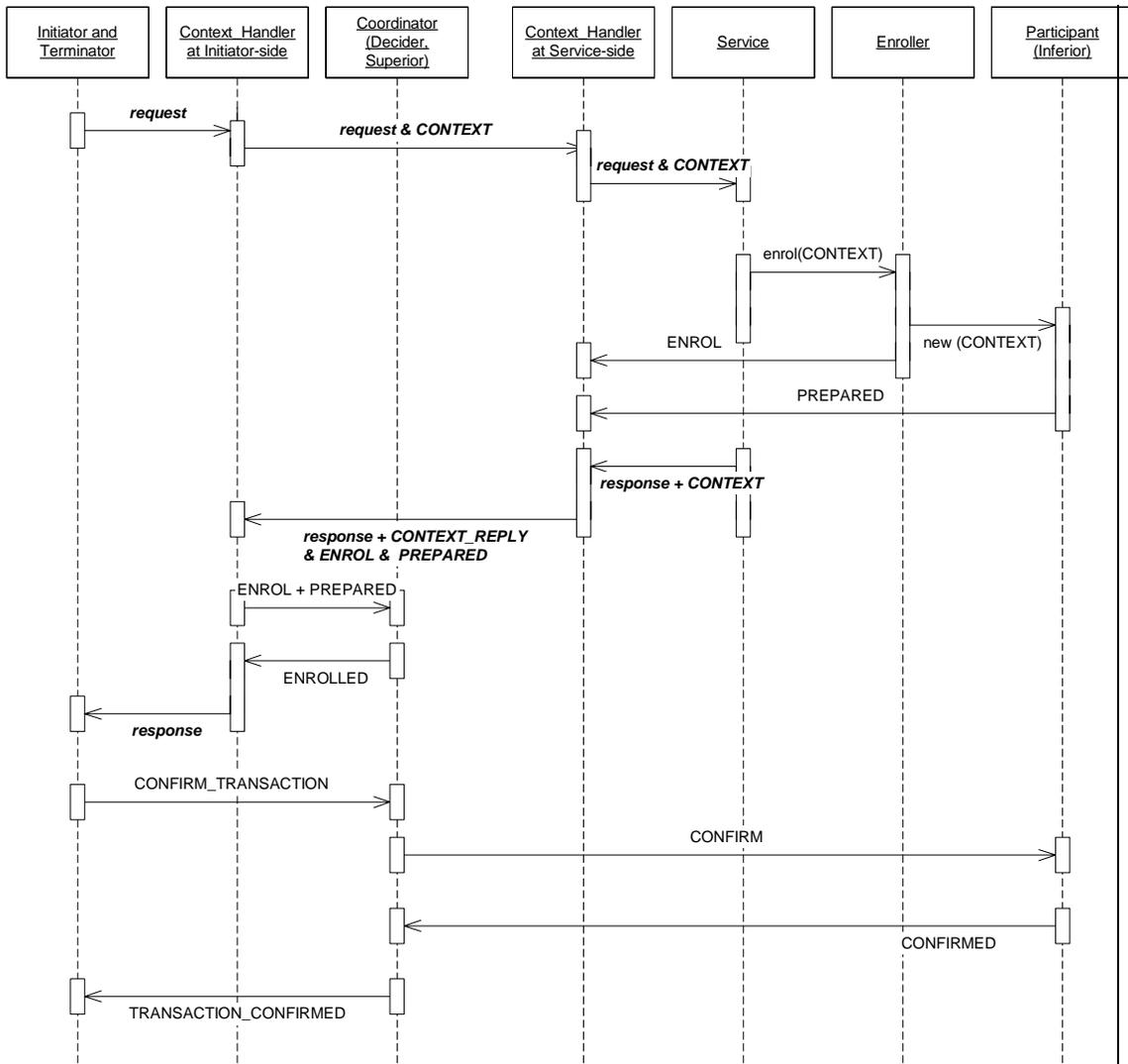
1352

1353 With “one-shot”, if there are multiple Inferiors created as a result of a single application
1354 message, there is an ENROL and PREPARED message for each one sent related with the
1355 CONTEXT_REPLY. If an operation fails, a CANCELLED message may be sent instead of a
1356 PREPARED – if the Superior is atomic, this will ensure it cancels, if cohesive, the client
1357 application will be aware of this and behave appropriately.

1358

1359 Whether the “one-shot” mechanism is used is determined by the implementation on the
1360 responding (Inferior) side. This may be subject to configuration and may also be constrained
1361 by the application or by the binding in use.

1362



1363

1364

Figure 14 – A message sequence showing the “one-shot” optimisation

1365

1366

Resignation

1367

After an Inferior is enrolled, it may be determined that the application work it is responsible for has no real effect – more exactly, that the counter-effect, if cancelled, and the final effect, if confirmed, will be identical. In such a case the Inferior can effectively un-enrol itself by sending a RESIGN message to the Superior. This can be done “spontaneously” (as far as BTP is concerned) or as a response to a received PREPARE message. It cannot be done after the Inferior has become prepared.

1368

1369

1370

1371

1372

1373

1374

An Inferior from which RESIGN has been received is not considered an Inferior in discussion of the confirm-set – the phrase “remaining Inferiors” is used to mean only non-resigned Inferiors.

1375

1376

1377 One-phase confirmation
1378 If a Coordinator or Composer that has been requested to confirm has only one (remaining)
1379 Inferior in the confirm-set, it may delegate the confirm-or-cancel decision to that Inferior, just
1380 requesting it to confirm rather than performing the two-phase exchange. This is done by
1381 sending the CONFIRM ONE PHASE message. Unlike the two-phase exchange
1382 (PREPARED received, CONFIRM sent), it is possible with CONFIRM ONE PHASE for a
1383 failure to occur that leads to the original Coordinator or Composer (and its controlling
1384 application element – the Terminator) being uncertain whether the outcome was confirmation
1385 or cancellation.

1386
1387 Autonomous cancel, autonomous confirm and contradictions
1388 As described above, BTP does not require a Participant, while it is responsible for holding
1389 application resources such that can be confirmed or cancelled, to use any particular
1390 mechanism for maintaining this state. A Participant that “becomes prepared” may choose to
1391 let the “provisional effect” be identical to the “final effect”, and hold a compensating
1392 “counter effect” ready to implement cancellation; or it may make the provisional effect
1393 effectively null, and only perform the real application work as the final effect if confirmed; or
1394 the “provisional effect” may involve performance of the application work and locking
1395 application data against other access; or other patterns, as may be constrained or permitted by
1396 the application.

1397
1398 Although a Participant is not required to lock data (as would be the case with some other
1399 transaction specifications) on becoming prepared, it is nevertheless in a state of doubt, and
1400 this doubt may have application or business implications. Accordingly it is recognised that a
1401 Participant (or, rather the business party controlling the application element and the
1402 Participant) may need to limit the promise made by sending PREPARED, and retain the right
1403 to apply its own decision to confirm or cancel to the Participant and the application effects it
1404 is responsible for. This is described as an “autonomous” decision. It is closely analogous to
1405 the heuristic decisions recognised in other transaction specifications. The only difference is
1406 the conceptual one that heuristic decisions are typically considered to occur only as a result of
1407 rare and unpredictable failure, whereas BTP recognises that the right to take an autonomous
1408 decision may be critical to the willingness of a business party to be involved in the business
1409 transaction at all. BTP therefore allows Participants (and all Inferiors) to indicate that there
1410 are limits on how long they are willing to promise to remain in the prepared state, and that
1411 after that time they may invoke their right of taking an autonomous decision.

1412
1413 Taking an autonomous decision will of course run the risk of breaking the intended
1414 consistency of outcome across the business transaction, if the autonomous decision of the
1415 Inferior contradicts the decision (for this Inferior) made by the Superior. The Superior will
1416 have received the PREPARED message and thus be permitted to make a confirm decision
1417 (directly, or through exchanges with a Terminator application element or with its own
1418 Superior). An Inferior taking an autonomous decision informs the Superior by sending
1419 CONFIRMED or CANCELLED, as appropriate, without waiting for an outcome order from
1420 the Superior. This may cross the outcome message from the Superior, or the Superior may not
1421 make its decision till later. If the decisions agree, the normal CONFIRM or CANCEL
1422 message is sent. In the case of CANCEL, this completes the relationship – the CANCEL and

1423 CANCELLED messages acknowledge each other, regardless of which travels first. In the
1424 case of CONFIRM, another CONFIRMED message is needed.

1425
1426 If the Superior's decision is contradicted by the autonomous decision, the Superior may need
1427 to record this, report it to management systems or inform the Terminator application or its
1428 own Superior. When this has been done (details are implementation-specific, but may be
1429 constrained by the application), the Superior sends a CONTRADICTION message to the
1430 Inferior. If an outcome message was sent earlier (crossing the announcement of the
1431 autonomous decision), the Inferior will already know there was a contradiction, but the
1432 receipt of the CONTRADICTION message informs the Inferior that the Superior knows and
1433 has done whatever it considers necessary to cope.

1434
1435 As mentioned, BTP allows an Inferior to inform the Superior, with a qualifier on the
1436 PREPARED message, that the promise to remain in the prepared state will expire. In turn this
1437 allows the application on the Superior side to avoid risking a contradictory decision by
1438 making and sending its own decision in time. The Superior side can also indicate, with
1439 another qualifier, a minimum time for which it expects the prepared promise to remain valid.

1440
1441 As well as deliberate and forewarned autonomous decisions, BTP recognises that failures and
1442 exceptional conditions may force unplanned autonomous decisions. In the protocol sequence
1443 these are treated exactly like planned autonomous decisions – if they contradict, the Superior
1444 will be informed and a CONTRADICTION message sent to the Inferior.

1445
1446 Autonomous decisions, planned or unplanned, are equivalent to the heuristic decisions of
1447 other transaction systems. The term is avoided in BTP since it may carry implications that it
1448 only occurs in an unplanned manner.

1449 **Recovery and failure handling**

1450 1451 **Types of failure**

1452
1453 BTP is designed to ensure the delivery of a consistent decision for a business transaction to
1454 the parties involved, even in the event of failure. Failures can be classified as:

1455
1456 **Communication failure:** messages between BTP actors are lost and not delivered. BTP
1457 assumes the carrier protocol ensures that messages are either delivered correctly (without
1458 corruption) or are lost, but does not assume that all losses are reported nor that messages
1459 sent separately are delivered in the order of sending.

1460
1461 **Node failure (system failure, site failure):** a machine hosting one or more BTP actors
1462 stops processing and all its volatile data is lost. BTP assumes a site fails by stopping – it
1463 either operates correctly or not at all, it never operates incorrectly.

1464
1465 Communication failure may become known to a BTP implementation by an indication from
1466 the lower layers or may be inferred (or suspected) by the expiry of a timeout. Recovery from
1467 a communication failure requires only that the two actors can again send messages to each
1468 other and continue or complete the progress of the business transaction.

1469
1470 A node failure is distinguished from communication failure because there is loss of volatile
1471 state. To ensure consistent application of the decision of a business transaction, BTP requires
1472 that some state information will be persisted despite node failure. Exactly what real events
1473 correspond to node failure but leave the persistent information undamaged is a matter for
1474 implementation choice, depending on application requirements; however, for most
1475 application uses, power failure should be survivable (an exception would be if the data
1476 manipulated by the associated operations was volatile). In all cases, there will be some level
1477 of event sufficiently catastrophic to lose persistent information and the ability to recover–
1478 destruction of the computer or bankruptcy of the organisation, for example.

1479
1480 Recovery from node failure involves recreating an accessible communications endpoint in a
1481 network node that has access to the persistent information for incomplete transactions. This
1482 may be a recreation of the original actor using the same addresses; or using a different
1483 address; or there may be a distinct recovery entity, which can access the persistent data, but
1484 has a different address; other implementation approaches are possible. The recovered, and
1485 possibly relocated actor may or may not be capable of performing new application work
1486 Restoration of the actor from persistent information will often result in a partial loss of state,
1487 relative to the volatile state reached before the failure. In some states, there may be total loss
1488 of knowledge of the business transaction, including particular Superior:Inferior relationships.
1489 After recovery from node failure, the implementation behaves much as if a communication
1490 failure had occurred.

Persistent information

1491
1492
1493
1494 BTP **requires** that certain state information is persisted – these are information that records
1495 an Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s
1496 autonomous decision . Requiring the first two to be persistent ensures that a consistent
1497 decision can be reached for the business transaction and that it is delivered to all involved
1498 nodes, despite failure. Requiring an Inferior’s autonomous decision to be persistent allows
1499 BTP to ensure that, if the autonomous decision is contradictory (i.e. opposite to the decision
1500 at the Superior), the contradiction will be reported to the Superior, despite failures.

1501
1502 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
1503 active state (unlike many transaction protocols, where a communication or node failure in
1504 active state would invariably cause rollback of the transaction). Recovery in the active state
1505 may require that the application exchange is resynchronised as well – BTP does not directly
1506 support this, but allows continuation of the business transaction if the application desires it.
1507 Apart from the (optional) recovery in active state, BTP follows the well-known presume-
1508 abort model – it is only **required** that information be persisted when decisions are made (and
1509 not, for example, on enrolment). This means that on recovery one side may have persistent
1510 information while the other does not. This occurs, among other cases, when an Inferior has
1511 decided to be prepared but the Superior never confirmed (so the decision is “presumed” to be
1512 cancelled), and when the Superior did confirm, the Inferior applied the confirmation and
1513 removed its persistent information but the acknowledgement message (CONFIRMED) was
1514 never received by the Superior.

1515
1516 Information to be persisted when an Inferior decides to be prepared has to be sufficient to re-
1517 establish communication with the Superior, to apply a confirm decision and to apply a cancel
1518 decision. It will thus need to include the addressing and identification information for the
1519 Superior. The information needed to apply the confirm or cancel decision will depend on the
1520 application and the associated operations.

1521
1522 A Superior must persist the corresponding information to allow it to re-establish
1523 communication with the Inferior – that is the addressing and identification information for the
1524 Inferior. When it must persist this information depends on its position within the transaction
1525 tree. If it is the top of the tree – i.e. it is the Decider for the business transaction -- it need only
1526 persist this information if and when it makes a decision to confirm (and, for a Cohesion, only
1527 if this Inferior is in the confirm-set). A Superior that is an intermediate in the tree – i.e. it is an
1528 Inferior to some other Superior –must persist the information about each of its own Inferiors
1529 as part of (or before) persisting its own decision to be prepared. For such an intermediate, the
1530 “decision to confirm” as Superior is made when either CONFIRM is received from its
1531 Superior or it makes an autonomous decision to confirm. If CONFIRM is received, the
1532 persistent information may be changed to show the confirm decision, but alternatively, the
1533 receipt of the CONFIRM can be treated as the decision itself and the CONFIRM message
1534 propagated to the Inferiors without changing the persistent information. If the persistent
1535 information is left unchanged and there is a node failure, on recovery the entity (as an
1536 Inferior) will be in a prepared state, and will rediscover the confirm decision (using the
1537 recovery exchanges to its Superior) before propagating it to its Inferior(s).

1538
1539 Since BTP messages may carry application-specified qualifiers, and the BTP messages may
1540 be repeated if they are lost in transit (see next section), the persistent information may need to
1541 include sufficient to recreate the qualifiers, to allow them to be resent with their carrying BTP
1542 message. This applies both to qualifiers on PREPARED (which would be persisted by the
1543 Inferior) and on CONFIRM (which would be persisted by the Superior).

1544
1545 In some cases, an implementation may not need to make an active change to have a persistent
1546 record of a decision, provided that the implementation will restore itself to the appropriate
1547 state on recovery. For example, an implementation that, as Inferior, always used the default-
1548 is-cancel mechanism, and recorded the timeout (to cancel) in the persistent information on
1549 becoming prepared, and always updated or removed that record when it applied a confirm
1550 instruction could treat the presence of an expired record as effectively a record of an
1551 autonomous cancel decision.

1552 1553 Recovery messages

1554
1555 Once the Superior:Inferior relationship has entered the completion phase – BTP does not
1556 generally use special messages in recovery, but merely permits the resending of the previous
1557 message – thus, for example, PREPARE, PREPARED, CANCEL, CONFIRM can all be sent
1558 repeatedly. Resending the previous message means a possible loss of the original message
1559 may be invisible to the receiver. The trigger for this re-sending is implementation dependent
1560 – a reported communication failure, a timeout expiry while waiting for a reply, the re-

1561 establishment of communications or the general restoration of function after a node failure
1562 are all possible triggers. An incoming repetition of the last message received, if it has already
1563 been replied to (e.g. receiving PREPARE after PREPARED has been sent), should normally
1564 trigger a resending of the last message sent – since that sent message may have got lost.⁴
1565

1566 While in the active phase – i.e. prior to entering completion – there is no appropriate last
1567 message that can be sent. However, for active-phase recovery there needs to be some way for
1568 the BTP actors to determine that the peer is still there and still aware of the Superior:Inferior
1569 relationship. In this case, the peers can interrogate each other using the INFERIOR_STATE
1570 or SUPERIOR_STATE messages, informing the peer of their own state and requesting a
1571 response – which may be the opposite message, or one of the main BTP messages (which
1572 perhaps had been lost). If it is another SUP|INFERIOR_STATE message, that reply does not
1573 ask for a response. Receiving a SUP|INFERIOR_STATE messages that asks for a response
1574 does not require an immediate response – especially if an implementation is waiting to
1575 determine a decision (perhaps because it is itself waiting for a decision from elsewhere), an
1576 implementation may choose not to reply until it wishes too.
1577

1578 The SUP|INFERIOR_STATE messages are also used as replies when the receiver of **any** of
1579 the Superior:Inferior message has determined that there is no corresponding state information
1580 – the targeted Superior or Inferior does not exist (or is known to have completed and is no
1581 longer an active entity). The SUP|INFERIOR_STATE messages with a status of “unknown”
1582 is the indication that the state information does not exist.
1583

1584 The SUP|INFERIOR_STATE messages are also available as replies to any Superior:Inferior
1585 message in the (transient, one hopes) case where, after failure an implementation cannot
1586 currently determine whether the persistent information exists or not, or what its state is, and
1587 so cannot give a definitive answer. The SUP|INFERIOR_STATE messages with a status of
1588 “inaccessible” is the indication that the existence of state information cannot be determined.
1589 The receiver of such a message should normally treat it as a “retry later” suggestion.
1590

1591 Redirection

1592
1593 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
1594 that there are cases where one side will attempt to re-establish communication when there is
1595 no persistent information for the relationship at the far-end, because that side either never
1596 reached a state where the state was persisted, or had been persisted, but then progressed to
1597 remove the state information. In such cases, it is important the side that is attempting
1598 recovery can distinguish between unsuccessful attempts to connect to the holder of the
1599 persistent information and when the information no longer exists. If the peer information does
1600 not exist, the side that is attempting recovery can draw appropriate conclusions (that the peer
1601 either was never prepared, never confirmed or has already completed) and complete its part
1602 of the transaction; if it merely fails to get through, it is stuck in attempting recovery.
1603

⁴ BTP's capability of binding to alternative carrier protocols is part of the motivation for not having a distinct recovery message sequence, since the carrier binding does not necessarily have a well-defined communication failure indication.

1604 Two mechanisms are provided to assist implementation flexibility while allowing completion
1605 of Superior:Inferior relationships when only one side has any persistent information. The
1606 mechanisms are:

- 1607 o Address fields which provide the address that will be used by the peer to send
1608 messages to an actor (effectively a “callback address”) can be a set of
1609 addresses, which are alternatives, one of which is chosen as the target address
1610 for the future message. If the sender of that message finds the address does
1611 not work, it can try a different alternative.
- 1612 o The REDIRECT message can be used to inform the peer that an address
1613 previously given is no longer valid and to supply a replacement address (or
1614 set of addresses). REDIRECT can be issued either as a response to receipt of
1615 a message or spontaneously.

1616
1617 The two mechanisms can be used in combination, with one or more of the original set of
1618 addresses just being a redirector, which does not itself ever have direct access to the state
1619 information for the transaction, but will respond to any message with an appropriate
1620 REDIRECT.

1621
1622 REDIRECT as a message is only used on the Superior:Inferior relationship, where each side
1623 holds the address of the other. On the other relationships (e.g. Terminator:Decider), one side
1624 (e.g. Terminator) has the address of the other, and initiates all the message exchanges.
1625 However, the entity whose address is known to the other may itself move - e.g. if a
1626 Coordinator, which will be both Decider and Superior changes its address as a Superior, it
1627 will probably change its address as a Decider too. In this case, a FAULT reply to a
1628 misdirected message can be used, assuming there is some entity available at, or on the path to
1629 the old address that understands BTP sufficiently to provide the redirection information.

1630
1631 Some implementations, in which a single addressable entity with one, constant address deals
1632 with all transactions, distinguishing them by identifier, will not need to supply “backup”
1633 addresses (and would only use REDIRECT if permanently migrated).

1634 Terminator:Decider failures and transaction timelimit

1635
1636
1637 BTP does not provide facilities or impose requirements on the recovery of
1638 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
1639 may survive failures (by retaining knowledge of the Decider’s address and identifier), but this
1640 is an implementation option. Although a Decider (if it decides to confirm) will persist
1641 information about the confirm decision, it is not required, after failure, to remain accessible
1642 using the address it originally gave to the Initiator (and used by the Terminator). Any such
1643 recovery is an implementation option.

1644
1645 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
1646 thus no way of detecting that a Terminator has failed. The Decider always has the right to
1647 initiate cancellation, but if the application (Terminator) and the Decider have different views
1648 about how long a “long time” is, then either the Decider might wait unnecessarily for a
1649 completion request (e.g. CONFIRM_TRANSACTION) that will never arrive, or it might

1650 initiate cancellation while the application is still active. To avoid these irritations, a standard
1651 qualifier “Transaction timelimit” can be used (by the Initiator) to inform the Decider when it
1652 can assume the Terminator will not request confirmation and so it (the Decider) should
1653 initiate cancellation.

1654

1655 Contradictions and hazard

1656 As described above (see “Autonomous cancel, autonomous confirm and contradictions”), in
1657 some circumstances an Inferior may apply a decision that is contradictory to the decision of
1658 the Superior. This can occur in a semi-planned manner, when the Inferior has announced a
1659 timeout on the PREPARED message but no outcome message has been received, or as a
1660 result of an exceptional condition that forces the Inferior to break the promise implicit in
1661 PREPARED, regardless of timers. In both cases, this is considered an autonomous decision
1662 by the Inferior. An autonomous decision, of itself, does not imply a contradiction – it only
1663 results in a contradiction if the decision is opposite to that of the Superior (in the case of a
1664 cohesive Superior, opposite to the decision that applies to this Inferior).

1665

1666 In order to ensure that a contradiction is detected despite node and communication failures, it
1667 is required that information about the taking of the autonomous decision be persisted until a
1668 BTP message received from the Superior indicates either that there was no contradiction (the
1669 decisions were in line – CANCEL is received after an autonomous cancel or CONFIRM is
1670 received after an autonomous confirm) or that the Superior is aware of the contradiction
1671 (CONTRADICTION is received). Note that the Inferior will become aware of the fact of the
1672 contradiction when it receives the “wrong” message, but must retain the record of its own
1673 decision until it receives the CONTRADICTION message, which tells it the Superior knows
1674 too.

1675

1676 The Superior’s action on becoming aware of the contradiction is not determined by this
1677 specification. In particular, if the Superior is a Sub-coordinator or Sub-composer, it is not
1678 required by this specification to report the contradiction to its own Superior (which may, for
1679 example, be controlled by a different organisation). The Superior may report the problem to
1680 management systems or record it for manual repair. However, BTP does provide mechanisms
1681 to report the contradiction to the next higher Superior (if there is one) or to the Terminator
1682 application element.

1683

1684 A contradiction occurring in an Inferior will usually mean the immediate Superior has a
1685 “mixed” condition – some of the application work it was responsible for has confirmed, some
1686 has cancelled (and contrary to any cohesion confirm-set selection). If the Superior is a Sub-
1687 coordinator or Sub-composer, it can report the mixed condition to its own Superior with the
1688 HAZARD message. If the Superior is the top-most in the tree, it can report the problem with
1689 the INFERIOR_STATUSES message, which will detail the state of all the Inferiors.

1690

1691 If a Sub-coordinator or Sub-composer having sent (or attempted to send) the outcome
1692 message to its Inferiors, is temporarily unable to get a response (CONFIRMED or
1693 CANCELLED), it may either wait until a response does come back or choose to reply to its
1694 own Superior with a HAZARD message indicating that a contradiction is “possible”. If it
1695 does choose to send HAZARD, it is required to persist a record of this until it receives a

1696 CONTRADICTION message from the Superior, or a message from the Inferior indicating
1697 there was no contradiction in fact.

1698
1699 HAZARD is also used to indicate that it has become impossible to cleanly and consistently
1700 achieve either a confirmed or a cancelled state for the application work. In this case, there is
1701 can be no guarantee that the problem will be reliably reported – especially because it may be
1702 the inability to persist information that is the cause of the problem.

1703 Relation of BTP to application and carrier protocols

1704
1705 BTP messages are communicated between actors in two distinguishable circumstances:

- 1706 a) in establishing and progressing the outcome and control relationships between BTP
1707 actors, and between application elements and BTP actors – Initiator:Factory,
1708 Terminator:Decider, Superior:Inferior etc.
1709 b) in association with application messages that are communicated between application
1710 elements.

1711
1712 In the first case, interoperable communication requires a specification of how the abstract
1713 BTP messages are represented and encoded, and how they are transmitted. This specification
1714 is a **carrier protocol binding** (or just “binding”, if the context is clear). BTP allows bindings
1715 to a multiplicity of carrier protocols. The only requirement that BTP makes is that the
1716 transmission of a message either delivers an uncorrupted message or fails. BTP does not
1717 require that the carrier report failure to deliver a message, to either side, nor that messages are
1718 delivered in the order they are sent (though implementations can take advantage of
1719 information from a richer carrier, which can improve performance in various ways). BTP
1720 messages communicated in this way have semantics that are defined in this specification – a
1721 PREPARE message (for example), refers back to the ENROL via the “inferior-identifier”
1722 parameter and is an instruction to the Inferior to become and report that it is prepared.

1723
1724 In the second case, the full semantics cannot be defined in this specification. Interoperation
1725 with BTP requires that the parties have a common understanding of what is being confirmed
1726 or cancelled, but this mutual understanding is defined by the contract of the application, not
1727 by BTP. (The contract may be explicit or implicit, declared by one side as take-it-or-leave-it,
1728 or may be negotiated in some way.) Part of this contract will include how the combination of
1729 the application protocol (i.e. the application messages and their sequencing) and BTP operate
1730 such that the two sides are agreed as to which application operations are part of which
1731 business transaction. This will often be achieved by sending application messages and BTP
1732 messages in “association” in some way – thus an application message sent in association with
1733 a CONTEXT can be specified (by the application contract) to mean that if work is done as
1734 result of the receipt of the message, one or more Inferiors should be enrolled to apply the
1735 confirm/cancel decision to that work. Similarly, an application message may be sent
1736 associated with an ENROL with the contractual understanding that the message refers to
1737 some application work that has been made the responsibility of the Inferior being enrolled.

1738
1739 The concrete representation of this “association” is also a matter for the application protocol
1740 specification. There are several ways this can be done, including:

- 1741 • the BTP message is contained within the application message, or both are contained
1742 within a larger construct;
1743 • the application message contains a field that is the superior-identifier or inferior-
1744 identifier that is also present on the CONTEXT or the ENROL
1745 • the BTP message contains a qualifier that references (a field of) the application
1746 message in some way (e.g. if the application message is an invoice, the qualifier
1747 might contain the invoice number)
1748 • the encoding of the BTP and application messages reference each other (e.g. using
1749 XML id and refid attributes)

1750

1751 In all cases, the application specification⁵ will need to define the mechanism so that both
1752 parties have common understanding. Many applications will use the same mechanism and
1753 their specifications can therefore take advantage of standard patterns, and their
1754 implementations of standard tools.

1755

1756 The association of an application message with a BTP message is analogous to the concept of
1757 “related” BTP messages. “Related” BTP messages are sent as a group, with a declared and
1758 defined semantic for the group. Associated application and BTP messages can be considered
1759 as “related”, with the proviso that the semantic is defined by the application, not by BTP.

1760

1761 There is no necessary relationship between how the application messages and any associated
1762 BTP messages are transmitted by carrier protocols, and the carrier binding for the BTP
1763 messages. BTP messages are invariably sent to a BTP actor whose address has been passed to
1764 the sender by some means – thus a CONTEXT contains the address of the Superior to which
1765 ENROLs will be sent, and the ENROL contains the address of the Inferior. Similarly,
1766 BEGUN contains the address (as Decider) of the new Composer or Coordinator. These
1767 addresses are all sets of addresses (possibly of cardinality one), and each individual address
1768 identifies which binding is to be used. Thus, for example, when a CONTEXT is sent
1769 associated with an application message, the ENROL will travel on a carrier binding identified
1770 by the particular address from the CONTEXT that the Enroller chooses to use – which may
1771 have no relationship to how the application message arrived.

1772

1773 Despite this, it will be common that the application binding and the BTP binding will use the
1774 same carrier. This is the case in the bindings specified in this edition of the specification,
1775 which define a binding of BTP to SOAP 1.1 over HTTP. Included in this SOAP/HTTP
1776 binding specification, are rules that allow an application to associate (relate) a single
1777 CONTEXT or a single ENROL (carried in the SOAP header) with the application message(s)
1778 carried in the SOAP body.

1779

1780

Other elements

1781

1782

Identifiers

1783

⁵ The “application specification”, or “application protocol specification” may be very informal or may be a standardised agreement.

1784 An Identifier is a globally unambiguous identification of the state corresponding to one of
1785 Decider, Superior or Inferior. Where a single entity has more than one of these roles (at the
1786 same node in the same transaction, as with a Sub-coordinator that is both Superior and
1787 Inferior), the Identifiers may be the same or different, at implementation option - they are
1788 distinguished by which messages the Identifier is used on. (A Superior has only one Superior-
1789 identifier, although it may be in multiple Superior:Inferior relationships, each with a separate
1790 state in terms of the state table).

1791
1792 The state identified by an Identifier can be accessed by BTP messages sent to any of the
1793 addresses supplied with the Identifier in the appropriate message (CONTEXT, BEGUN,
1794 ENROL), or as updated by REDIRECT. An Identifier itself has no location implications.
1795 (Identifiers are specified, in the XML representation, as syntactically URIs - by their use as
1796 names of BTP entities, they are URNs. If an Identifier happens to specify a network location
1797 (i.e. it is a URL), it is treated as an opaque value by BTP)

1798
1799 Identifiers are specified as being globally unambiguous - the same Identifier only ever
1800 identifies one Decider, Superior or Inferior over all systems and all time. In practice, an
1801 Identifier could be re-used if there is no possibility of the colliding values being confused.
1802 However implementations are recommended to use truly unambiguous Identifiers (that is to
1803 use them as URNs).

1804
1805 **Addresses**
1806 In most cases, BTP actors that need to communicate are informed of each others addresses
1807 from received BTP messages. When an Inferior is to be enrolled, a CONTEXT message
1808 which contains the address of the Superior will have been received or otherwise passed to the
1809 Enroller and the Inferior. The ENROL message received by the Superior contains the address
1810 of the Inferior. The BEGUN returned from a Factory to the Initiator contains the address of
1811 the Decider, and this can be passed to the Terminator or any Status Requestor.

1812
1813 The addresses carried in these messages (which are effectively “call-back” addresses, to be
1814 used as the destination of future messages) are sets of tripartite addresses. Each contains an
1815 identifier (binding name) for the binding to an underlying transport, or carrier protocol, a
1816 “binding address”, in a format specific to the carrier which is the information necessary to
1817 connect using that carrier, and an optional additional information field. This additional
1818 information is opaque to all but the future destination (which also created this address for
1819 itself) and is used however the implementation there wishes (e.g. it can be used to distinguish
1820 a particular program object, or to relay on, perhaps over a different protocol). The multiple
1821 members of the set allow support of multiple carrier bindings (including both different
1822 versions of standard bindings and proprietary bindings) and for relocation of the BTP actor.

1823
1824 When a message is actually to be sent, the sender, possessing the set of addresses for the
1825 destination, chooses one - restricting its choice to bindings that it supports obviously, but not
1826 otherwise constrained by the specification. The binding address will be used by the senders
1827 carrier implementation (depending on the protocol, the address may or may not be transmitted
1828 – with http, for example, it is). The additional information, if present, will be included in the
1829 BTP message. The chosen address is considered the “target-address” when considering the

1830 [abstract message, but only the additional information will normally appear within the](#)
1831 [encoded BTP-message \(the encoding used is part of the binding specification, which could](#)
1832 [require that all of the address is \(redundantly\) transmitted, if the specifier so chose\).](#)

1833
1834 [Where a BTP message invokes a reply – as with the Initiator:Factory, Terminator:Decider](#)
1835 [and Status Requestor:various roles – the receiver \(Factory, Decider, etc\) of the message will](#)
1836 [not know *a priori* the address of the sender. Accordingly, in these cases the abstract messages](#)
1837 [are specified as containing a single “reply-address”. Depending on the binding, and the](#)
1838 [particular use of the binding, the “reply-address” may be directly represented in the encoding](#)
1839 [of the BTP message, or may be implicit in the carrier protocol. Similar considerations apply](#)
1840 [in the Superior:Inferior relationship, where although the addresses are normally known by the](#)
1841 [other side, there are cases when a message is received, and must be responded to, but the peer](#)
1842 [is unknown. Accordingly, the Superior:Inferior messages contain \(in abstract\) a single](#)
1843 [“senders-address”. As with the the “reply-address”es, it may be implicit in the carrier](#)
1844 [protocol.](#)

1845
1846 [The CONTEXT message does not contain a “target-address”, even as an abstract message, as](#)
1847 [it is never transmitted between BTP actors on its own – it is always either related to a BTP](#)
1848 [BEGIN or BEGUN message, or is passed between application elements with some](#)
1849 [\(application-detailed\) association with application messages.](#)

1850
1851 **Qualifiers**
1852 [Qualifiers are elements of the BTP messages used to exchange additional information](#)
1853 [between the actors. Qualifiers can be specified in the BTP specification \(“standard](#)
1854 [qualifiers”\), by industry groups, by BTP implmentors or for the purposes of particular](#)
1855 [applications. Of the standard qualifiers in this version of the specification some are](#)
1856 [constraints on the BTP contract, such as time limits, and some are further identifiers used to](#)
1857 [distinguish specific parties in the BTP interchange. Non-standard qualifiers could extend the](#)
1858 [protocol or carry application-specific information.](#)

1859

1860 **~~Overview of the Business Transaction Protocol~~**

1861
1862 ~~[A Business Transaction is a consistent change in the state of a business relationship between](#)~~
1863 ~~[two or more parties. BTP provides means to allow the consistent and coordinated changes in](#)~~
1864 ~~[the relationship as viewed from each party.](#)~~

1865
1866 ~~[BTP assumes that for a given business transaction state changes occur, or are desired, in some](#)~~
1867 ~~[set of parties, and that these changes are related in some business defined manner.](#)~~

1868
1869 ~~[Typically business defined messages \(“application messages”\) are exchanged between the](#)~~
1870 ~~[parties to the transaction, which result in the performance of some set of operations. These](#)~~
1871 ~~[operations create provisional or tentative state changes \(the transaction’s effect\). The](#)~~
1872 ~~[provisional changes of each party must either be confirmed \(given final effect\), or must be](#)~~
1873 ~~[cancelled \(counter effected\). Those parties which are confirmed create an atomic unit, within](#)~~
1874 ~~[which the business transaction should have a consistent final effect.](#)~~

1875

1876 The meaning of “effect”, “final effect” and “counter effect” is specific to each business
1877 transaction and to each party’s role within it. A party may log intended changes (as its effect)
1878 and only process them as visible state changes on confirmation (its final effect). Or it may
1879 make visible state changes and store the information needed to cancel (its effect), and then
1880 simply delete the information needed for cancellation (its final effect). A counter effect may
1881 be a precise inversion or removal of provisional changes, or it may be the processing of
1882 operations that in some way compensate for, make good, alleviate or supplement their effect.
1883
1884 To ensure that confirmation or cancellation of the provisional effect within different parties
1885 can be consistently performed, it is necessary that each party should
1886
1887 —determine whether it is able both to cancel (counter effect) and to confirm (give final
1888 effect to) its effect
1889
1890 —report its ability or inability to cancel or confirm (its preparedness) to a central
1891 coordinating entity
1892
1893 After receiving these reports, the coordinating entity is responsible for determining which of
1894 the parties should be instructed to confirm and which should be instructed to cancel.
1895
1896 Such a two phase exchange (ask, instruct) mediated by a central coordinator is required to
1897 achieve a consistent outcome for a set of operations. BTP defines the means for software
1898 agents executing on network nodes to interoperate using a two phase coordination protocol,
1899 leading either to the abandonment of the entire attempted transaction, or to the selection of an
1900 internally consistent set of confirmed operations.
1901
1902 BTP centres on the bilateral relationship between the computer systems of the coordinating
1903 entity and those of one of the parties in the overall business transaction. In that relationship a
1904 software agent within the coordinating entity’s systems plays the BTP role of Superior for a
1905 given transaction and one or more software agents within the systems of the party play the
1906 BTP role of Inferior. Each Inferior has one Superior, therefore, while a single Superior may
1907 have multiple Inferiors within each party to the transaction, and may be related to Inferiors
1908 within multiple parties. Each Superior:Inferior pair exchanges protocol defined messages.
1909
1910 An Inferior is associated with some set of operation invocations that creates effect
1911 (provisional or tentative changes) within the party, for a given business transaction. The
1912 Inferior is responsible for reporting to its related Superior whether its associated operations’
1913 effect can be confirmed/cancelled. A Superior is responsible for gathering the reports of all of
1914 its Inferiors, in order to ascertain which should be cancelled or confirmed. For example, if a
1915 Superior is acting as an atomic Coordinator it will treat any Inferior which cannot prepare to
1916 cancel/confirm as having veto power over the whole business transaction, causing the
1917 Superior to instruct all its Inferiors to cancel. A Superior may, under the dictates of a
1918 controlling application, increase or reduce the set of Inferiors to which a common confirm or
1919 cancel outcome may be delivered. Thus, the set of prepared Inferiors may be larger than the
1920 set of confirmed Inferiors.
1921

1922 An Inferior:Superior relationship is typically established in relation to one or more
1923 application messages sent from one part of the application (linked to the Superior) to some
1924 other part of the application to request the performance of operations that are to be subject to
1925 the confirm or cancel decision of the Superior. If an application is divided between a client
1926 and a service, which use RPCs to communicate application requests and responses, then the
1927 client would typically be associated with the Superior and the service would typically host the
1928 Inferior(s). (BTP does not mandate such an application topology nor does it require the use of
1929 RPC or any other application communication paradigm.)

1930
1931 BTP defines a CONTEXT message that can be sent “in relation to” such application
1932 messages. On receipt of a CONTEXT, one or more Inferiors may be created and “enrolled”
1933 with the Superior, establishing the Superior:Inferior relationships. The particular mechanisms
1934 by which a CONTEXT is “related” to application messages is an issue for the application
1935 protocol and its binding to carrier mechanisms. BTP does not require that the enrolment is
1936 requested by any particular entity—in a particular implementation this may be done by the
1937 Inferior itself, by parts of the application or by other entities involved in the transmission of
1938 the CONTEXT and the application messages. BTP defines a CONTEXT_REPLY message
1939 that can be sent on the return path of the CONTEXT to indicate whether the enrolment was
1940 successful. Without CONTEXT_REPLY it would be possible for a Superior to have an
1941 incorrect view of which Inferiors it was supposed to involve in its confirm decision.

1942
1943 It should be noted that this BTP specification recognises that:

- 1944 —an Inferior may itself be a Superior to other BTP Inferiors; this occurs when some of
1945 the operations associated with the Inferior involve other application elements whose
1946 operations are to be subject to the confirm/cancel instruction sent to the Inferior. The
1947 specification treats any lower Inferiors as part of the associated operations;
- 1948 —the requirement on an Inferior to be able to confirm or cancel does not include any
1949 specific mechanism to determine the isolation of the effects of operations; the
1950 requirement is only that the Inferior is able to confirm or cancel the operations, as
1951 their effects are known to the Superior and the application directly in contact with the
1952 Superior. Thus the confirm or cancel requirement may be achieved by performing all
1953 the operations and remembering a compensating counter operation (that will be
1954 triggered by a cancel order); or by remembering the operations (having checked they
1955 are valid) and performing them only if a confirm order is received; or by forbidding
1956 any other access to data changed by the operations and releasing them in their
1957 unchanged state (if cancelled) or their changed state (if confirmed); or by various
1958 combinations of these. In addition, a cancellation may not return data to their original
1959 state, but only to a state accepted by the application as appropriate to a cancelled
1960 operation.

1961
1962
1963
1964
1965
1966
1967

1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999

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.

2003
2004
2005
2006

Relationships

There are two primary relationships in BTP.

2007 □ Between an application element that determines that a business transaction should be
2008 completed (the role of Terminator) and the BTP actor at the top of the transaction tree
2009 (the role of Decider);

2010

2011 □ Between BTP actors within the tree, where one (the Superior) will inform the other
2012 (the Inferior) what the outcome decision is.

2013

2014 These primary relationships are involved in arriving at a decision on the outcome of a
2015 business transaction, and propagating that decision to all parties to the transaction. Taking the
2016 path that is followed when a business transaction is confirmed:

2017 1. The Terminator determines that the business transaction should confirm, if it can; or
2018 (for a Cohesion), which parts should confirm

2019 2. The Terminator asks the Decider to apply the desired outcome to the tree, if it can
2020 guarantee the consistency of the confirm decision

2021 3. The Decider, which is Superior to one or more Inferiors, asks its Inferiors if they can
2022 agree to a confirm decision (for a Cohesion, this may not be all the Inferiors)

2023 4. If any of those Inferiors are also Superiors, they ask their Inferiors and so on down
2024 the tree

2025 5. Inferiors that are not Superiors report if they can agree to a confirm to their Superior

2026 6. Inferiors that are also Superiors report their agreement only if they received such
2027 agreement from their Inferiors, and can agree themselves

2028 7. Eventually agreement (or not) is reported to the Decider. If all have agreed, the
2029 Decider makes and persists the confirm decision (hence the term “Decider” – it
2030 decides, everything else just asked); if any have disagreed, or if the confirm decision
2031 cannot be persisted, a cancel decision is made

2032 8. The Decider, as Superior tells its Inferiors of the outcome

2033 9. Inferiors that are also Superiors tell their Inferiors, recursively down the tree

2034 10. The Decider replies to the Terminator’s request to confirm, reporting the outcome
2035 decision

2036

2037 There are other relationships that are secondary to Terminator:Decider, Superior:Inferior,
2038 mostly involved in the establishment of the primary relationships. The various particular
2039 relationships can be grouped as the “control” relationships – primarily Terminator:Decider,
2040 but also Initiator:Factory; and the “outcome” relationships – primarily Superior:Inferior, but
2041 also Enroller:Superior.

2042

2043 The two groups of relationships are linked in that a Decider is a Superior to one or more
2044 Inferiors. There are also similarities in the semantics of some of the exchanges (messages)
2045 within the relationships. However they differ in that

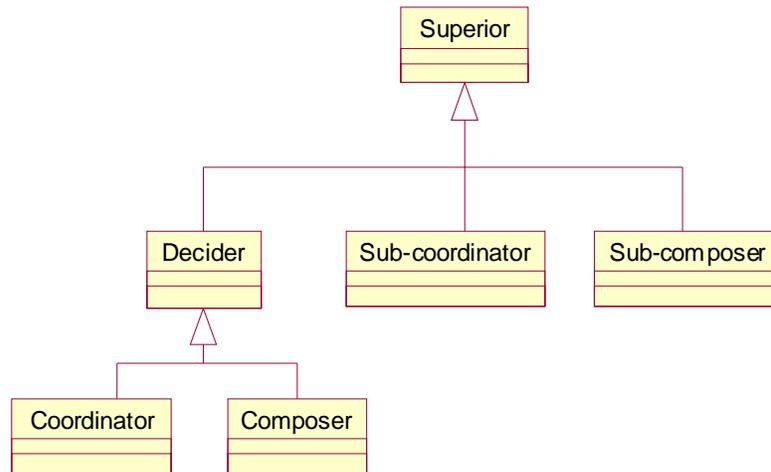
2046

2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064

1. All exchanges between Terminator and Decider are initiated by the Terminator (it is essentially a request/response relationship); either of Superior or Inferior may initiate messages to the other
2. The Superior:Inferior relationship is recoverable – depending on the progress of the relationship, the two sides will re-establish their shared state after failure; the Terminator:Decider relationship is not recoverable
3. The nature of the Superior:Inferior relationship requires that the two parties know of each other's addresses from when the relationship is established; the Decider does not need to know the address of the Terminator (provided it has some way of returning the response to a received message).

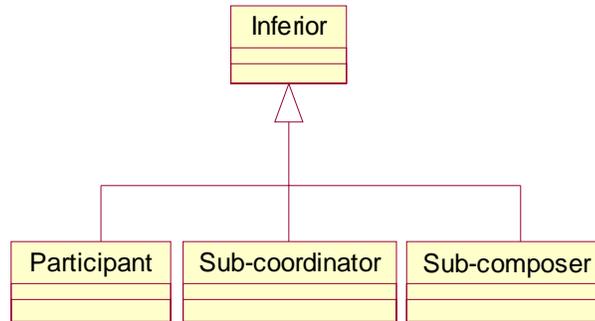
Roles

[Figure 15](#)~~Figure 1~~~~Figure 1~~~~Figure 1~~~~Figure 1~~ and [Figure 16](#)~~Figure 2~~~~Figure 2~~~~Figure 2~~~~Figure 2~~ show the BTP roles that are specialisations of the central Superior and Inferior roles.



2065
2066
2067

[Figure 15](#)~~1~~ Superior and derived roles



2068

2069

Figure 16.2 Inferior and derived roles

2070

2071

2072

2073

2074

2075

2076

2077

2078

2079

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

For each role, a table shows which messages are received and sent. Where the messages appear on the same line, the second is a reply to the first. (Consequently the columns are sometimes sent first, received second, sometimes vice versa.)

2080

Roles involved in the outcome relationships

2081

2082

Superior

2083

2084

2085

2086

2087

2088

2089

2090

2091

2092

2093

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

2094

2095

2096

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

2097

2098

2099

2100

2101

2102

2103

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

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

2104

<u>Superior receives</u>	<u>Superior sends</u>
<u>ENROL</u>	<u>ENROLLED</u>
	<u>PREPARE</u>
	<u>CONFIRM</u>
	<u>CANCEL</u>
	<u>RESIGNED</u>
	<u>CONFIRM_ONE_PHASE</u>
	<u>CONTRADICTION</u>
	<u>SUPERIOR_STATE</u>
<u>PREPARED</u>	
<u>CONFIRMED</u>	
<u>CANCELLED</u>	
<u>HAZARD</u>	
<u>RESIGN</u>	
<u>INFERIOR_STATE</u>	
<u>REQUEST STATUS</u>	<u>STATUS</u>
<u>REQUEST INFERIORS STATUS</u>	<u>INFERIOR_STATUSES</u>

2105

2106

2107

2108

2109

2110

2111

2112

2113

2114

2115

2116

2117

2118

2119

2120

2121

2122

2123

2124

2125

2126

2127

2128

2129

2130

2131

2132

Receipt of ENROL establishes a new Superior:Inferior relationship (unless the ENROL is a duplicate). ENROLLED is sent only if a reply is asked for on the ENROL. ~~A Superior receives~~

~~ENROL~~

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

~~A Superior sends~~

~~ENROLLED~~

~~in reply to ENROL, if the appropriate parameter on the ENROL asked for the reply.~~

~~A Superior sends~~

~~PREPARE~~

~~CONFIRM~~

~~CANCEL~~

~~RESIGNED~~

~~CONFIRM_ONE_PHASE~~

~~SUPERIOR_STATE~~

~~to an enrolled Inferior.~~

~~A Superior receives~~

2133
 2134 **PREPARED**
 2135 **CANCELLED**
 2136 **CONFIRMED**
 2137 **HAZARD**
 2138 **RESIGN**
 2139 **INFERIOR_STATE**

2141 ~~from an enrolled Inferior.~~

2142
 2143 **Inferior**

2144
 2145 Responsible for applying the Outcome to some set of associated operations – the application
 2146 determines which operations are the responsibility of a particular Inferior.

2147
 2148 An Inferior is **Enrolled** with a single Superior (hereafter referred to as “its Superior”),
 2149 establishing a Superior:Inferior relationship. If the Inferior is able to ensure that either a
 2150 confirm or cancel decision can be applied to the associated operations, and can persist
 2151 information to retain that condition, it sends a PREPARED message to the Superior. When
 2152 the Outcome is received from the Superior, the Inferior applies it, deletes the persistent
 2153 information, and replies with CANCELLED or CONFIRMED as appropriate.

2154
 2155 If an Inferior is unable to come to a prepared state, it cancels the associated operations and
 2156 informs the Superior with a CANCELLED message. If it is unable to either come to a
 2157 prepared state, or to cancel the associated operations, it informs the Superior with a
 2158 HAZARD message.

2159
 2160 An Inferior that has become prepared may, exceptionally, make an autonomous decision to be
 2161 applied to the associated operations, without waiting for the Outcome from the Superior. It is
 2162 required to persist this autonomous decision and report it to the Superior with CONFIRMED
 2163 or CANCELLED as appropriate. If, when CONFIRM or CANCEL is received, the
 2164 autonomous decision and the decision received from the Superior are contradictory, the
 2165 Inferior must retain the record of the autonomous decision until receiving a
 2166 CONTRADICTION message.

<u>Inferior receives</u>	<u>Inferior sends</u>
<u>PREPARE</u>	
<u>CONFIRM</u>	
<u>CANCEL</u>	
<u>RESIGNED</u>	
<u>CONFIRM ONE PHASE</u>	
<u>CONTRADICTION</u>	
<u>SUPERIOR_STATE</u>	
	<u>PREPARED</u>
	<u>CONFIRMED</u>
	<u>CANCELLED</u>

	<u>HAZARD</u>
	<u>RESIGN</u>
	<u>INFERIOR_STATE</u>
<u>REQUEST_STATUS</u>	<u>STATUS</u>
<u>REQUEST_INFERIORS_STATUS</u>	<u>INFERIOR_STATUSES</u>

2168

2169

An Inferior receives

2170

PREPARE

2171

CONFIRM

2172

CANCEL

2173

RESIGNED

2174

CONFIRM_ONE_PHASE

2175

SUPERIOR_STATE

2176

2177

2178

from its Superior.

2179

2180

An Inferior sends

2181

PREPARED

2182

CANCELLED

2183

CONFIRMED

2184

HAZARD

2185

RESIGN

2186

INFERIOR_STATE

2187

2188

2189

to its Superior.

2190

2191

Enroller

2192

2193

2194

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

2195

2196

2197

2198

2199

<u>Enroller sends</u>	<u>Enroller receives</u>
<u>ENROL</u>	<u>ENROLLER</u>

2200

An Enroller sends

2201

ENROL

2202

2203

to a Superior.

2204

2205

2206

An Enroller receives

2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233

2234

2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246

2247

ENROLLED [is received only if the](#)

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

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

Participant

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

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

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

Note – Possible approaches are:

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

2248 [Since a Participant is an Inferior, it sends and receives the messages for an Inferior.](#)

2249

2250 **Sub-coordinator**

2251

2252 An Inferior which is also an Atomic Superior.

2253

2254 A sub-coordinator is the Inferior in one Superior:Inferior relationship and the Superior in one
2255 or more Superior:Inferior relationships.

2256

2257 From the perspective of its Superior (the one the sub-coordinator is Inferior to), there is no
2258 difference between a sub-coordinator and any other Inferior. From this perspective, the
2259 “associated operations” of the sub-coordinator as an Inferior include the relationships with its
2260 Inferiors.

2261

2262 A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and
2263 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is
2264 propagated to all Inferiors.

2265

2266 [Since a Sub-coordinator is both an Inferior and a Superior, it sends and receives the messages
2267 for both.](#)

2268

2269 **Sub-composer**

2270

2271 An Inferior which is also a Cohesive Superior.

2272

2273 Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from
2274 the perspective of its Superior.

2275

2276 A sub-composer is similar to a sub-coordinator, except that the constraints linking the
2277 different Inferiors concern only those Inferiors in the confirm-set. How the confirm-set is
2278 controlled, and when, is not defined in this specification.

2279

2280 If the sub-composer is instructed to cancel, by receiving a CANCEL message from its
2281 Superior, the cancellation is propagated to all its Inferiors.

2282

2283 [Since a Sub-composer is both an Inferior and a Superior, it sends and receives the messages
2284 for both.](#)

2285

2286 **Roles involved in the control relationships**

2287

2288 **Decider**

2289

2290 A Superior that is not also the Inferior on a Superior:Inferior relationship. It is the top-node in
2291 the transaction tree and receives requests from a Terminator as to the desired outcome for the
2292 business transaction. If the Terminator asks the Decider to confirm the business transaction, it
2293 is the responsibility of the Decider to finally take the confirm decision. The taking of the

2294 decision is synonymous with the persisting of information identifying the Inferiors that are to
 2295 be confirmed. An Inferior cannot be confirmed unless PREPARED has been received from it.
 2296
 2297 A Decider is instructed to cancel by receiving CANCEL_TRANSACTION.
 2298
 2299 A Decider that is an Atomic Superior (all Inferiors will have the same outcome) is a
 2300 Coordinator. A Decider that is a Cohesive Superior (some Inferiors may cancel, some
 2301 confirm) is a Cohesion.
 2302

<u>Decider receives</u>	<u>Decider sends</u>
<u>CONFIRM_TRANSACTION</u>	<u>TRANSACTION_CONFIRMED</u> <u>TRANSACTION_CANCELLED</u> <u>INFERIOR_STATUSES</u>
<u>CANCEL_TRANSACTION</u>	<u>TRANSACTION_CANCELLED</u> <u>INFERIOR_STATUSES</u>
<u>REQUEST_INFERIOR_STATUSES</u>	<u>INFERIOR_STATUSES</u>

2303
 2304 [A Decider is also a Superior and thus sends and receives the messages for a Superior.](#)

2305 ~~All Deciders receive~~
 2306 ~~—CONFIRM_TRANSACTION~~
 2307 ~~—CANCEL_TRANSACTION~~
 2308 ~~—REQUEST_INFERIOR_STATUSES~~

2309
 2310 ~~All Deciders send~~
 2311 ~~—TRANSACTION_CONFIRMED~~
 2312 ~~—TRANSACTION_CANCELLED~~
 2313 ~~—INFERIOR_STATUSES~~

2314
 2315
 2316 **Coordinator**

2317
 2318 A Decider that is an Atomic Superior. The same outcome decision will be applied to all
 2319 Inferiors (excluding any from which RESIGN is received).
 2320

2321 PREPARED must be received from all remaining Inferiors for a confirm decision to be taken.
 2322

2323 A Coordinator must make a cancel decision if
 2324 it is instructed to cancel by the Terminator
 2325 if CANCELLED is received from any Inferior
 2326 if it is unable to persist a confirm decision
 2327

2328 [Since a Coordinator is a Decider, it receives the messages appropriate for a Decider and a](#)
 2329 [Superior.](#)

2330
 2331 **Composer**

2332

2333 A Decider that is a Cohesive Superior. If the Terminator requests confirmation of the
 2334 Cohesion, that request will determine the confirm-set of the Cohesion.
 2335
 2336 PREPARED must be received from all Inferiors in the confirm-set (excluding any from
 2337 which RESIGN is received) for a confirm decision to be taken.
 2338
 2339 A Composer must make a cancel decision (applying to all Inferiors) if
 2340 it is instructed to cancel by the Terminator
 2341 if CANCELLED is received from any Inferior in the confirm-set
 2342 if it is unable to persist a confirm decision
 2343
 2344 A Composer may be asked to prepare some or all of its Inferiors by receiving
 2345 PREPARE_INFERIORS. It issues PREPARE to any of those Inferiors from which none of
 2346 PREPARED, CANCELLED or RESIGN have been received, and replies to the
 2347 PREPARE_INFERIORS with INFERIOR_STATUSES.
 2348
 2349 A Composer may be asked to cancel some of its Inferiors, but not itself, by receiving
 2350 CANCEL_INFERIORS.
 2351

<u>Composer receives</u>	<u>Composer sends</u>
<u>PREPARE_INFERIORS</u>	<u>INFERIOR_STATUSES</u>
<u>CANCEL_INFERIORS</u>	<u>INFERIOR_STATUSES</u>

2352
 2353
 2354
 2355
 2356
 2357
 2358
 2359
 2360
 2361
 2362
 2363
 2364
 2365
 2366
 2367
 2368
 2369
 2370
 2371
 2372
 2373
 2374
 2375

Terminator

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

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

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

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

A Terminator may send CANCEL_TRANSACTION to instruct the Decider to cancel the whole business transaction., The Decider replies with CANCEL_COMPLETE if all Inferiors cancel successfully, and with INFERIOR_STATUSES in the case of problems.. If the

2376 Decider is a Cohesion Composer, the Terminator may send CANCEL_INFERIORS to cancel
 2377 some of the Inferiors; the Decider always replies with INFERIOR_STATUSES.
 2378
 2379 A Terminator may check the status of the Inferiors of the Decider by sending
 2380 REQUEST_INFERIOR_STATUSES. The Decider replies with INFERIOR_STATUSES.
 2381

<u>Terminator sends</u>	<u>Terminator receives</u>
<u>CONFIRM_TRANSACTION</u>	<u>TRANSACTION_CONFIRMED</u> <u>TRANSACTION_CANCELLED</u> <u>INFERIOR_STATUSES</u>
<u>CANCEL_TRANSACTION</u>	<u>TRANSACTION_CANCELLED</u> <u>INFERIOR_STATUSES</u>
<u>PREPARE_INFERIORS</u>	<u>INFERIOR_STATUSES</u>
<u>CANCEL_INFERIORS</u>	<u>INFERIOR_STATUSES</u>
<u>REQUEST_INFERIOR_STATUSES</u>	<u>INFERIOR_STATUSES</u>

2382 ~~A Terminator sends~~
 2383 ~~—CONFIRM_TRANSACTION~~
 2384 ~~—CANCEL_TRANSACTION~~
 2385 ~~—CANCEL_INFERIORS~~
 2386 ~~—PREPARE_INFERIORS~~
 2387 ~~—REQUEST_INFERIOR_STATUSES~~
 2388

2389 ~~A Terminator receives~~
 2390 ~~—TRANSACTION_CONFIRMED~~
 2391 ~~—TRANSACTION_CANCELLED~~ ~~—INFERIOR_STATUSES~~
 2392

Initiator

2393
 2394
 2395 Requests a **Factory** to create a Superior – this will either be a Decider (representing a new
 2396 top-level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an
 2397 existing business transaction.
 2398

<u>Initiator sends</u>	<u>Initiator receives</u>
<u>BEGIN</u>	<u>BEGUN & CONTEXT</u>
<u>BEGIN & CONTEXT</u>	<u>BEGUN & CONTEXT</u>

2399
 2400 The received CONTEXT is that for the new Superior. ~~An Initiator sends~~
 2401
 2402 ~~—BEGIN~~
 2403 ~~—BEGIN & CONTEXT~~
 2404
 2405 ~~to a Factory, and receives in reply~~
 2406
 2407 ~~—BEGUN & CONTEXT~~
 2408

2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419

Factory

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

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

<u>Factory receives</u>	<u>Factory sends</u>
<u>BEGIN</u>	<u>BEGUN & CONTEXT</u>
<u>BEGIN & CONTEXT</u>	<u>BEGUN & CONTEXT</u>

2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438

~~A Factory receives~~

~~—BEGIN~~

~~—BEGIN & CONTEXT~~

~~and replies with~~

~~BEGUN & CONTEXT~~

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

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

Other roles

2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451

Redirector

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

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

If a Superior moves from the superior-address in its CONTEXT, or an Inferior moves from the inferior-address in the ENROL message, the implementation **must** ensure that a

2452 Redirector catches any inbound messages using the old address and replies with a
 2453 REDIRECT message giving the new address. (Note that the inbound message may itself be a
 2454 REDIRECT message, in which case the Redirector shall use the new address in the received
 2455 message as the target for the REDIRECT that it sends.)

2456
 2457
 2458
 2459
 2460

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

<u>Redirector receives</u>	<u>Redirector sends</u>
<u>Any message for Superior or Inferior</u>	<u>REDIRECT</u>

2461
 2462
 2463
 2464
 2465
 2466
 2467
 2468
 2469

Status Requestor

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

<u>Status Requestor sends</u>	<u>Status Requestor receives</u>
<u>REQUEST_STATUS</u>	<u>STATUS</u>
<u>REQUEST_INFERIOR_STATUS</u>	<u>INFERIOR_STATUSES</u>

2470
 2471
 2472
 2473
 2474
 2475
 2476
 2477
 2478
 2479
 2480
 2481
 2482
 2483
 2484
 2485

~~A-Status-Requestor-sends~~

~~REQUEST_STATUS
 REQUEST_INFERIOR_STATUSES~~

~~and-receives~~

~~—STATUS
 —INFERIOR_STATUSES~~

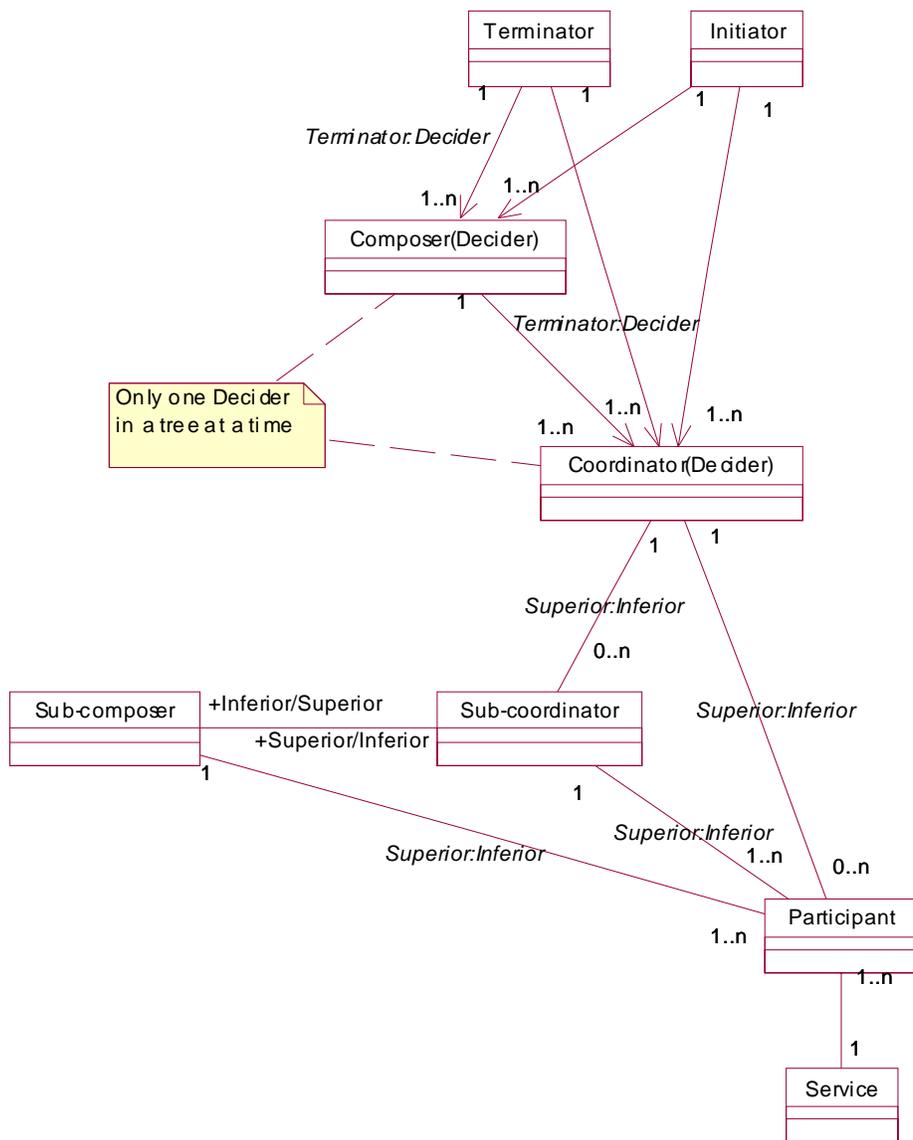
~~in response.~~

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

Summary of relationships

2486
 2487
 2488
 2489
 2490

Figure 17~~Figure 3~~~~Figure 3~~~~Figure 3~~~~Figure 3~~ summarises the relationships between the BTP roles. BTP can be implemented using proprietary equivalents of the Terminator and Decider roles.



2491
 2492
 2493

Figure 173 Summary of relationships between roles

2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538

Abstract Messages and Associated Contracts

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

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

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

The abstract messages include **Delivery parameters** that are concerned with the transmission and delivery of the messages as well as **Payload parameters** directly concened with the progression of the BTP relationships. When bound to a particular carrier protocol and for particular implementation configurations, parts or all of the Delivery parameters may be implicit in the carrier protocol and will not appear in the "on-the-wire" representation of the BTP messages as such. Delivery parameters are defined as being only those parameters that are concerned with the transmission of this message, or of an immediate reply (thus address parameters to be used in repeated later messages and the identifiers of both sender and receiver are Payload parameters). In the tables in this section, Delivery parameters are shown in shaded cells.

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.

2539 In all cases, the exact value will ~~invariably~~ have been originally determined by the
2540 implementation that is the target or ~~desired future~~intended target.

2541
2542 The detailed format of the address will depend on the particular carrier protocol, but at this
2543 abstract level is considered to have three parts. The first part, the “binding name”, identifies
2544 the binding to a particular carrier protocol – some bindings are specified in this document,
2545 others can be specified elsewhere. The second part of the address, the “binding address”, is
2546 meaningful to the carrier protocol itself, which will use it for the communication (i.e. it will
2547 permit a message to be delivered to a receiver). The third part, “additional information”, is
2548 not used or understood by the carrier protocol. The “additional information” may be a
2549 structured value.

2550
2551 When a message is actually transmitted, the “binding name” of the target address will identify
2552 which carrier protocol is in use and the “binding address” will identify the destination, as
2553 known to the carrier protocol. The entire binding address is considered to be “consumed” by
2554 the carrier protocol implementation. All of it may be used by the sending implementation, or
2555 some of it may be transmitted in headers, or as part of a URL in the carrier protocol, but then
2556 used or consumed by the receiving implementation of the carrier protocol to direct the BTP
2557 message to a BTP-aware entity (BTP-aware in that it is capable of interpreting the BTP
2558 messages). The “additional information” of the target address will be part of the BTP
2559 message itself and used in some way by the receiving BTP-aware entity (it could be used to
2560 route the message on to some other BTP entity). Thus, for the target address, only the
2561 “additional information” field is transmitted in the BTP message and the “additional
2562 information” is opaque to parties other than the recipient.

2563
2564 For other addresses in BTP messages, all three components will be within the message.

2565
2566 All messages that concern a particular Superior:Inferior relationship have an identifier
2567 parameter for the target side as well as the target address. This allows full flexibility for
2568 implementation choices – an implementation can:

- 2569
- 2570 a) Use the same binding address and additional information for multiple business
2571 transactions, using the identifier parameter to locate the relevant state
2572 information;
 - 2573 b) Use the same binding address for multiple business transactions and use the
2574 additional information to locate the information; or
 - 2575 c) Use a different binding address for each business transaction.
- 2576

2577 Which of these choices is used is opaque to the entity sending the message – both parts of the
2578 address and the identifier originated at the recipient of this message (and were transmitted as
2579 parameters of earlier messages in the opposite direction).

2580
2581 BTP recovery requires that the state information for a Superior or Inferior is accessible after
2582 failure and that the peer can distinguish between temporary inaccessibility and the permanent
2583 non-existence of the state information. As is explained in “” below, BTP provides
2584 mechanisms – having a set of BTP addresses for some parameters, and the REDIRECT

2585 message – that make this possible, even if the recovered state information is on a different
2586 address to the original one (as may be the case if case c) above is used).

2587
2588

2589 **Request/response pairs**

2590

2591 Many of the messages combine in pairs as a request and its response. However, in some cases
2592 the response message is sent without a triggering request, or as a possible response to more
2593 than one type of request. To allow for this, the abstract message set treats each message as
2594 standalone; but where a request does expect a reply, a “reply-address” parameter will be
2595 present. For any message with a reply address parameter, in the case of certain errors, a
2596 FAULT message will be sent to the reply address instead of the expected reply.

2597

2598 Between Superior and Inferior the address of the peer is normally known (from the “superior-
2599 address” on an earlier CONTEXT or the “inferior-address” on a received ENROL). However,
2600 in some cases a message will be received for a Superior or Inferior that is not known – the
2601 state information no longer exists. This is not an exceptional condition but occurs when one
2602 side has either not created or has removed its persistent state in accordance with the
2603 procedures, but a message has got lost in a failure, and the peer still has state information.
2604 The response to a message for an unknown (and logically non-existent) Superior is
2605 SUPERIOR_STATE/unknown, for an unknown Inferior it is INFERIOR_STATE/unknown.
2606 However, since the intended target is unknown, there is no information to locate the peer,
2607 which sent the undeliverable message. To enable the receiver to reply with the appropriate
2608 *_STATE/unknown, all the messages between Superior and Inferior have a “senders-
2609 address” parameter. If a FAULT message is to be sent in response to message which (as an
2610 abstract message) has a “senders-address” parameter, the FAULT message is sent to that
2611 address.

2612

2613

Note – Both reply-address and senders-address may be absent when the
2614 carrier protocol itself has a request/response pattern. In these cases, the reply
2615 or sender address is implicitly that of the sender of the request (and thus the
2616 destination of a response)

2617 **Compounding messages**

2618

2619 BTP messages may be sent in combination with each other, or with other (application)
2620 messages. There are two cases:

2621

- 2622 a) Sending the messages together where the combination has semantic
2623 significance. One message is said to be “related to” the other – the combination
2624 is termed a “group”.
- 2625 b) Sending of the messages where the combination has no semantic significance,
2626 but is merely a convenience or optimisation. This is termed “bundling” – the
2627 combination is termed a “bundle”.

2628

2629 The form A&B is used to refer to a combination (group) where message B is sent in relation
2630 to A (“relation” is asymmetric). The form A+B is used to refer to A and B bundled together-
2631 the transmission of the bundle "A+B" is semantically identical to the transmission of A
2632 followed by the transmission of B.

2633
2634 Only certain combinations of messages are possible in a group, and the meaning of the
2635 relation is specifically defined for each such combination in the next section. A particular
2636 group is treated as a unit for transmission – it has a single target address. This is usually that
2637 of one of the messages in the group – the specification for the group defines which.

2638
2639 A “bundle” of messages may contain both unrelated messages and groups of related
2640 messages. The only constraint on which messages and groups can be bundled is that all have
2641 the same binding address, but may have different “additional information” values. (Messages
2642 within a related group may have different addresses, where the rules of their relatedness
2643 permit this). Unless constrained by the binding, any messages or groups that are to be sent to
2644 the same binding address may be bundled – the fact that the binding addresses are the same is
2645 a necessary and sufficient condition for the sender to determine that the messages can be
2646 bundled.

2647
2648 A particular and important case of related messages is where a BTP CONTEXT message is
2649 sent related to an application message. In this case, the target of the application message
2650 defines the destination of the CONTEXT message. The receiving implementation may in fact
2651 remove the CONTEXT before delivering the application message to the application (Service)
2652 proper, but from the perspective of the sender, the two are sent to the same place.
2653 The compounding mechanisms, and the multi-part address structures, support the “one-wire”
2654 and “one-shot” communication patterns.

2655
2656 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,
2657 including the associated application messages, pass via the same “endpoints”. These
2658 “endpoints” may in fact be relays, routing messages on to particular actors within their
2659 domain. The onward routing will require some further addressing, but this has to be opaque to
2660 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors
2661 in its domain have the relay’s address as their binding address, and any routing information it
2662 will need in its own domain is placed in the additional information. (This may involve the
2663 relay changing addresses in messages as they pass through it on the way out). On receiving a
2664 message, it determines the within-domain destination from the received additional
2665 information (which is thus rewritten) and forwards the message appropriately. The sender is
2666 unaware of this, and merely sees addresses with the same binding address, which it is
2667 permitted to bundle. The content of the “additional information” is a matter only for the relay
2668 – it could put an entire BTP address in there, or other implementation-defined information.
2669 Note that a quite different one-wire implementation can be constructed where there is no
2670 relaying, but the receiving entity effectively performs all roles, using the received identifiers
2671 to locate the appropriate state.

2672
2673 “One-shot” communication makes it possible to send an application message, receive the
2674 application reply, enrol an Inferior to be responsible for the confirm/cancel of the operations
2675 of those message and inform the Superior that the Inferior is prepared, all in one two-way

2676 exchange across the network (e.g. one request/reply of a carrier protocol).. The application
2677 request is sent with a related CONTEXT message. The application response is sent with a
2678 relation group of CONTEXT_REPLY/related, ENROL/no-rsp-req message and a
2679 PREPARED message. This is possible even if the Superior address is different from the
2680 address of the application element that sends the original message (if the application
2681 exchange is request/reply, there may not even be an identifiable address for the application
2682 element). The target addresses of the ENROL and PREPARED (the Superior address) are not
2683 transmitted; the actor that was originally responsible for adding the CONTEXT to the
2684 outbound application message remembers the Superior address and forwards the ENROL and
2685 PREPARED appropriately.

2686
2687 With “one-shot”, if there are multiple Inferiors created as a result of a single application
2688 message, there is an ENROL and PREPARED message for each sent related to the
2689 CONTEXT_REPLY. If an operation fails, a CANCELLED message is sent instead of a
2690 PREPARED.

2691
2692 If the CONTEXT has “superior-type” of “atom”, then subsequent messages to the same
2693 Service, with the same related CONTEXT/atom, can have their associated operations put
2694 under the control of the same Inferior, and only a CONTEXT_REPLY/completed is sent back
2695 with the response (if the new operations fail, it will be necessary to send back
2696 CONTEXT_REPLY/repudiated, or send CANCELLED). If the “superior type” on the
2697 CONTEXT is “cohesive”, each operation will require separate enrolment.

2698
2699 Whether the “one-shot” mechanism is used is determined by the implementation on the
2700 responding (Inferior) side. This may be subject to configuration and may also be constrained
2701 by the application or by the binding in use.

2702

2703 **Extensibility**

2704

2705 To simplify interoperation between implementations of this edition of BTP with
2706 implementations of future editions, the “must-be-understood” sub-parameter as specified for
2707 Qualifiers may be defined for use with any parameter added to an existing message in a future
2708 revision of this specification. The default for “must-be-understood” shall be “true”, so an
2709 implementation receiving an unrecognised parameter without a “false” value for “must-be-
2710 understood” shall not accept it (the FAULT value “UnrecognisedParameter” is available, but
2711 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If
2712 “must-be-understood” with the value “false” is present as a sub-parameter of a parameter in
2713 any message, a receiving implementation **should** ignore the parameter.

2714

2715 How the sub-parameter is associated with the new parameter is determined by the particular
2716 binding.

2717

2718 No special mechanism is provided to allow for the introduction of completely new messages.

2719

2720 **Messages**

2721

2722
2723
2724
2725
2726

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

2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757

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.

2758 The messages in this section are used between various roles. CONTEXT message is used in
 2759 the Initiator:Factory relationship (when it is related to BEGIN or to BEGUN), and related to
 2760 an application ‘message’ to propagate the business transaction between parts of the
 2761 application. CONTEXT_REPLY is used as the reply to a CONTEXT.REQUEST_STATUS
 2762 can be issued to, and STATUS returned by any of Decider, Superior or Inferior. FAULT can
 2763 be used on any relationship to indicate an error condition back to the sender of a message.
 2764

2765 CONTEXT

2766 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more
 2767 application messages. (The means by which this relationship is represented is determined by
 2768 the binding and the binding mechanisms of the application protocol.) The “superior-type”
 2769 parameter identifies whether the Superior will apply the same decision to all Inferiors
 2770 enrolled using the same superior identifier (“superior-type” is “atom”) or whether it may
 2771 apply different decisions (“superior-type” is “cohesion”).
 2772
 2773

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

2774
 2775 **superior-address** the address to which ENROL and other messages from an
 2776 enrolled Inferior are to be sent. This can be a set of alternative addresses.
 2777
 2778 **superior-identifier** identifies the Superior. This shall be globally unambiguous.
 2779
 2780 **superior-type** identifies whether the CONTEXT refers to a Cohesion or an
 2781 Atom. Default is atom.
 2782
 2783 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
 2784 timelimit” is carried by CONTEXT.
 2785
 2786 **reply-address** the address to which a replying CONTEXT_REPLY is to be sent.
 2787 This may be different each time the CONTEXT is transmitted – it refers to the
 2788 destination of a replying CONTEXT_REPLY for this particular transmission of
 2789 the CONTEXT.
 2790
 2791 There is no “target-address” parameter for CONTEXT as it is only transmitted in relation to
 2792 the application messages, BEGIN and BEGUN.
 2793

2794 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the
2795 “superior-type” with the appropriate value.
2796

2797

2798 CONTEXT_REPLY

2799

2800 CONTEXT_REPLY is sent after receipt of CONTEXT (related to application message(s)) to
2801 indicate whether all necessary enrolments have already completed (ENROLLED has been
2802 received) or will be completed by ENROL messages sent in relation to the
2803 CONTEXT_REPLY or if an enrolment attempt has failed. CONTEXT_REPLY may be sent
2804 related to an application message (typically the response to the application message related to
2805 the CONTEXT). In some bindings the CONTEXT_REPLY may be implicit in the application
2806 message. [CONTEXT_REPLY is used in some of the related groups to allow BTP messages
2807 to be sent to a Superior with an application message.](#)
2808

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

2809

2810

2811

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

2812

2813

2814

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>incomplete</i>	Further enrolments are possible (used only in related groups with other BTP messages)
<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.

2815

2816

2817

qualifiers standardised or other qualifiers.

2818

2819

2820

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

2821 The form CONTEXT_REPLY/completed, CONTEXT_REPLY/related and
 2822 CONTEXT_REPLY/repudiated refer to CONTEXT_REPLY messages with status having the
 2823 appropriate value. The form CONTEXT_REPLY/ok refers to either of
 2824 CONTEXT_REPLY/completed or CONTEXT_REPLY/related.

2825
 2826 If there are no necessary enrolments (e.g. the application messages related to the received
 2827 CONTEXT did not require the enrolment of any Inferiors), then
 2828 CONTEXT_REPLY/completed is used.

2829
 2830 If a CONTEXT_REPLY/repudiated is received, the receiving implementation **must** ensure
 2831 that the business transaction will not be confirmed.

2832
 2833

2834 REQUEST_STATUS

2835
 2836 Sent to an Inferior, Superior or to a Decider to ask it to reply with STATUS. The receiver
 2837 may reject the request with a FAULT(StatusRefused).
 2838

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

2839
 2840 **target identifier** The identifier for the business transaction, or part of business
 2841 transaction whose status is sought. If the target-address is a “decider-address”,
 2842 this parameter shall be the “transaction-identifier” on the BEGUN message. If the
 2843 “target-address” is an “inferior-address”, this parameter shall be the “inferior-
 2844 identifier” on the ENROL message. If the “target-address” is a “superior-
 2845 address”, this parameter shall be the “superior-identifier” on the CONTEXT.

2846
 2847 **qualifiers** standardised or other qualifiers.

2848
 2849 **target-address** the address to which the REQUEST_STATUS message is sent.
 2850 This can be any of “decider-address”, “inferior-address” or “superior-address”.

2851
 2852 **reply-address** the address to which the replying STATUS should be sent.

2853
 2854 Types of FAULT possible (sent to “reply-address”)

2855
 2856 *General*

2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867

Redirect – if the intended target now has a different address
StatusRefused – if the receiver is not prepared to report its status to the sender of this message
UnknownTransaction – if the target-identifier is unknown

STATUS

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

2868
2869
2870
2871
2872
2873
2874
2875
2876
2877

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

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

2878
2879
2880
2881
2882
2883
2884
2885

qualifiers standardised or other qualifiers.

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

Types of FAULT possible

2886
2887
2888
2889
2890
2891
2892

General

FAULT

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

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

2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906

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	None
<i>InvalidDecider</i>	The address the message was sent to is not valid (at all or for this Terminator and transaction identifier)	The address
<i>InvalidInferior</i>	The "inferior-identifier" in the message or at least one "inferior-identifier"s in an "inferior-list" parameter is not known or does not identify a known Inferior	One or more invalid identifiers
<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 None
<i>InvalidTerminator</i>	The address the message was sent to is not valid (at all or for this Decider and transaction identifier)	The address
<i>UnknownParameter</i>	A BTP message has been received with an unrecognised parameter	None
<i>UnknownTransaction</i>	The transaction-identifier is unknown	The transaction-identifier
<i>UnsupportedQualifier</i>	A qualifier has been received that is not recognised and on which "must-be-Understood" is "true".	Qualifier group and name
<i>WrongState</i>	The message has arrived when the recipient or the transaction identified by a related CONTEXT is in an invalid state.	None

2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920

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

qualifiers standardised or other qualifiers.

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

2921
2922
2923

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

2924

REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES

2925
2926
2927
2928
2929
2930
2931
2932
2933
2934

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

2935
2936

Messages used in the outcome relationships

2937
2938

ENROL

2939
2940
2941
2942

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

Parameter	type
superior-identifier	Identifier
response-requested	Boolean
inferior-address	Set of BTP addresses
inferior-identifier	Identifier

qualifiers	List of qualifiers
target-address	BTP address
reply-address	BTP address

2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983

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

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

inferior-address the address to which PREPARE, CONFIRM, CANCEL and SUPERIOR_STATE messages for this Inferior are to be sent.

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

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

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

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

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

- General*
- InvalidSuperior* – if “superior-identifier” is unknown
- Redirect* – if the Superior now has a different superior-address
- DuplicateInferior* – if inferior with at least one of the set “inferior-address” the same and the same “inferior-identifier” is already enrolled
- WrongState* – if it is too late to enrol new Inferiors (generally if the Superior has already sent a PREPARED message to its superior or terminator, or if it has already issued CONFIRM to other Inferiors).

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

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

2984
2985
2986
2987
2988
2989

ENROLLED

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

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

2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014

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

qualifiers standardised or other qualifiers.

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”s if the “reply-address” was empty)

sender-address the address from which the ENROLLED is sent. This is an address of the Superior.

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
superior-identifier	identifier
inferior-identifier	identifier
response-requested	Boolean
qualifiers	List of qualifiers

target-address	BTP address
sender-address	BTP address

3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051

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

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

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

qualifiers standardised or other qualifiers.

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

sender-address the address from which the RESIGN is sent. This is an address of the Inferior.

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

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

General

- InvalidSuperior** – if “superior-identifier” is unknown
- InvalidInferior** – if no ENROL had been received for this “inferior-identifier”inferior-
- 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
inferior-identifier	Identifier
qualifiers	List of qualifiers
target-address	BTP address

sender-address	BTP address
----------------	-------------

3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078

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

qualifiers standardised or other qualifiers.

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

sender-address the address from which the RESIGNED is sent. This is an address of the Superior.

After receiving this message the Inferior will not receive any more messages with this “inferior-identifier”.

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

General

WrongState - if RESIGN has not been sent

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

3079
3080
3081
3082
3083
3084
3085
3086
3087

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

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

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

3088 **sender-address** the address from which the PREPARE is sent. This is an
3089 address of the Superior.

3090
3091 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or
3092 RESIGN.

3093
3094 Types of FAULT possible (sent to “sender-address”)

3095
3096 **General**
3097 **InvalidInferior** – if “inferior-identifier” is unknown, or an inferior-handle
3098 on the inferiors-list is unknown

3099 **WrongState** – if a CONFIRM or CANCEL has already been received by
3100 this Inferior.

3101
3102

3103 **PREPARED**

3104
3105 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when
3106 the Inferior has determined the operations associated with the Inferior can be confirmed and
3107 can be cancelled, as may be instructed by the Superior. The level of isolation is a local matter
3108 (i.e. it is the Inferiors choice, as constrained by the shared understanding of the application
3109 exchanges) – other access may be blocked, may see applied results of operations or may see
3110 the original state.

3111

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

3112

3113 **superior-identifier** the “superior-identifier” as on the ENROL message

3114

3115 **inferior-identifier** The “inferior-identifier” as on the ENROL message

3116

3117 **default-is cancel** if “true”, the Inferior states that if the outcome at the Superior
3118 is to cancel the operations associated with this Inferior, no further messages need
3119 be sent to the Inferior. If the Inferior does not receive a CONFIRM message, it
3120 will cancel the associated operations. The value “true” will invariably be used
3121 with a qualifier indicating under what circumstances (usually a timeout) an
3122 autonomous decision to cancel will be made. If “false”, the Inferior will expect

3123 a CONFIRM or CANCEL message as appropriate, even if qualifiers indicate that
3124 an autonomous decision will be made.

3125
3126 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
3127 timeout” may be carried by PREPARED.

3128
3129 **target-address** the address to which the PREPARED is sent. This will be the
3130 Superior address as on the ENROL message.

3131
3132 **sender-address** the address from which the PREPARED is sent. This is an
3133 address of the Inferior.

3134
3135 On sending a PREPARED, the Inferior undertakes to maintain its ability to confirm or cancel
3136 the effects of the associated operations until it receives a CONFIRM or CANCEL message.
3137 Qualifiers may define a time limit or other constraints on this promise. The “default-is
3138 cancel” parameter affects only the subsequent message exchanges and does not of itself state
3139 that cancellation will occur.

3140
3141 Types of FAULT possible (sent to “sender-address”)

3142
3143 **General**
3144 **InvalidSuperior** – if “superior-identifier” is unknown
3145 **InvalidInferior** – if no ENROL has been received for this “inferior-
3146 identifier”, or if RESIGN has been received from this Inferior

3147
3148 The form PREPARED/cancel refers to a PREPARED message with “default-is cancel” =
3149 “true”. The unqualified form PREPARED refers to a PREPARED message with “default-is
3150 cancel” = “false”.

3151
3152

3153 CONFIRM

3154
3155 Sent by the Superior to an Inferior from whom PREPARED has been received.
3156

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

3157
3158 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for
3159 this Inferior.
3160

3161 **qualifiers** standardised or other qualifiers.

3162

3163 **target-address** the address to which the CONFIRM message is sent. This will
3164 be the “inferior-address” from the ENROL message.

3165

3166 **sender-address** the address from which the CONFIRM is sent. This is an
3167 address of the Superior.

3168

3169 On receiving CONFIRM, the Inferior is released from its promise to be able to undo the
3170 operations of associated with the Inferior. The effects of the operations can be made available
3171 to everyone (if they weren’t already).

3172

3173 Types of FAULT possible (sent to “sender-address”)

3174

3175 *General*

3176 *InvalidInferior* – if “inferior-identifier” is unknown

3177 *WrongState* – if no PREPARED has been sent by, or if CANCEL has
3178 been received by this Inferior.

3179

3180

3181 **CONFIRMED**

3182

3183 Sent after the Inferior has applied the confirmation, both in reply to CONFIRM or when the
3184 Inferior has made an autonomous confirm decision, and in reply to a
3185 CONFIRM_ONE_PHASE if the Inferior decides to confirm its associated operations.

3186

3187

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

3188

3189 **superior-identifier** the “superior-identifier” as on the CONTEXT message.

3190

3191 **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.

3192

3193

3194 **confirm-received** “true” if CONFIRMED is sent after receiving a CONFIRM
3195 message; “false” if an autonomous confirm decision has been made and either if

3196 no CONFIRM message has been received or the implementation cannot
3197 determine if CONFIRM has been received (due to loss of state information in a
3198 failure).
3199

3200 **qualifiers** standardised or other qualifiers.
3201

3202 **target-address** the address to which the CONFIRMED is sent. This will be the
3203 Superior address as on the CONTEXT message.
3204

3205 **sender-address** the address from which the CONFIRMED is sent. This is an
3206 address of the Inferior.
3207

3208 Types of FAULT possible (sent to “sender-address”)
3209

3210 *General*

3211 *InvalidSuperior* – if “superior-identifier” is unknown
3212

3213 *InvalidInferior* – if no ENROL has been received for this “inferior-
3214 identifier”, or if RESIGN has been received from this Inferior.

3215 Note – A CONFIRMED message arriving before a CONFIRM message is
3216 sent, or after a CANCEL has been sent will occur when the Inferior has
3217 taken an autonomous decision and is not regarded as occurring in the wrong
3218 state. (The latter will cause a CONTRADICTION message to be sent.)

3219 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm-
3220 received” = “false”; CONFIRMED/response refers to a CONFIRMED message
3221 with “confirm-received” = ”true”.
3222
3223
3224

3225 **CANCEL**
3226

3227 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.
3228

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

3229 **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.
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

qualifiers standardised or other qualifiers.

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

sender-address the address from which the CANCEL is sent. This is an address of the Superior.

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 “sender-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

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

sender-address	BTP address
----------------	-------------

3271
3272 **superior-identifier** the “superior-identifier” as on the CONTEXT message.
3273
3274 **inferior-identifier** the inferior identifier as on the earlier ENROL message.
3275
3276 **qualifiers** standardised or other qualifiers.
3277
3278 **target-address** the address to which the CANCELLED is sent. This will be the
3279 Superior address as on the CONTEXT message.
3280
3281 **sender-address** the address from which the CANCELLED is sent. This is an
3282 address of the Inferior.

3283
3284 Types of FAULT possible (sent to “sender-address”)

3285
3286 **General**

3287 **InvalidSuperior** – if “superior-identifier” is unknown

3288 **InvalidInferior** – if no ENROL has been received for this “inferior-
3289 identifier”, or if RESIGN has been received from this Inferior

3290 **WrongState** – if CONFIRM has been sent
3291

3292 Note – A CANCELLED message arriving before a CANCEL message is
3293 sent, or after a CONFIRM has been sent will occur when the Inferior has
3294 taken an autonomous decision and is not regarded as occurring in the wrong
3295 state. (The latter will cause a CONTRADICTION message to be sent.)

3296
3297
3298 **CONFIRM_ONE_PHASE**
3299

3300 Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In
3301 this case the two-phase exchange is not performed between the Superior and Inferior and the
3302 outcome decision for the operations associated with the Inferior is determined by the Inferior.
3303

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

3304
3305 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for
3306 this Inferior.
3307
3308 **report hazard** Defines whether the superior wishes to be informed if a mixed
3309 condition occurs for the operations associated with the Inferior. If “report-
3310 hazard” is “true”, the Inferior will reply with HAZARD if a mixed condition
3311 occurs, or if the Inferior cannot determine that a mixed condition has not
3312 occurred. If “report-hazard” is false, the Inferior will report only its own decision,
3313 regardless of whether that decision was correctly and consistently applied.
3314 Default is false.
3315
3316 **qualifiers** standardised or other qualifiers.
3317
3318 **target-address** the address to which the CONFIRM_ONE_PHASE message is
3319 sent This will be the “inferior-address” on the ENROL message.
3320
3321 **sender-address** the address from which the CONFIRM_ONE_PHASE is sent.
3322 This is an address of the Superior.

3323
3324 CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom
3325 PREPARED has been received (subject to the requirement that there is only one enrolled
3326 Inferior).

3327
3328 Types of FAULT possible (sent to “sender-address”)
3329

3330 *General*
3331 *InvalidInferior* – if “inferior-identifier” is unknown
3332 *WrongState* – if a PREPARE has already been sent to this Inferior
3333

3334 **HAZARD**
3335

3336 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly
3337 and consistently cancel or confirm the operations in accord with the decision , or when the
3338 Inferior is unable to determine that a “mixed” condition has not occurred.
3339

3340 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there
3341 is a mixed condition within its associated operations or is unable to determine that there is not
3342 a mixed condition.
3343

3344 Note - If the Inferior makes its own autonomous decision then it signals that
3345 decision with CONFIRMED or CANCELLED and waits to receive a
3346 confirmatory CONFIRM or CANCEL, or a CONTRADICTION if the
3347 autonomous decision by the Inferior was the opposite of that made by the
3348 Superior.

3349

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

3350

3351

3352

3353

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

3354

3355

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

3356

3357

3358

3359

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.

3360

3361

qualifiers standardised or other qualifiers.

3362

3363

3364

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

3365

3366

sender-address the address from which the HAZARD is sent. This is an address of the Inferior.

3367

3368

3369

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

3370

General

3371

InvalidSuperior – if “superior-identifier” is unknown

3372

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

3373

3374

3375

3376

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

3377

3378

3379

CONTRADICTION

3380

3381

3382

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

3383 CONFIRMED or CANCELLED is received. CONTRADICTION is also sent in response to a
3384 HAZARD message.
3385

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

3386
3387 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for
3388 this Inferior.

3389
3390 **qualifiers** standardised or other qualifiers.

3391
3392 **target-address** the address to which the CONTRADICTION message is sent.
3393 This will be the “inferior-address” from the ENROL message.

3394
3395 **sender-address** the address from which the CONTRADICTION is sent. This is
3396 an address of the Superior.

3397
3398 Types of FAULT possible (sent to “sender-address”)
3399

3400 *General*
3401 *InvalidInferior* – if “inferior-identifier” is unknown
3402 *WrongState* – if neither CONFIRMED or CANCELLED has been sent
3403 by this Inferior
3404

3405 SUPERIOR_STATE

3406 Sent by a Superior as a query to an Inferior when
3407

- 3408
- 3409 1. in the active state
 - 3410 2. there is uncertainty what state the Inferior has reached (due to recovery from
3411 previous failure or other reason).

3412
3413 Also sent by the Superior to the Inferior in response to a received INFERIOR_STATE, in
3414 particular states.
3415

3416

Parameter	Type
inferior-identifier	Identifier
status	<i>see below</i>

Parameter	Type
response-requested	Boolean
qualifiers	List of qualifiers
target-address	BTP address
sender-address	BTP address

3417
3418
3419
3420
3421
3422
3423

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

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

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

3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437

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.

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

sender-address the address from which the SUPERIOR_STATE is sent. This is an address of the Superior.

3438 The Inferior, on receiving SUPERIOR_STATE with “response-requested = true, should reply
3439 in a timely manner by (depending on its state) repeating the previous message it sent or by
3440 sending INFERIOR_STATE with the appropriate status value.

3441
3442 A status of unknown shall only be sent if it has been determined for certain that the Superior
3443 has no knowledge of the Inferior, or (equivalently) it can be determined that the relationship
3444 with the Inferior was cancelled. If there could be persistent information corresponding to the
3445 Superior, but it is not accessible from the entity receiving an INFERIOR_STATE/*y (or
3446 other) message targeted to the Superior or that entity cannot determine whether any such
3447 persistent information exists or not, the response shall be Inaccessible.

3448
3449 SUPERIOR_STATE/unknown is also used as a response to messages, other than
3450 INFERIOR_STATE/*y that are received when the Inferior is not known (and it is known
3451 there is no state information for it).

3452
3453 The form SUPERIOR_STATE/abcd refers to a SUPERIOR_STATE message status having a
3454 value equivalent to “abcd” (for active, prepared-received, unknown and inaccessible) and
3455 with “response-requested” = “false”. SUPERIOR_STATE/abcd/y refers to a similar message,
3456 but with “response-requested” = “true”. The form SUPERIOR_STATE/*y refers to a
3457 SUPERIOR_STATE message with “response-requested” = “true” and any value for status.

3458
3459

3460 INFERIOR_STATE

3461
3462 Sent by an Inferior as a query when in the active state to a Superior, when (due recovery from
3463 previous failure or other reason) there is uncertainty what state the Superior has reached.

3464
3465 Also sent by the Inferior to the Superior in response to a received SUPERIOR_STATE, in
3466 particular states.

3467

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

3468

3469 **superior-identifier** The “superior-identifier” as used on the ENROL message

3470

3471 **inferior-identifier** The “inferior-identifier” as on the ENROL message

3472

3473 **status** states the current state of the Inferior for the atomic business transaction,
3474 which corresponds to the last message sent to the Superior by (or in the case of
3475 ENROL for) the Inferior
3476

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

3477
3478 **response-requested** “true” if INFERIOR_STATE is sent as a query at the
3479 Superior’s initiative; “false” if INFERIOR_STATE is sent in reply to a received
3480 SUPERIOR_STATE or other message. Can only be “true” if “status” is “active”
3481 or “prepared-received”. Default is “false”
3482

3483 **qualifiers** standardised or other qualifiers.
3484

3485 **target-address** the address to which the INFERIOR_STATE is sent. This will
3486 be the “target-address” as used the original ENROL message.
3487

3488 **sender-address** the address from which the INFERIOR_STATE is sent. This is
3489 an address of the Inferior.
3490

3491 The Superior, on receiving INFERIOR_STATE with “response-requested” = “true”, should
3492 reply in a timely manner by (depending on its state) repeating the previous message it sent or
3493 by sending SUPERIOR_STATE with the appropriate status value.
3494

3495 A status of “unknown” shall only be sent if it has been determined for certain that the Inferior
3496 has no knowledge of a relationship with the Superior. If there could be persistent information
3497 corresponding to the Superior, but it is not accessible from the entity receiving an
3498 SUPERIOR_STATE/*y (or other) message targetted on the Inferior or the entity cannot
3499 determine whether any such persistent information exists, the response shall be
3500 “inaccessible”.
3501

3502 INFERIOR_STATE/unknown is also used as a response to messages, other than
3503 SUPERIOR_STATE/*y that are received when the Inferior is not known (and it is known
3504 there is no state information for it).
3505

3506 A SUPERIOR_STATE/INFERIOR_STATE exchange that determines that one or both sides
3507 are in the active state does not require that the Inferior be cancelled (unlike some other two-

3508 phase commit protocols). The relationship between Superior and Inferior, and related
 3509 application elements may be continued, with new application messages carrying the same
 3510 CONTEXT. Similarly, if the Inferior is prepared but the Superior is active, there is no
 3511 required impact on the progression of the relationship between them.

3512
 3513 The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a
 3514 value equivalent to “abcd” (for active, unknown and inaccessible) and with “response-
 3515 requested” = “false”. INFERIOR_STATE/abcd/y refers to a similar message, but with
 3516 “response-requested” = “true”. The form INFERIOR_STATE/*y refers to a
 3517 INFERIOR_STATE message with “response-requested” = “true” and any value for status.

3518
 3519

3520 **REDIRECT**

3521
 3522 Sent when the address previously given for a Superior or Inferior is no longer valid and the
 3523 relevant state information is now accessible with a different address (but the same superior or
 3524 “inferior-identifier”).

3525

Parameter	Type
superior-identifier	Identifier
inferior-identifier	Identifier
old-address	Set of BTP addresses
new-address	Set of BTP addresses
qualifiers	List of qualifiers
target-address	BTP address

3526

3527 **superior-identifier** The “superior-identifier” as on the CONTEXT message and
 3528 used on an ENROL message. (present only if the REDIRECT is sent from the
 3529 Inferior).

3530

3531 **inferior-identifier** The “inferior-identifier” as on the ENROL message

3532

3533 **old-address** The previous address of the sender of REDIRECT. A match is
 3534 considered to apply if any of the “old-address” values match one that is already
 3535 known.

3536

3537 **new-address** The (set of alternatives) “new-address” values to be used for
 3538 messages sent to this entity.

3539

3540 **qualifiers** standardised or other qualifiers.

3541

3542 **target-address** the address to which the REDIRECT is sent. This is the address
3543 of the opposite side (superior/inferior) as given in a CONTEXT or ENROL
3544 message

3545
3546 If the actor whose address is changed is an Inferior, the “new-address” value replaces the
3547 “inferior-address” as present in the ENROL.

3548
3549 If the actor whose address is changed is a Superior, the “new-address” value replaces the
3550 Superior address as present in the CONTEXT message (or as present in any other mechanism
3551 used to establish the Superior:Inferior relationship).

3552
3553

3554 **Messages used in control relationships**

3555

3556 **BEGIN**

3557

3558 A request to a Factory to create a new Business Transaction. This may either be a new top-
3559 level transaction, in which case the Composer or Coordinator will be the Decider, or the new
3560 Business Transaction may be immediately made the Inferior within an existing Business
3561 Transaction (thus creating a sub-Composer or sub-Coordinator).

3562

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

3563

3564 **transaction-type** identifies whether a new Cohesion or new Atom is to be
3565 created; this value will be the “superior-type” in the new CONTEXT

3566

3567 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
3568 timelimit” may be present on BEGIN, to set the timelimit for the new business
3569 transaction and will be copied to the new CONTEXT. The standard qualifier
3570 “Inferior name” may be present if there is a CONTEXT related to the BEGIN.

3571

3572 **target-address** the address of the entity to which the BEGIN is sent. How this
3573 address is acquired and the nature of the entity are outside the scope of this
3574 specification.

3575

3576 **reply-address** the address to which the replying BEGUN and related
3577 CONTEXT message should be sent.

3578

3579 A new top-level Business Transaction is created if there is no CONTEXT related to the
 3580 BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is
 3581 created if the CONTEXT message for the existing Business Transaction is related to the
 3582 BEGIN. In this case, the Factory is responsible for enrolling the new Composer or
 3583 Coordinator as an Inferior of the Superior identified in that CONTEXT.
 3584

3585 Note – This specification does not provide a standardised means to
 3586 determine which of the Inferiors of a sub-Composer are in its confirm set.
 3587 This is considered part of the application:inferior relationship.

3588 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction-type” having
 3589 the corresponding value.
 3590

3591 Types of FAULT possible (sent to “reply-address”)
 3592
 3593

General

Redirect – if the Factory now has a different address

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

3599

BEGUN

3600

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

3601
 3602
 3603
 3604

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

3605

decider-address 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

3610
 3611

inferior-address for a non-top-most transaction (a CONTEXT was related to the BEGIN), this is the “inferior-address” used in the enrolment with the Superior

3612

3613

3614 identified by the CONTEXT related to the BEGIN. The parameter is optional
3615 (implementor's choice) if this is not a top-most transaction; it shall be absent if
3616 this is a top-most transaction.

3617
3618 **transaction-identifier** if this is a top-most transaction, this is an globally-
3619 unambiguous identifier for the new Decider (Composer or Coordinator). If this is
3620 not a top-most transaction, the transaction-identifier shall be the inferior-
3621 identifier used in the enrolment with the Superior identified by the CONTEXT
3622 related to the BEGIN.
3623

3624 Note – The “transaction-identifier” may be identical to the “superior-
3625 identifier” in the CONTEXT that is related to the BEGUN

3626
3627 **qualifiers** standardised or other qualifiers.

3628
3629 **target-address** the address to which the BEGUN is sent. This will be the “reply-
3630 address” from the BEGIN.

3631
3632 At implementation option, the “decider-address” and/or “inferior-address” and the “superior-
3633 address” in the related CONTEXT may be the same or may be different. There is no general
3634 requirement that they even use the same bindings. Any may also be the same as the “target-
3635 address” of the BEGIN message (the identifier on messages will ensure they are applied to
3636 the appropriate Composer or Coordinator).

3637
3638 No FAULT messages are issued on receiving BEGUN.

3639 **PREPARE_INFERIORS**

3640
3641 Sent from a Terminator to a Decider, but only if it is a Cohesion Composer, to tell it to
3642 prepare all or some of its inferiors, by sending PREPARE to any that have not already sent
3643 PREPARED, RESIGN or CANCELLED to the Decider (Composer) on its relationships as
3644 Superior. If the inferiors-list parameter is absent, the request applies to all the inferiors; if the
3645 parameter is present, it applies only to the identified inferiors of the Decider (Composer).
3646
3647

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

3648

3649 **transaction identifier** identifies the Decider and will be the transaction-identifier
3650 from the BEGUN message.

3651
3652 **inferiors-list** defines which of the Inferiors of this Decider preparation is
3653 requested for, using the “inferior-identifiers” as on the ENROL received by the
3654 Decider (in its role as Superior). If this parameter is absent, the PREPARE
3655 applies to all Inferiors.

3656
3657 **qualifiers** standardised or other qualifiers.

3658
3659 **target-address** the address to which the PREPARE_INFERIORS message is
3660 sent. This will be the decider-address from the BEGUN message.

3661
3662 **reply-address** the address of the Terminator sending the
3663 PREPARE_INFERIORS message.

3664
3665 For all Inferiors identified in the inferiors-list parameter (all Inferiors if the parameter is
3666 absent), from which none of PREPARED, CANCELLED or RESIGNED has been received,
3667 the Decider shall issue PREPARE. It will reply to the Terminator, using the “reply-address”
3668 on the PREPARE_INFERIORS message, sending an INFERIOR_STATUSES message
3669 giving the status of the Inferiors identified on the inferiors-list parameter (all of them if the
3670 parameter was absent).

3671
3672 If one or more of the “inferior-identifier”s in the "inferior-list" is unknown (does not
3673 correspond to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. It is an
3674 implementation option whether CANCEL is sent to any of the Inferiors that are validly
3675 identified in the "inferiors-list".

3676
3677
3678 Types of FAULT possible (sent to Superior address)

3679
3680 *General*
3681 *InvalidDecider* – if Decider address is unknown
3682 *Redirect* – if the Decider now has a different “decider-address”
3683 *UnknownTransaction* – if the transaction-identifier is unknown
3684 *InvalidInferior* – if one or more inferior-handles on the inferiors-list is
3685 unknown
3686 *WrongState* – if a CONFIRM_TRANSACTION or
3687 CANCEL_TRANSACTION has already been received by this
3688 Composer.

3689
3690 The form PREPARE_INFERIORS/all refers to a PREPARE_INFERIORS message where
3691 the “inferiors-list” parameter is absent. The form PREPARE_INFERIORS/specific refers to a
3692 PREPARE_INFERIORS message where the “inferiors-list” parameter is present.

3693
3694

3695
3696
3697
3698
3699
3700

CONFIRM_TRANSACTION

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

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

3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730

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

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

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

qualifiers standardised or other qualifiers.

target-address the address to which the CONFIRM_TRANSACTION message is sent. This will be the “decider-address” on the BEGUN message.

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

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

3731
3732 Any Inferiors from which RESIGN is received are not counted in the confirm-set.
3733
3734 If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED
3735 has not been received, PREPARE shall be issued to that Inferior.
3736

3737 NOTE -- If PREPARE has been sent but PREPARED not yet received from
3738 an Inferior in the confirm-set, it is an implementation option whether and
3739 when to re-send PREPARE. The Superior implementation may choose to re-
3740 send PREPARE if there are indications that the earlier PREPARE was not
3741 delivered.

3742
3743 A confirm decision may be made only if PREPARED has been received from all Inferiors in
3744 the "confirm-set". The making of the decision shall be persistent (and if it is not possible to
3745 persist the decision, it is not made). If there is only one remaining Inferior in the "confirm
3746 set" and PREPARE has not been sent to it, CONFIRM_ONE_PHASE may be sent to it.
3747

3748 All remaining Inferiors that are not in the confirm set shall be cancelled.
3749

3750 If a confirm decision is made and "report-hazard" was "false", a
3751 TRANSACTION_CONFIRMED message shall be sent to the "reply-address".
3752

3753 If a cancel decision is made and "report-hazard" was "false", a
3754 TRANSACTION_CANCELLED message shall be sent to the "reply-address".
3755

3756 If "report-hazard" was "true", TRANSACTION_CONFIRMED shall be sent to the "reply-
3757 address" after CONFIRMED has been received from each Inferior in the confirm-set and
3758 CANCELLED or RESIGN from each and any Inferior not in the confirm-set.

3759
3760 If "report-hazard" was "true" and any HAZARD or contradictory message was received (i.e.
3761 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in
3762 the confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent
3763 to the "reply-address".
3764

3765 If one or more of the "inferior-identifier"s in the "inferior-list" is unknown (does not
3766 correspond to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. The Decider
3767 shall not make a confirm decision and shall not send CONFIRM to any Inferior.
3768

3769 Types of FAULT possible (sent to "reply-address")
3770

3771 **General**

3772 **InvalidDecider** – if Decider address is unknown

3773 **Redirect** – if the Decider now has a different "decider-address"

3774 **UnknownTransaction** – if the transaction-identifier is unknown

3775 *InvalidInferior* – if one or more inferior handles in the inferiors-list is
 3776 unknown
 3777 *WrongState* – if a CANCEL_TRANSACTION has already been
 3778 received .
 3779

3780 The form CONFIRM_TRANSACTION/all refers to a CONFIRM_TRANSACTION message
 3781 where the “inferiors-list” parameter is absent. The form
 3782 CONFIRM_TRANSACTION/specific refers to a CONFIRM_TRANSACTION message
 3783 where the “inferiors-list” parameter is present.
 3784

3785 **TRANSACTION_CONFIRMED**

3786
 3787 A Decider sends TRANSACTION_CONFIRMED to a Terminator in reply to
 3788 CONFIRM_TRANSACTION if all of the confirm-set confirms (and, for a Cohesion, all other
 3789 Inferiors cancel) without reporting hazards, or if the Decider made a confirm decision and the
 3790 CONFIRM_TRANSACTION had a “report-hazards” value of “false”.
 3791

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

3792
 3793 **transaction-identifier** the “transaction-identifier” as on the BEGUN message
 3794 (i.e. the identifier of the Decider as a whole).
 3795

3796 **qualifiers** standardised or other qualifiers.
 3797

3798 **target-address** the address to which the TRANSACTION_CONFIRMED is
 3799 sent., this will be the “reply-address” from the CONFIRM_TRANSACTION
 3800 message
 3801

3802 Types of FAULT possible (sent to “decider-address”)
 3803

3804 *General*

3805 *InvalidTerminator* – if Terminator address is unknown

3806 *UnknownTransaction* – if the transaction-identifier is unknown
 3807

3808 **CANCEL_TRANSACTION**

3809
 3810 Sent by a Terminator to a Decider at any time before CONFIRM_TRANSACTION has been
 3811 sent.
 3812

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

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

3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850

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.

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.

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.

If "report-hazard" was "false", a TRANSACTION_CANCELLED message shall be sent to the "reply-address".

If "report-hazard" was "true" and any HAZARD or CONFIRMED message was received, an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent to the "reply-address".

If "report-hazard" was "true", TRANSACTION_CANCELLED shall be sent to the "reply-address" after CANCELLED or RESIGN has been received from each Inferior.

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”

UnknownTransaction – if the transaction-identifier is unknown

3851 *WrongState* – if a CONFIRM_TRANSACTION has been received by
3852 this Composer.

3853
3854

CANCEL_INFERIORS

3855
3856
3857
3858
3859

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

3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878

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.

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.

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.

3879
3880
3881

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.

3882
3883
3884

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

3885 implementation option whether CANCEL is sent to any of the Inferiors that are validly
3886 identified in the "inferiors-list".

3887

3888 Types of FAULT possible (sent to Superior address)

3889

3890 *General*

3891 *InvalidDecider* – if Decider address is unknown

3892 *Redirect* – if the Decider now has a different "decider-address"

3893 *UnknownTransaction* – if the transaction-identifier is unknown

3894 *InvalidInferior* – if one or more inferior-handle on the inferiors-list is

3895

unknown

3896

WrongState – if a CONFIRM_TRANSACTION or

3897

CANCEL_TRANSACTION has been received by this Composer.

3898

3899

3900

3901 **TRANSACTION_CANCELLED**

3902

3903 A Decider sends TRANSACTION_CANCELLED to a Terminator in reply to
3904 CANCEL_TRANSACTION or in reply to CONFIRM_TRANSACTION if the Decider
3905 decided to cancel. In both cases, TRANSACTION_CANCELLED is used only if all Inferiors
3906 cancelled without reporting hazards or the CANCEL_TRANSACTION or
3907 CONFIRM_TRANSACTION had a "report-hazard" value of "false."
3908

Parameter

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

3909

3910 **transaction-identifier** the "transaction-identifier" as on the BEGUN message
3911 (i.e. the identifier of the Decider as a whole).

3912

3913 **qualifiers** standardised or other qualifiers.

3914

3915 **target-address** the address to which the TRANSACTION_CANCELLED is
3916 sent. This will be the "reply-address" from the CANCEL_TRANSACTION or
3917 CONFIRM_TRANSACTION message.

3918

3919 Types of FAULT possible (sent to "decider-address")

3920

3921 *General*

3922 *InvalidTerminator* – if Terminator address is unknown

3923

UnknownTransaction – if the transaction-identifier is unknown

3924

3925
3926
3927
3928
3929
3930
3931
3932
3933
3934

REQUEST_INFERIOR_STATUSES

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

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

3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960

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

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

qualifiers standardised or other qualifiers.

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

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

Types of FAULT possible (sent to reply-address)

General

Redirect – if the intended target now has a different address

StatusRefused – if the receiver is not prepared to report its status to the

3961 sender of this message. This "fault-type" shall not be issued when a Decider
3962 receives REQUES_STATUSES from the Terminator.

3963 **UnknownTransaction** – if the transaction-identifier is unknown

3964

3965

3966 The form REQUEST_INFERIOR_STATUSES/all refers to a REQUEST_STATUS with the
3967 inferiors-list absent. The form REQUEST_INFERIOR_STATUS/specific refers to a
3968 REQUEST_INFERIOR_STATUS with the inferiors-list present.

3969

3970 INFERIOR_STATUSES

3971

3972 Sent by a Decider to report the status of all or some of its inferiors in response to a
3973 REQUEST_INFERIOR_STATUSES, PREPARE_INFERIORS, CANCEL_INFERIORS,
3974 CANCEL_TRANSACTION with "report-hazard" value of "true" and
3975 CONFIRM_TRANSACTION with "report-hazard" value of "true". It is also used by any
3976 actor in response to a received REQUEST_INFERIOR_STATUSES to report the status of
3977 inferiors, if there are any.

3978

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

3979

3980

3981

3982

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

3983

3984

3985

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

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

3986

3987

3988

3989

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

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

3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002

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.

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

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).

4003
4004 Types of FAULT possible (sent to “decider-address”)

4005
4006 *General*
4007 *InvalidTerminator* – if Terminator address is unknown
4008 *UnknownTransaction* – if the transaction-identifier is unknown

4009
4010
4011
4012

4013 **Groups – combinations of related messages**

4014
4015 The following combinations of messages form related groups, for which the meaning of the
4016 group is not just the aggregate of the meanings of the messages. The “&” notation is used to
4017 indicate relatedness. Messages appearing in parentheses in the names of groups in this section
4018 indicate messages that may or may not be present. The notation A & B / & C in a group name
4019 in this section indicates a group that contains A and B or A and C or A, B and C, possibly
4020 with any of those appearing more than once.

4021 4022 **CONTEXT & application message**

4023
4024 **Meaning:** the transmission of the application message is deemed to be part of the
4025 business transaction identified by the CONTEXT. The exact effect of this for application
4026 work implied by the transmission of the message is determined by the application – in
4027 many cases, it will mean the effects of the application message are to be subject to the
4028 outcome delivered to an enrolled Inferior, thus requiring the enrolment of a new Inferior
4029 if no appropriate Inferior is enrolled or if the CONTEXT is for cohesion.

4030
4031 **target-address:** the “target-address” is that of the application message. It is not required
4032 that the application address be a BTP address (in particular, there is no BTP-defined
4033 “additional information” field – the application protocol (and its binding) may or may not
4034 have a similar construct).

4035
4036 There may be multiple application messages related to a single CONTEXT message. All
4037 the application messages so related are deemed to be part of the business transaction
4038 identified by the CONTEXT. This specification does not imply any further relatedness
4039 among the application messages themselves (though the application might).

4040
4041 The actor that sends the group shall retain knowledge of the Superior address in the
4042 CONTEXT. If the CONTEXT is a CONTEXT/atom, the actor shall also keep track of
4043 transmitted CONTEXTs for which no CONTEXT_REPLY has been received.

4044
4045 If the CONTEXT is a CONTEXT/atom, the actor receiving the CONTEXT shall ensure
4046 that a CONTEXT_REPLY message is sent back to the “reply-address” of the CONTEXT
4047 with the appropriate completion status.

4048

4049
4050
4051
4052
4053

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.

4054
4055
4056

CONTEXT_REPLY & ENROL

4057
4058
4059
4060
4061

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.

4062
4063
4064
4065

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).

4066
4067

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

4068
4069
4070
4071
4072

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.

4073
4074
4075

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

4076
4077
4078
4079
4080
4081
4082
4083
4084

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.

4085
4086
4087
4088
4089

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.

4090
4091
4092

A CONTEXT_REPLY & ENROL group may contain multiple ENROL messages, for several Inferiors. Each ENROL shall be forwarded and an ENROLLED reply received

4093 before the Superior is allowed to confirm if the “completion-status” in the
4094 CONTEXT_REPLY was “related”.

4095
4096 When the group is constructed, if the CONTEXT had “superior-type” value of “atom”,
4097 the “completion-status” of the CONTEXT_REPLY shall be “related”. If the “superior-
4098 type” was “cohesive”, the “completion-status” shall be “~~incomplete~~” or “related” (as
4099 required by the application). If the value is “~~completed~~incomplete”, the actor receiving
4100 the group shall forward the ENROLs, but is not required to ~~(though it may)~~ prevent
4101 confirmation (though it may do so).

4102

4103 CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED

4104

4105 This combination is characterised by a related CONTEXT_REPLY and either or both of
4106 PREPARED and CANCELLED, with or without ENROL.

4107

4108 **Meaning:** If ENROL is present, the meaning and required processing is the same as for
4109 CONTEXT_REPLY & ENROL. The PREPARED or CANCELLED message(s) are
4110 forwarded to the Superior identified in the CONTEXT message this CONTEXT_REPLY
4111 is replying to.

4112

4113

Note – the combination of CONTEXT_REPLY & ENROL & CANCELLED
4114 may be used to force cancellation of an atom

4115

4116 **target-address:** the “target-address” is that of the CONTEXT_REPLY. This will be the
4117 “reply-address” of the CONTEXT message (in many cases, including request/reply
4118 application exchanges, this address will usually be implicit).

4119

4120 The “target-address” of the PREPARED and CANCELLED message is omitted – they
4121 will be sent to the Superior identified in the earlier CONTEXT message.

4122

4123 The actor receiving the group forwards the PREPARED or CANCELLED message to the
4124 Superior in as for an ENROL, using the retained Superior address from the CONTEXT
4125 sent earlier, except there is no reply required from the Superior.

4126

4127 If (as is usual) an ENROL and PREPARED or CANCELLED message are for the same
4128 Inferior, the ENROL shall be sent first, but the actor need not wait for the ENROLLED to
4129 come back before sending the PREPARED or CANCELLED (so an
4130 ENROL+PREPARED bundle from this actor to the Superior could be used).

4131

4132 The group can contain multiple ENROL, PREPARED and CANCELLED messages.
4133 Each PREPARED and CANCELLED message will be for a different Inferior.. There is
4134 no constraint on the order of their forwarding, except that ENROL and PREPARED or
4135 CANCELLED for the same Inferior shall be delivered to the Superior in the order
4136 ENROL first, followed by the other message for that Inferior.

4137

4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183

CONTEXT_REPLY & ENROL & application message (& PREPARED)

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

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

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

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

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

BEGUN & CONTEXT

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

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

BEGIN & CONTEXT

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

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

4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210

Standard qualifiers

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

Transaction timelimit

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

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

The qualifier is propagated on a CONTEXT message.

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

The “Content” shall contain the following field:

Content field	Type
Timelimit	Integer

4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226

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

Inferior timeout

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

It should be noted that BTP recognises the possibility that an Inferior may be forced to apply a confirm or cancel decision before the CONFIRM or CANCEL is received and before this timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,

4227 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
4228 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
4229 expiry of this timeout, is liable to cause contradictory decisions across the business
4230 transaction. BTP ensures that at least the occurrence of such a contradiction will be
4231 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic
4232 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
4233 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
4234 change in the probability that it will.

4235
4236 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
4237 intended decision, only that is at liberty to do so. An implementation may choose to only
4238 apply the decision if there is contention for the underlying resource, for example.
4239 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for
4240 the business transaction are made before these timeouts expire (and allow a margin of error
4241 for network latency etc.).

4242
4243 The qualifier may be present on a PREPARED message. If the PREPARED message has the
4244 “default-is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall
4245 have the value “cancel”.

4246
4247 The “Qualifier name” shall be “inferior-timeout” .

4248
4249 The “Content” shall contain the following fields:

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

4250
4251
4252 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
4253 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
4254 effects of the associated operations, as ordered by the receiving Superior.

4255
4256 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an
4257 autonomous decision is made.

4258 **Minimum inferior timeout**

4259
4260 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the
4261 Inferior. If a Superior knows that the decision for the business transaction will not be
4262 determined for some period, it can require that Inferiors do not send PREPARED messages
4263 with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to
4264 send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with
4265 CANCELLED.
4266
4267

4268 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If
4269 present on more than one, and with different values of the MinimumTimeout field, the value
4270 on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall
4271 prevail over either of the others.

4272
4273 The “Qualifier name” shall be “minimum-inferior-timeout”.

4274
4275 The “Content” shall contain the following field:
4276

Content field	Type
MinimumTimeout	Integer

4277
4278 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be
4279 acceptable in the Inferior timeout qualifier on an answering PREPARED message.

4280
4281 **Inferior name**

4282
4283 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on
4284 INFERIOR_STATUSES and thus allow the Terminator to determine which Inferior (of the
4285 Composer or Coordinator) is related to which application work. This is in addition to the
4286 “inferior-identifier” field. The name can be human-readable and can also be used in fault
4287 tracing, debugging and auditing.

4288
4289 The name is never used by the BTP actors themselves to identify each other or to direct
4290 messages. (The BTP actors use the addresses and the identifiers in the message parameters
4291 for those purposes.)

4292
4293 This specification makes no requirement that the names are unambiguous within any scope
4294 (unlike the globally unambiguous “inferior-identifier” on ENROLLED and BEGUN). Other
4295 specifications, including those defining use of BTP with a particular application may place
4296 requirements on the use and form of the names. (This may include reference to information
4297 passed in application messages or in other, non-standardised, qualifiers.)

4298
4299 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item
4300 in INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if
4301 present, the same qualifier value **should** be included in the consequent ENROL. If
4302 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an
4303 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.

4304
4305 The “Qualifier -name” shall be “inferior-name”

4306
4307 The “Content” shall contain the following fields:
4308

Content field	Type
---------------	------

inferior-name String

4309

4310

4311

Inferior name the name assigned to the enrolling Inferior.

4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357

State Tables

~~Explanation of the state tables~~

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

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

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

In the state tables, 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. Normally, on entry to a state that allows the sending of any message other than one of the *_STATE messages, the implementation will send that message – failure to do so will cause the relationship to lock up. The message can be resent if the implementation determines that the original message (or the next message sent in reply) may have been lost.

Status queries

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

Note that a Superior (or some entity standing in for a now-extinct Superior) uses SUPERIOR_STATE/unknown to reply to messages received from an Inferior where the Superior:Inferior relationship is in an unknown (using state “Y1”). The *_STATE messages with a “state” value “inaccessible” can be used as a reply when **any** message is received and the implementation is temporarily unable to determine whether the relationship is known or what the state is. ~~Other than these cases, the *_STATE messages with “response requested” equal to “false” are only sent when the other message with “response requested” equal to~~

4358 ~~“true” has been received and no other message has been sent.~~ [Receipt of the](#)
4359 [* STATE/inaccessible messages is not shown in the tables and has no effect on the state at](#)
4360 [the receiving side \(though it may cause the implementation to resend its own message after](#)
4361 [some interval of its own choosing\).](#)

4363 Decision events

4364
4365 The persistent state changes (equivalent to logging in a regular transaction system) and some
4366 other events are modelled as “decision events” (e.g. “decide to confirm”, “decide to be
4367 prepared”). The exact nature of the real events and changes in an implementation that are
4368 modelled by these events depends on the position of the Superior or Inferior within the
4369 business transaction and on features of the implementation (e.g. making of a persistent record
4370 of the decision means that the information will survive at least some failures that otherwise
4371 lose state information, but the level of survival depends on the purpose of the
4372 implementation). [Table 3](#)[Table 2](#)[Table 2](#)[Table 2](#)[Table 2](#)[Table 2](#) and [Table 4](#)[Table 3](#)[Table](#)
4373 [3](#)[Table 3](#)[Table 3](#)[Table 3](#) define the decision events.

4374
4375 ~~In some cases, an implementation may not need to make an active change to have a persistent~~
4376 ~~record of a decision, provided that the implementation will restore itself to the appropriate~~
4377 ~~state on recovery. For example, an (inferior) implementation that “decided to be prepared”,~~
4378 ~~and recorded a timeout (to cancel) in the persistent information for that decision (signalled via~~
4379 ~~the appropriate qualifier on PREPARED), could treat the presence of an expired record as a~~
4380 ~~record of “decide to cancel autonomously”, provided it always updated such a record as part~~
4381 ~~of the “apply ordered confirmation” decision event.~~

4382
4383 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
4384 PREPARE indicates that the application exchange (to associate operations with the Inferior)
4385 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
4386 state corresponding to an incomplete application exchange. However, implementations are
4387 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
4388 that experiences failure after sending PREPARE may, on recovery, have no information
4389 about the transaction, in which case it is considered to be in the completed state (Z), which
4390 will imply the cancellation of the Inferior and its associated operations.

4391
4392 Where a Superior is [an Intermediate \(i.e. is itself an Inferior \(to another Superior entity\), in a](#)
4393 [transaction hierarchy](#) tree, its “decide to confirm” and “decide to cancel” decisions will in fact
4394 be the receipt of a CONFIRM or CANCEL instruction from its own Superior, without
4395 necessary change of local persistent information (which would combine both superior and
4396 inferior information, pointing both up and down the tree).

4397
4398

4399 Disruptions – failure events

4400
4401 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or
4402 may) cause a change of state. The disruption events in the state tables model different extents
4403 of loss of state information. An implementation is **not** required to exhibit all the possible

4404 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
4405 possible disruption. [The different levels of disruption describe legitimate states for the](#)
4406 [endpoint to be in after it has been restored to normal functioning. The absence of a destination](#)
4407 [state for the disruption events means that such a transition is not legitimate – thus, for](#)
4408 [example, an Inferior that has decided to be prepared will always recover to the same state, by](#)
4409 [virtue of the information persisted in the “decide to be prepared” event.](#)

4410
4411 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,
4412 which involves possible interruption of service and loss of messages in transit, but no change
4413 of state (either because no state information was lost, or because recovery from persistent
4414 information restores the implementation to the same state). The “disruption 0” event would
4415 typically be an appropriate abstraction for a communication failure.

4416 **Invalid cells and assumptions of the communication mechanism**

4417
4418
4419 The empty cells in state table represent events that cannot happen. For events corresponding
4420 to sending a message or any of the decision events, this prohibition is absolute – e.g. a
4421 conformant implementation in the Superior active state “B1” will not send CONFIRM. For
4422 events corresponding to receiving a message, the interpretation depends on the properties of
4423 the underlying communications mechanism.

4424
4425 For all communication mechanisms, it is assumed that

- 4426 a) the two directions of the Superior:Inferior communication are not synchronised –
- 4427 that is messages travelling in opposite directions can cross each other to any
- 4428 degree; any number of messages may be in transit in either direction; and
- 4429 b) messages may be lost arbitrarily

4430
4431 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered
4432 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a
4433 state where the corresponding cell is empty indicates that the far-side has sent a message out
4434 of order – a FAULT message with the “fault-type” “WrongState” can be returned.

4435
4436 If the communication mechanisms cannot guarantee ordered delivery, then messages received
4437 where the corresponding cell is empty should be ignored. Assuming the far-side is
4438 conformant, these messages can assumed to be “stale” and have been overtaken by messages
4439 sent later but already delivered. (If the far-side is non-conformant, there is a problem
4440 anyway).

4441 **Meaning of state table events**

4442
4443 The tables in this section define the events (rows) in the state tables. [Table 2](#)~~Table 1~~[Table 1](#)~~Table 1~~[Table 1](#) defines the events corresponding to sending or receiving BTP
4445 messages and the disruption events. [Table 3](#)~~Table 2~~[Table 2](#)~~Table 2~~[Table 2](#) describes
4446 the decision events for an Inferior, [Table 4](#)~~Table 3~~[Table 3](#)~~Table 3~~[Table 3](#) those for a
4447 Superior.
4448
4449

4450 The decision events for a Superior, defined in [Table 4](#)~~Table 3~~~~Table 3~~~~Table 3~~~~Table 3~~~~Table 3~~
4451 cannot be specified without reference to other Inferiors to which it is Superior and to its
4452 relation with the application or other entity that (acting ultimately on behalf of the
4453 application) drives it.

4454
4455 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and
4456 which are to be treated as an atomic decision unit with (and thus including) the Inferior on
4457 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior-
4458 type” of “atom”, this will be all Inferiors established with same Superior address and
4459 “superior-identifier” except those from which RESIGN has been received. If the CONTEXT
4460 had “superior-type” of “cohesion”, the “remaining Inferiors” excludes any that it has been
4461 determined will be cancelled, as well as any that have resigned – in other words it includes
4462 only those for which a confirm decision is still possible or has been made. The determination
4463 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some
4464 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this
4465 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.

4466
4467 In order to ensure that the confirmation decision is delivered to all remaining Inferiors,
4468 despite failures, the Superior must persistently record which these Inferiors are (i.e. their
4469 addresses and identifiers). It must also either record that the decision is confirm, or ensure
4470 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it
4471 will be told about it. This latter would apply if the Superior were also BTP Inferior to another
4472 entity which persisted a confirm decision (or recursively deferred it still higher). However,
4473 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
4474 behaviour of asking another entity to make (and persist) the confirm decision is termed
4475 "offering confirmation" - the Superior offers the possible confirmation of itself, and its
4476 remaining Inferiors to some other entity. If that entity (or something higher up) then does
4477 make and persist a confirm decision, the Superior is "instructed to confirm" (which is
4478 equivalent BTP CONFIRM).

4479
4480 The application, or an entity acting indirectly on behalf of the application, may request a
4481 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
4482 more operations associated with the Inferior. Following a request to prepare all remaining
4483 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
4484 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
4485 application.)

4486
4487 The application, or an entity acting indirectly on behalf of the application, may also request
4488 confirmation. This means the Superior is to attempt to make and persist a confirm decision
4489 itself, rather than offer confirmation.

4490
4491
4492

Table 21: send, receive and disruption events

Event name	Meaning
send/receive ENROL/rsp-req	send/receive ENROL with response-requested = true

Event name	Meaning
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
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

4493

4494

Table 32 : 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

Event name	Meaning
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;
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)

4495

4496

Table 43: 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;

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

4497

4498

4499

4500

4501

4502

4503

4504

4505

4506

4507

4508

4509

4510

4511

4512

4513

4514

4515

4516

4517

4518

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.

4519 The degree of persistence of the recording of a hazard (problem) at an Inferior and recording
4520 of a detected contradiction at a Superior may be different from that applying to the persistent
4521 prepared and confirm information. Implementations and configuration may choose to pass
4522 hazard and contradiction information via management mechanisms rather than through BTP.
4523 Such passing of information to a management mechanism could be treated as “record
4524 problem” or “record contradiction”.
4525

Table 54: 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 65 : 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

State	summary
x2	completed, presuming abort after prepared/cancel
y1	completed, queried
y2	completed, default cancel, a message received
z	completed
z1	completed with default cancel

4529

4530

4530

Superior state table

4531

Table 76: Superior state table – normal forward progression

	I 1	A 1	B 1	B 2	C 1	D 1	E 1	E 2	F 1	F 2
recei ve ENROL/rsp-req	A1	A1	B2	B2		D1				
recei ve ENROL/no-rsp-req	B1		B1	B1		D1				
recei ve RESI GN/rsp-req	Y1		C1	C1	C1	C1				
recei ve RESI GN/no-rsp-req	Z		Z	Z	Z	Z				
recei ve PREPARED	Y1		E1	E1		E1	E1		F1	
recei ve PREPARED/cancel	Y1		E2	E2		E2		E2	F1	
recei ve CONFIR MED/auto	Q1		H1	H1		H1	H1		F1	
recei ve CONFIR MED/response									F2	F2
recei ve CANCELLED	Y1		Z	Z		Z	J1	J1	K1	
recei ve HAZARD	P1	P1	P1	P1		P1	P1	P1	P3	
recei ve INF_STATE/acti ve/y	Y1	A1	B1	B2		D1				
recei ve INF_STATE/acti ve			B1	B2		D1				
recei ve INF_STATE/unknown			Z	Z	Z	Z				
send ENROLLED		B1		B1						
send RESI GNED					Z					
send PREPARE						D1	E1	E2		
send CONFIR M_ONE_PHASE									F1	
send CONFIR M										
send CANCEL										
send CONTRADI CTI ON										
send SUP_STATE/acti ve/y			B1							
send SUP_STATE/acti ve			B1							
send SUP_STATE/prepared-rcvd/y							E1	E2		
send SUP_STATE/prepared-rcvd							E1	E2		
send SUP_STATE/unknown										
deci de to confi rm one-phase			S1	S1			S1	S1		
deci de to prepare			D1	D1						
deci de to confi rm							F1	F1		
deci de to cancel			G1	G1		G1	G1	Z		
remove persi stent i nformati on record contradi cti on										Z
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										

4532

4533

Table 87: 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		

4534

4535

Table 98: 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 109: 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		

4538

4539

4539

Inferior state table

4540

Table 1110: Inferior state table – normal forward progression

	i 1	a 1	b 1	c 1	d 1	e 1	e 2	f 1	f 2
send ENROL/rsp-req	a 1	a 1							
send ENROL/no-rsp-req	b 1		b 1						
send RESI GN/rsp-req				c 1					
send RESI GN/no-rsp-req				z					
send PREPARED						e 1			
send PREPARED/cancel							e 2		
send CONFIR MED/auto									
send CONFIR MED/response									
send CANCELLED			z		z				
send HAZARD									
send INF_STATE/active/y		a 1	b 1		d 1				
send INF_STATE/active			b 1		d 1				
send INF_STATE/unknown									
receive ENROLLED		b 1	b 1	c 1		e 1	e 2		
receive RESI GNED				z					
receive PREPARE		d 1	d 1	c 1	d 1	e 1	e 2		
receive CONFIR M_ONE_PHASE		s 2	s 2	z		s 1	s 1		
receive CONFIR M						f 1	f 2	f 1	f 2
receive CANCEL		n 1	n 1	z	n 1	g 1	g 2		
receive CONTRADI CTI ON									
receive SUP_STATE/active/y		b 1	b 1	c 1		e 1	e 2		
receive SUP_STATE/active		b 1	b 1	c 1		e 1	e 2		
receive SUP_STATE/prepared-rcvd/y						e 1	e 2		
receive SUP_STATE/prepared-rcvd						e 1	e 2		
receive SUP_STATE/unknown		z	z	z	z	x 1	x 2		
decide to resi gn			c 1		c 1				
decide to be prepared			e 1		e 1				
decide to be prepared/cancel			e 2		e 2				
decide to confi rm autonomously						h 1			
decide to cancel autonomously						j 1	z 1		
apply ordered confi rmation								m 1	m 1
remove persi stent i nformati on									
detect probl em		p 1	p 1		p 1	p 2	p 2	p 2	p 2
detect and record probl em									
di srupti on I		z	z	z	z			e 1	e 2
di srupti on II					b 1				
di srupti on III									

4541

4542

4542

Table 1211: 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	

4543

4544

Table 1312: Inferior state table – confirm, cancel ordered and hazard recording

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

4546

Table 1413: Inferior state table – request confirm states

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

4547

4548

4548

Table 1514: Inferior state table – completed states (including presume-abort and queried)

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

4549

4550

4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595

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.

4596 ~~After recovery from node failure, the implementation behaves much as if a communication~~
4597 ~~failure had occurred.~~
4598

4599 **Persistent information**

4600 ~~BTP requires that some decision events are persisted—that information recording an~~
4601 ~~Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s~~
4602 ~~autonomous decision survive failure. Making the first two decisions persistent ensures that a~~
4603 ~~consistent decision can be reached for the business transaction and that it is delivered to all~~
4604 ~~involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to~~
4605 ~~ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the~~
4606 ~~contradiction will be reported to the Superior, despite failures.~~

4609 ~~BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the~~
4610 ~~active state (unlike many transaction protocols, where a communication or endpoint failure in~~
4611 ~~active state would invariably cause rollback of the transaction). Recovery in the active state~~
4612 ~~may require that the application exchange is resynchronised as well—BTP does not directly~~
4613 ~~support this, but does allow continuation of the business transaction as such. In the state~~
4614 ~~tables, from some states, there are several levels of disruption, distinguished by which state~~
4615 ~~the implementation transits to—this represents the survival of different extents of state~~
4616 ~~information over failure and recovery. The different levels of disruption describe legitimate~~
4617 ~~states for the endpoint to be in after it has recovered—**they do not require that all**~~
4618 ~~**implementations are able to exhibit the appropriate partial loss of state information.**~~
4619 ~~The absence of a destination state for the disruption events means that such a transition is not~~
4620 ~~legitimate—thus, for example, an Inferior that has decided to be prepared will always recover~~
4621 ~~to the same state, by virtue of the information persisted in the “decide to be prepared” event.~~

4622
4623 ~~Apart from the (optional) recovery in active state, BTP follows the well known presume-~~
4624 ~~abort model—it is only required that information be persisted when decisions are made (and~~
4625 ~~not, e.g. on enrolment). This means that on recovery, one side may have persistent~~
4626 ~~information but the other does not. This occurs when an Inferior has decided to be prepared~~
4627 ~~but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the~~
4628 ~~Superior did confirm, and the Inferior applied the confirm, removed its persistent information~~
4629 ~~but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it~~
4630 ~~still had the persistent information when the failure occurred).~~

4631 The BTP recovery mechanisms require that information is persisted by the BTP actors that
4632 perform the Superior and Inferior roles. To ensure consistent application of the outcome,
4633 despite failures, the Inferior must persist some state information at the point of becoming
4634 prepared, and the Superior at the point of making a confirm decision. If the Superior is a Sub-
4635 coordinator or Sub-composer, it must persist information when, as an Inferior it becomes
4636 prepared. The minimum information to be persisted is the identifiers and addresses of the
4637 peer Inferiors and Superior – the fact of the persistence being itself an indication of the
4638 preparedness or confirm decision. However, BTP allows recovery of a Superior:Inferior
4639 relationship to occur in other cases – during the active phase, and before a confirm decision
4640 has been made. Thus, in general, the BTP actors will need to persist the current state of the
4641 relationships.

4642 Since BTP messages may carry application-specified qualifiers, which may need to be re-sent
4643 in the case of failure (because the first attempt got lost). BTP actors should be prepared to
4644 persist such qualifiers as well.

4645
4646
4647 A Participant will normally also need to persist some information concerning the application
4648 work whose final or counter effect it is responsible for. The nature of this information is not
4649 considered further in this specification.

4650
4651 Information to be persisted for an Inferior's "decision to be prepared" must be sufficient to
4652 re-establish communication with the Superior, to apply a confirm decision and to apply a
4653 cancel decision. It will thus need to include

4654 ~~—Inferior identity (this may be an index used to locate the information)~~
4655 “sSuperior_address_address”(as on CONTEXT as updated by REDIRECT)
4656 “superior-identifier” (as on CONTEXT)
4657 “default-is-cancel” value (as on PREPARED)

4658
4659 ~~The information needed to apply confirm/cancel decisions will depend on the application and~~
4660 ~~the associated operations. It may also normally be necessary to persist any qualifiers that~~
4661 ~~were sent with the PREPARED message or application messages sent with the PREPARED,~~
4662 ~~since the PREPARED message will be repeated if a failure occurs.~~

4663
4664 A Superior must record corresponding information to allow it to re-establish communication
4665 with the Inferior. Thus, for each Inferior:

4666 “iInferior_address” (as on ENROL, as updated by REDIRECT)
4667 “inferior-identifier” (as on ENROL)

4668
4669 In order to recover their own function, both Superior and Inferior will need to persist their
4670 own Identifier (“superior-identifier” and “inferior-identifier”) and, depending on the
4671 implementation, may need to persist their original “superior-address” or “inferior-address”.

4672
4673 ~~A Superior that is the Decider for the business transaction need only persist this information~~
4674 ~~if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A~~
4675 ~~Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as~~
4676 ~~atom in a cohesion, sub-coordinator or sub-composer) must persist this information as~~
4677 ~~Superior (to this Inferior) as part of the persistent information of its decision to be prepared~~
4678 ~~(as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and~~
4679 ~~if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If~~
4680 ~~CONFIRM is received, the persistent information may be changed to show the confirm~~
4681 ~~decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.~~
4682 ~~If the persistent information is left unchanged and there is a node failure, on recovery the~~
4683 ~~entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision~~
4684 ~~(using the recovery exchanges to its Superior) before propagating it to its Inferior(s).~~

4685
4686 ~~After failure, an implementation may not be able to restore an endpoint to the appropriate~~
4687 ~~state immediately—in particular, the necessary persistent information may be inaccessible,~~
4688 ~~although the implementation can respond to received BTP messages. In such a case, a~~

4689 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a “response
4690 requested” value “false”) with SUPERIOR_STATE/inaccessible and an Inferior to any BTP
4691 message except SUPERIOR_STATE/* with “INFERIOR_STATE/inaccessible. Receipt of
4692 the *_STATE/inaccessible messages has no effect on the endpoint state.

4693

4694 **Redirection**

4695

4696 ~~As described above, BTP uses the presume-abort model for recovery. A corollary of this is~~
4697 ~~that there are cases where one side will attempt to re-establish communication when there is~~
4698 ~~no persistent information for the relationship at the far end. In such cases, it is important the~~
4699 ~~side that is attempting recovery can distinguish between unsuccessful attempts to connect to~~
4700 ~~the holder of the persistent information and when the information no longer exists. If the peer~~
4701 ~~information does not exist, this side can draw conclusions and complete appropriately; if they~~
4702 ~~merely fail to get through they are stuck in attempting recovery.~~

4703

4704 ~~Two mechanisms are provided to make it possible that even when one side of a~~
4705 ~~Superior:Inferior relationship has completed, that a message can eventually get through to~~
4706 ~~something that can definitively report the status, distinguishing this case from a temporary~~
4707 ~~inability to access the state of a continuing transaction element. The mechanisms are:~~

4708 ~~oAddress fields which provide a “callback address” can be a set of addresses,~~
4709 ~~which are alternatives one of which is chosen as the “target address” for the~~
4710 ~~future message. If the sender of that message finds the address does not work,~~
4711 ~~it can try a different alternative.~~

4712 ~~oThe REDIRECT message can be used to inform the peer that an address~~
4713 ~~previously given is no longer valid and to supply a replacement address (or~~
4714 ~~set of addresses). REDIRECT can be issued either as a response to receipt of~~
4715 ~~a message or spontaneously.~~

4716

4717 ~~The two mechanisms can be used in combination, with one or more of the original set of~~
4718 ~~addresses just being a redirector, which does not itself ever have direct access to the state~~
4719 ~~information for the transaction, but will respond to any message with an appropriate~~
4720 ~~REDIRECT.~~

4721

4722 ~~An alternative implementation approach is to have a single addressable entity that uses the~~
4723 ~~same address for all transactions, distinguishing them by identifier, and which always~~
4724 ~~recovers to use the same address. Such an implementation would not need to supply~~
4725 ~~“backup” addresses (and would only use REDIRECT if it was being permanently migrated).~~

4726

4727 **Terminator:Decider failures**

4728

4729 ~~BTP does not provide facilities or impose requirements on the recovery of~~
4730 ~~Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator~~
4731 ~~may survive failures (by retaining knowledge of the Decider’s address and identifier), but this~~
4732 ~~is an implementation option. Although a Decider (if it decides to confirm) will persist~~
4733 ~~information about the confirm decision, it is not required, after failure, to remain accessible~~

4734 ~~using the inferior address it offered to the Terminator. Any such recovery is an~~
4735 ~~implementation option.~~
4736
4737 ~~A Decider's address (as returned on BEGUN) may be a set of addresses, allowing a failed~~
4738 ~~Decider to be recovered at a different address.~~
4739
4740 ~~A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and~~
4741 ~~thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for~~
4742 ~~a CONFIRM_TRANSACTION that will never arrive, the standard qualifier "Transaction~~
4743 ~~timelimit" can be used (by the Initiator) to inform the Decider when it can assume the~~
4744 ~~Terminator will not issue CONFIRM_TRANSACTION and so it (the Decider) should initiate~~
4745 ~~cancellation.~~
4746

4747 XML representation of Message Set

4748
4749 This section describes the syntax for BTP messages in XML. These XML messages represent
4750 a midpoint between the abstract messages and what actually gets sent on the wire.
4751

4752 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)
4753 [3121](#)

4754
4755 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:1.0:core
4756

4757 In addition to an XML schema, this specification uses an informal syntax to describe the
4758 structure of the BTP messages. The syntax appears as an XML instance, but the values
4759 contain data types instead of values. The following symbols are appended to some of the
4760 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
4761 these symbols corresponds to "one and only one."
4762

4763 [The Delivery parameters are shown in the XML with a darker background.](#)
4764

4765 Addresses

4766
4767 As described in the "Abstract Message and Associated Contracts – Addresses" section, a BTP
4768 address comprises three parts, and for a "target-address" only the "additional information"
4769 field is inside the BTP messages. For all BTP messages whose abstract form includes a
4770 "target-address" parameter, the corresponding XML representation includes a "target-
4771 additional-information" element. This element may be omitted if it would be empty.
4772

4773 For other addresses, all three fields are represent, as in:

```
4774  
4775 <btm:some-address>  
4776   <btm:binding-name>...carrier binding URI...</btm:binding-name>  
4777   <btm:binding-address>...carrier specific  
4778   address...</btm:binding-address>
```

4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816

4817
4818
4819

4820
4821
4822
4823
4824

```
<ctp:additional-information>...optional additional addressing  
information...</ctp:additional-information> ?  
</ctp:some-address>
```

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

Qualifiers

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

Examples:

```
<ctpq:inferior-timeout  
  xmlns:ctpq="urn:oasis:names:tc:BTP:1.0:qualifiers"  
  xmlns:ctp="urn:oasis:names:tc:BTP:1.0:core"  
  ctp:must-be-understood="false"  
  ctp:to-be-propagated="false">1800</ctpq:inferior-timeout>  
  
<auth:username  
  xmlns:auth="http://www.example.com/ns/auth"  
  xmlns:ctp="urn:oasis:names:tc:BTP:1.0:core"  
  ctp:must-be-understood="true"  
  ctp:to-be-propagated="true">jtauber</auth:username>
```

Attributes must-be-understood **has default value "true"** and to-be-propagated has default value "false".

Identifiers

Identifiers shall be URIs "

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

Message References

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

4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874

Messages

CONTEXT

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

CONTEXT_REPLY

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

REQUEST_STATUS

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

STATUS

```

4875 <ctp:status id?>
4876   <ctp:responders-identifier>...URI...</ctp:responders-identifier>
4877   <ctp:status-value>created|enrolling|active|resigning|
4878     resigned|preparing|prepared|
4879     confirming|confirmed|cancelling|cancelled|
4880     cancel-contradiction|confirm-contradiction|
4881     hazard|contradicted|unknown|inaccessible</ctp:status-
4882 value>
4883   <ctp:qualifiers> ?
4884     ...qualifiers...
4885 </ctp:qualifiers>
4886 <ctp:target-additional-information> ?
4887   ...additional address information...
4888 </ctp:target-additional-information>
4889 </ctp:status>

```

FAULT

```

4893 <ctp:fault id?>
4894   <ctp:superior-identifier>...URI...</ctp:superior-identifier> ?
4895   <ctp:inferior-identifier>...URI...</ctp:inferior-identifier> ?
4896   <ctp:fault-type>...fault type name...</ctp:fault-type>
4897   <ctp:fault-data>...fault data...</ctp:fault-data> ?
4898   <ctp:fault-text>...string data ...</ctp:fault-data> ?
4899   <ctp:qualifiers> ?
4900     ...qualifiers...
4901 </ctp:qualifiers>
4902 <ctp:target-additional-information> ?
4903   ...additional address information...
4904 </ctp:target-additional-information>
4905 </ctp:fault>

```

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

- 4910 o communication-failure
- 4911 o duplicate-inferior
- 4912 o general
- 4913 o invalid-decider
- 4914 o invalid-inferior
- 4915 o invalid-superior
- 4916 o status-refused
- 4917 o invalid-terminator
- 4918 o unknown-parameter
- 4919 o unknown-transaction
- 4920 o unsupported-qualifier
- 4921 o wrong-state
- 4922 o redirect

4924 Revisions of this specification may add other fault type names, which shall be simple strings
4925 of letters, numbers and hyphens. If other specifications define fault type names to be used
4926 with BTP, the names shall be URIs.

4927

4928 Fault data can take on various forms:

4929

4930

4931 Identifier:

4932

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

4934

4935

4936 Inferior Identity:

4937

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

4944

4945

ENROL

4946

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

4964

4965

4966

ENROLLED

4967

```
<btp: enrolled id?>  
<btp: sender-address> ?  
  ...address...  
</btp: sender-address>  
<btp: inferior-identifier>...URI...</btp: inferior-identifier>  
<btp: qualifiers> ?
```

4973

```
4974     ...qualifiers...
4975     </btp:qualifiers>
4976     <btp:target-additional-information> ?
4977     ...additional address information...
4978     </btp:target-additional-information>
4979 </btp:enrolled>
```

RESIGN

```
4983
4984 <btp:resign id?>
4985   <btp:superior-identifier>...URI...</btp:superior-identifier>
4986   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
4987   <btp:response-requested>true|false</btp:response-requested>
4988   <btp:qualifiers> ?
4989     ...qualifiers...
4990   </btp:qualifiers>
4991   <btp:target-additional-information> ?
4992     ...additional address information...
4993   </btp:target-additional-information>
4994   <btp:sender-address> ?
4995     ...address...
4996   </btp:sender-address>
4997 </btp:resign>
```

RESIGNED

```
5000
5001
5002 <btp:resigned id?>
5003   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
5004   <btp:qualifiers> ?
5005     ...qualifiers...
5006   </btp:qualifiers>
5007   <btp:target-additional-information> ?
5008     ...additional address information...
5009   </btp:target-additional-information>
5010   <btp:sender-address> ?
5011     ...address...
5012   </btp:sender-address>
5013 </btp:resigned>
```

PREPARE

```
5016
5017
5018 <btp:prepare id?>
5019   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
5020   <btp:qualifiers> ?
5021     ...qualifiers...
5022   </btp:qualifiers>
5023   <btp:target-additional-information> ?
```

```
5024     ...additional address information...
5025     </btp:target-additional-information>
5026     <btp:sender-address> ?
5027     ...address...
5028     </btp:sender-address>
5029 </btp:prepare>
```

5032 PREPARED

```
5033
5034 <btp:prepared id?>
5035   <btp:superior-identifier>...URI...</btp:superior-identifier>
5036   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
5037   <btp:default-is-cancel>true|false</btp:default-is-cancel>
5038   <btp:qualifiers> ?
5039   ...qualifiers...
5040 </btp:qualifiers>
5041 <btp:target-additional-information> ?
5042   ...additional address information...
5043 </btp:target-additional-information>
5044 <btp:sender-address> ?
5045   ...address...
5046 </btp:sender-address>
5047 </btp:prepared>
```

5050 CONFIRM

```
5051
5052 <btp:confirm id?>
5053   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
5054   <btp:qualifiers> ?
5055   ...qualifiers...
5056 </btp:qualifiers>
5057 <btp:target-additional-information> ?
5058   ...additional address information...
5059 </btp:target-additional-information>
5060 <btp:sender-address> ?
5061   ...address...
5062 </btp:sender-address>
5063 </btp:confirm>
```

5066 CONFIRMED

```
5067
5068 <btp:confirmed id?>
5069   <btp:superior-identifier>...URI...</btp:superior-identifier>
5070   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
5071   <btp:confirmed-received>true|false</btp:confirmed-received>
5072   <btp:qualifiers> ?
5073   ...qualifiers...
```

```
5074 </btp:qualifiers>
5075 <btp:target-additional-information> ?
5076   ...additional address information...
5077 </btp:target-additional-information>
5078 <btp:sender-address> ?
5079   ...address...
5080 </btp:sender-address>
5081 </btp:confirmed>
```

CANCEL

```
5082
5083
5084
5085
5086 <btp:cancel id?>
5087   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
5088   <btp:reply-address> ?
5089     ...address...
5090 </btp:reply-address>
5091   <btp:qualifiers> ?
5092     ...qualifiers...
5093 </btp:qualifiers>
5094   <btp:target-additional-information> ?
5095     ...additional address information...
5096 </btp:target-additional-information>
5097   <btp:sender-address> ?
5098     ...address...
5099 </btp:sender-address>
5100 </btp:cancel>
```

5101
5102

CANCELLED

```
5103
5104
5105 <btp:cancelled id?>
5106   <btp:superior-identifier>...URI...</btp:superior-identifier>
5107
5108   <btp:inferior-identifier>...URI...</btp:inferior-identifier> ?
5109   <btp:qualifiers> ?
5110     ...qualifiers...
5111 </btp:qualifiers>
5112   <btp:target-additional-information> ?
5113     ...additional address information...
5114 </btp:target-additional-information>
5115   <btp:sender-address> ?
5116     ...address...
5117 </btp:sender-address>
5118 </btp:cancelled>
```

5119
5120

CONFIRM_ONE_PHASE

```
5121
5122
5123 <btp:confirm-one-phase id?>
```

```
5124 <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>
5125 <btpt:report-hazard>true|false</btpt:report-hazard>
5126 <btpt:qualifiers> ?
5127   ...qualifiers...
5128 </btpt:qualifiers>
5129 <btpt:target-additional-information> ?
5130   ...additional address information...
5131 </btpt:target-additional-information>
5132 <btpt:sender-address> ?
5133   ...address...
5134 </btpt:sender-address>
5135 </btpt:confirm-one-phase>
```

5136

HAZARD

```
5137
5138
5139 <btpt:hazard id?>
5140   <btpt:superior-identifier>...URI...</btpt:superior-identifier>
5141
5142   <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>
5143   <btpt:level>mixed|possible</btpt:level>
5144   <btpt:qualifiers> ?
5145     ...qualifiers...
5146   </btpt:qualifiers>
5147   <btpt:target-additional-information> ?
5148     ...additional address information...
5149   </btpt:target-additional-information>
5150   <btpt:sender-address> ?
5151     ...address...
5152   </btpt:sender-address>
5153 </btpt:hazard>
```

5154

5155

CONTRADICTION

```
5156
5157
5158 <btpt:contradiction id?>
5159   <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>
5160   <btpt:qualifiers> ?
5161     ...qualifiers...
5162   </btpt:qualifiers>
5163   <btpt:target-additional-information> ?
5164     ...additional address information...
5165   </btpt:target-additional-information>
5166   <btpt:sender-address> ?
5167     ...address...
5168   </btpt:sender-address>
5169 </btpt:contradiction>
```

5170

5171

SUPERIOR_STATE

5172

5173

```
5174 <btp:superior-state id?>
5175   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
5176   <btp:status>active|prepared-
5177   received|inaccessible|unknown</btp:status>
5178   <btp:response-requested>true|false</btp:response-requested>
5179   <btp:qualifiers> ?
5180     ...qualifiers...
5181   </btp:qualifiers>
5182   <btp:target-additional-information> ?
5183     ...additional address information...
5184   </btp:target-additional-information>
5185   <btp:sender-address> ?
5186     ...address...
5187   </btp:sender-address>
5188 </btp:superior-state>
```

5189
5190

5191 INFERIOR_STATE

```
5192
5193 <btp:inferior-state id?>
5194   <btp:superior-identifier>...URI...</btp:superior-identifier>
5195
5196   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
5197   <btp:status>active|inaccessible|unknown</btp:status>
5198   <btp:response-requested>true|false</btp:response-requested>
5199   <btp:qualifiers> ?
5200     ...qualifiers...
5201   </btp:qualifiers>
5202   <btp:target-additional-information> ?
5203     ...additional address information...
5204   </btp:target-additional-information>
5205   <btp:sender-address> ?
5206     ...address...
5207   </btp:sender-address>
5208 </btp:inferior-state>
```

5209
5210

5211 REDIRECT

```
5212
5213 <btp:redirect id?>
5214   <btp:superior-identifier>...URI...</btp:superior-identifier> ?
5215   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
5216   <btp:old-address> +
5217     ...address...
5218   </btp:old-address>
5219   <btp:new-address> +
5220     ...address...
5221   </btp:new-address>
5222   <btp:qualifiers> ?
5223     ...qualifiers...
5224   </btp:qualifiers>
```

```
5225 <btptarget-additional-information> ?
5226 ...additional address information...
5227 </btptarget-additional-information>
5228 </btptredirect>
```

BEGIN

```
5231
5232 <btptbegin id?>
5233 <btpttransaction-type>cohesion|atom</btpttransaction-type>
5234 <btptqualifiers> ?
5235 ...qualifiers...
5236 </btptqualifiers>
5237 <btpttarget-additional-information> ?
5238 ...additional address information...
5239 </btpttarget-additional-information>
5240 <btptreply-address> ?
5241 ...address...
5242 </btptreply-address>
5243 </btptbegin>
```

BEGUN

```
5247
5248 <btptbegun id?>
5249 <btptdecider-address> *
5250 ...address...
5251 </btptdecider-address>
5252 <btptinferior-address> *
5253 ...address...
5254 </btptinferior-address>
5255 <btpttransaction-identifier>...URI...</btpttransaction-
5256 identifier>
5257 <btptqualifiers> ?
5258 ...qualifiers...
5259 </btptqualifiers>
5260 <btpttarget-additional-information> ?
5261 ...additional address information...
5262 </btpttarget-additional-information>
5263 </btptbegun>
```

PREPARE_INFERIORS

```
5267
5268 <btptprepare-inferiors id?>
5269 <btpttransaction-identifier>...URI...</btpttransaction-
5270 identifier>
5271 <btptinferiors-list> ?
5272 <btptinferior-handle>...URI...</btptinferior-handle> +
5273 </btptinferiors-list>
5274 <btptqualifiers> ?
```

```
5275     ...qualifiers...
5276     </btp:qualifiers>
5277     <btp:target-additional-information> ?
5278     ...additional address information...
5279     </btp:target-additional-information>
5280     <btp:reply-address> ?
5281     ...address...
5282     </btp:reply-address>
5283 </btp:prepare-inferiors>
```

5286 CONFIRM_TRANSACTION

```
5287
5288     <btp:confirm-transaction id?>
5289     <btp:transaction-identifier>...URI...</btp:transaction-
5290     identifier>
5291     <btp:inferiors-list> ?
5292     <btp:inferior-handle>...URI...</btp:inferior-handle> +
5293     </btp:inferiors-list>
5294     <btp:report-hazard>true|false</btp:report-hazard>
5295     <btp:qualifiers> ?
5296     ...qualifiers...
5297     </btp:qualifiers>
5298     <btp:target-additional-information> ?
5299     ...additional address information...
5300     </btp:target-additional-information>
5301     <btp:reply-address> ?
5302     ...address...
5303     </btp:reply-address>
5304 </btp:confirm_transaction>
```

5305

5306

5307 TRANSACTION_CONFIRMED

5308

```
5309     <btp:transaction-confirmed id?>
5310     <btp:transaction-identifier>...URI...</btp:transaction-
5311     identifier>
5312     <btp:qualifiers> ?
5313     ...qualifiers...
5314     </btp:qualifiers>
5315     <btp:target-additional-information> ?
5316     ...additional address information...
5317     </btp:target-additional-information>
5318 </btp:transaction-confirmed>
```

5319

5320

5321 CANCEL_TRANSACTION

5322

```
5323     <btp:cancel-transaction id?>
```

```
5324 <btp:transaction-identifier>...URI...</btp:transaction-
5325 identifier>
5326 <btp:report-hazard>true|false</btp:report-hazard>
5327 <btp:qualifiers> ?
5328 ...qualifiers...
5329 </btp:qualifiers>
5330 <btp:target-additional-information> ?
5331 ...additional address information...
5332 </btp:target-additional-information>
5333 <btp:reply-address> ?
5334 ...address...
5335 </btp:reply-address>
5336 </btp:cancel-transaction>
```

5337 CANCEL_INFERIORS

```
5338
5339
5340 <btp:cancel-inferiors id?>
5341 <btp:transaction-identifier>...URI...</btp:transaction-
5342 identifier> ?
5343 <btp:inferiors-list>
5344 <btp:inferior-handle>...URI...</btp:inferior-handle> +
5345 </btp:inferiors-list>
5346 <btp:qualifiers> ?
5347 ...qualifiers...
5348 </btp:qualifiers>
5349 <btp:target-additional-information> ?
5350 ...additional address information...
5351 </btp:target-additional-information>
5352 <btp:reply-address> ?
5353 ...address...
5354 </btp:reply-address>
5355 </btp:cancel-inferiors>
```

5356 TRANSACTION_CANCELLED

```
5357
5358
5359 <btp:transaction-cancelled id?>
5360 <btp:transaction-identifier>...URI...</btp:transaction-
5361 identifier>
5362 <btp:qualifiers> ?
5363 ...qualifiers...
5364 </btp:qualifiers>
5365 <btp:target-additional-information> ?
5366 ...additional address information...
5367 </btp:target-additional-information>
5368 </btp:transaction-cancelled>
```

5370 REQUEST_INFERIOR_STATUSES

5371
5372
5373

```

5374 <btpr:request-inferior-statuses id?>
5375 <btpr:target-identifier>...URI...</btpr:target-identifier>
5376 <btpr:inferiors-list> ?
5377 <btpr:inferior-handle>...URI...</btpr:inferior-handle> +
5378 </btpr:inferiors-list>
5379 <btpr:qualifiers> ?
5380 ...qualifiers...
5381 </btpr:qualifiers>
5382 <btpr:target-additional-information> ?
5383 ...additional address information...
5384 </btpr:target-additional-information>
5385 <btpr:reply-address> ?
5386 ...address...
5387 </btpr:reply-address>
5388 </btpr:request-inferior-statuses>

```

INFERIOR_STATUSES

```

5391
5392
5393 <btpr:inferior-statuses id?>
5394 <btpr:responders-identifier>...URI...</btpr:responders-identifier>
5395 <btpr:status-list>
5396 <btpr:status-item> +
5397 <btpr:inferior-handle>...URI...</btpr:inferior-handle>
5398 <btpr:status>active|resigned|preparing|prepared|
5399 autonomously-confirmed|autonomously-cancelled|
5400 confirming|confirmed|cancelling|cancelled|
5401 cancel-contradiction|confirm-contradiction|
5402 hazard|invalid</btpr:status>
5403 <btpr:qualifiers> ?
5404 ...qualifiers...
5405 </btpr:qualifiers>
5406 </btpr:status-item>
5407 </btpr:status-list>
5408 <btpr:qualifiers> ?
5409 ...qualifiers...
5410 </btpr:qualifiers>
5411 <btpr:target-additional-information> ?
5412 ...additional address information...
5413 </btpr:target-additional-information>
5414 </btpr:inferior-statuses>
5415

```

Standard qualifiers

The informal syntax for these messages assumes the namespace prefix “btpr” is associated with the URI “urn:oasis:names:tc:BTP:1.0:qualifiers”.

Transaction timelimit

```

5421
5422 <btprq:transaction-timelimit>
5423 <btprq:timelimit>

```

```
5424     ...time in seconds...
5425     </btpq:timelimit>
5426 </btpq:transaction-timelimit>
```

Inferior timeout

```
5429 <btpq:inferior-timeout>
5430   <btpq:timeout>
5431     ...time in seconds...
5432   </btpq:timeout>
5433   <btpq:intended-decision>confirm|cancel</btpq:intended-decision>
5434 </btpq:inferior-timeout>
```

Minimum inferior timeout

```
5437 <btpq:minimum-inferior-timeout>
5438   <btpq:minimum-timeout>
5439     ...time in seconds...
5440   </btpq:minimum-timeout>
5441 </btpq:minimum-inferior-timeout>
```

Inferior name

```
5444 <btpq:inferior-name>
5445   <btpq:inferior-name>
5446     ...string...
5447   </btpq:inferior-name>
5448 </btpq:inferior-name>
```

Compounding of Messages

5450
5451
5452 Relating BTP to one another, in a “group” is represented by containing them within the
5453 btp:related-group element, with the related messages as child elements. The processing for
5454 the group is defined in the section “Groups – combinations of related messages”. For example

```
5455  
5456 <btp:related-group>
5457   <btp:context-reply>
5458     ...<completion-status>related</completion-status> ...
5459   </btp:context-reply>
5460   <btp:enrol>...</btp:enrol>
5461   <btp:prepared>...</btp:prepared>
5462 </btp:related-group>
```

5463
5464 If the rules for the group state that the “target-address” of the abstract message is omitted, the
5465 corresponding target-address-information element shall be absent in the message in the
5466 related-group. The carrier protocol binding specifies how a relation between application and
5467 BTP messages is represented.

5468
5469 Bundling (semantically insignificant combination) of BTP messages and related groups is
5470 indicated with the "btp:messages" element, with the bundled messages and related groups as
5471 child elements. For example (confirming one and cancelling another inferiors of a cohesion):
5472

5473
5474
5475
5476
5477
5478
5479

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

5479

XML Schemas

5480

5481

XML schema for BTP messages

5482

5483

5484

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

5485

```
<schema
```

5486

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

5487

```
  targetNamespace="urn:oasis:names:tc:BTP:1.0:core"
```

5488

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

5489

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

5490

5491

5492

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

5493

5494

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

5495

```
    <simpleContent>
```

5496

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

5497

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

5498

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

5499

```
      </extension>
```

5500

```
    </simpleContent>
```

5501

```
  </complexType>
```

5502

5503

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

5504

5505

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

5506

```
    <complexType>
```

5507

```
      <sequence>
```

5508

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

5509

```
      </sequence>
```

5510

```
    </complexType>
```

5511

```
  </element>
```

5512

5513

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

5514

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

5515

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

5516

```
  -->
```

5517

5518

5519

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

5520

5521

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

5522

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

5523

```
  </simpleType>
```

5524

5525

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

5526

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

5527

```
  </simpleType>
```

5528

5529

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

```

5530     <sequence>
5531         <element name="binding-name" type="anyURI"/>
5532         <element name="binding-address" type="string"/>
5533         <element name="additional-information" type="btp:additional-
5534 information" minOccurs="0" />
5535     </sequence>
5536 </complexType>
5537
5538 <simpleType name="superior-type">
5539     <restriction base="string">
5540         <enumeration value="cohesion"/>
5541         <enumeration value="atom"/>
5542     </restriction>
5543 </simpleType>
5544
5545 <simpleType name="transaction-type">
5546     <restriction base="string">
5547         <enumeration value="cohesion"/>
5548         <enumeration value="atom"/>
5549     </restriction>
5550 </simpleType>
5551
5552 <!-- Compounding -->
5553
5554 <element name="messages">
5555     <complexType>
5556         <sequence>
5557             <element ref="btp:message" minOccurs="0"
5558 maxOccurs="unbounded"/>
5559         </sequence>
5560     </complexType>
5561 </element>
5562
5563 <element name="related-group" substitutionGroup="btp:message">
5564     <complexType>
5565         <sequence>
5566             <element ref="btp:message" minOccurs="0"
5567 maxOccurs="unbounded"/>
5568         </sequence>
5569     </complexType>
5570 </element>
5571
5572 <!-- Message set -->
5573
5574 <element name="message" abstract="true" />
5575
5576 <element name="context" substitutionGroup="btp:message">
5577     <complexType>
5578         <sequence>

```

```

5581         <element name="superior-address" type="btp:address"
5582 maxOccurs="unbounded"/>
5583         <element name="superior-identifier" type="btp:identifier"/>
5584         <element name="superior-type" type="btp:superior-type"/>
5585         <element ref="btp:qualifiers" minOccurs="0"/>
5586         <element name="reply-address" type="btp:address"
5587 minOccurs="0"/>
5588     </sequence>
5589     <attribute name="id" type="ID" use="optional"/>
5590 </complexType>
5591 </element>
5592
5593 <element name="context-reply" substitutionGroup="btp:message">
5594 <complexType>
5595 <sequence>
5596 <element name="superior-identifier" type="btp:identifier"/>
5597 <element name="completion-status">
5598 <simpleType>
5599 <restriction base="string">
5600 <enumeration value="completed"/>
5601 <enumeration value="incomplete"/>
5602 <enumeration value="related"/>
5603 <enumeration value="repudiated"/>
5604 </restriction>
5605 </simpleType>
5606 </element>
5607 <element ref="btp:qualifiers" minOccurs="0"/>
5608 <element name="target-additional-information"
5609 type="btp:additional-information" minOccurs="0"/>
5610 </sequence>
5611 <attribute name="id" type="ID"/>
5612 </complexType>
5613 </element>
5614
5615 <element name="request-status" substitutionGroup="btp:message">
5616 <complexType>
5617 <sequence>
5618 <element name="target-identifier" type="btp:identifier"/>
5619 <element ref="btp:qualifiers" minOccurs="0"/>
5620 <element name="target-additional-information"
5621 type="btp:additional-information" minOccurs="0"/>
5622 <element name="reply-address" type="btp:address"
5623 minOccurs="0"/>
5624 </sequence>
5625 <attribute name="id" type="ID"/>
5626 </complexType>
5627 </element>
5628
5629 <element name="status" substitutionGroup="btp:message">
5630 <complexType>
5631 <sequence>

```

```

5632         <element name="responders-identifier"
5633 type="btp:identifier"/>
5634         <element name="status-value">
5635             <simpleType>
5636                 <restriction base="string">
5637                     <enumeration value="created"/>
5638                     <enumeration value="enrolling"/>
5639                     <enumeration value="active"/>
5640                     <enumeration value="resigning"/>
5641                     <enumeration value="resigned"/>
5642                     <enumeration value="preparing"/>
5643                     <enumeration value="prepared"/>
5644                     <enumeration value="confirming"/>
5645                     <enumeration value="confirmed"/>
5646                     <enumeration value="cancelling"/>
5647                     <enumeration value="cancelled"/>
5648                     <enumeration value="cancel-contradiction"/>
5649                     <enumeration value="confirm-contradiction"/>
5650                     <enumeration value="hazard"/>
5651                     <enumeration value="contradicted"/>
5652                     <enumeration value="unknown"/>
5653                     <enumeration value="inaccessible"/>
5654                 </restriction>
5655             </simpleType>
5656         </element>
5657         <element ref="btp:qualifiers" minOccurs="0"/>
5658         <element name="target-additional-information"
5659 type="btp:additional-information" minOccurs="0"/>
5660     </sequence>
5661     <attribute name="id" type="ID"/>
5662 </complexType>
5663 </element>
5664
5665     <element name="fault" substitutionGroup="btp:message">
5666         <complexType>
5667             <sequence>
5668                 <element name="superior-identifier" type="btp:identifier"
5669 minOccurs="0"/>
5670                 <element name="inferior-identifier" type="btp:identifier"
5671 minOccurs="0"/>
5672                 <element name="fault-type">
5673                     <simpleType>
5674                         <restriction base="string">
5675                             <enumeration value="communication-failure"/>
5676                             <enumeration value="duplicate-inferior"/>
5677                             <enumeration value="general"/>
5678                             <enumeration value="invalid-decider"/>
5679                             <enumeration value="invalid-inferior"/>
5680                             <enumeration value="invalid-superior"/>
5681                             <enumeration value="status-refused"/>
5682                             <enumeration value="invalid-terminator"/>
5683                             <enumeration value="unknown-parameter"/>

```

```

5684         <enumeration value="unknown-transaction"/>
5685         <enumeration value="unsupported-qualifier"/>
5686         <enumeration value="wrong-state"/>
5687     </restriction>
5688 </simpleType>
5689 </element>
5690 <element name="fault-data" type="anyType" minOccurs="0"/>
5691 <element ref="btp:qualifiers" minOccurs="0"/>
5692 <element name="target-additional-information"
5693 type="btp:additional-information" minOccurs="0"/>
5694 </sequence>
5695 <attribute name="id" type="ID"/>
5696 </complexType>
5697 </element>
5698
5699 <element name="enrol" substitutionGroup="btp:message">
5700 <complexType>
5701 <sequence>
5702 <element name="superior-identifier" type="btp:identifier"/>
5703 <element name="response-requested" type="boolean"/>
5704 <element name="reply-address" type="btp:address"
5705 minOccurs="0"/>
5706 <element name="inferior-address" type="btp:address"
5707 minOccurs="1" maxOccurs="unbounded"/>
5708 <element name="inferior-identifier" type="btp:identifier"/>
5709 <element ref="btp:qualifiers" minOccurs="0"/>
5710 <element name="target-additional-information"
5711 type="btp:additional-information" minOccurs="0"/>
5712 </sequence>
5713 <attribute name="id" type="ID"/>
5714 </complexType>
5715 </element>
5716
5717
5718 <element name="enrolled" substitutionGroup="btp:message">
5719 <complexType>
5720 <sequence>
5721 <element name="inferior-identifier" type="btp:identifier"/>
5722 <element ref="btp:qualifiers" minOccurs="0"/>
5723 <element name="target-additional-information"
5724 type="btp:additional-information" minOccurs="0"/>
5725 </sequence>
5726 <attribute name="id" type="ID"/>
5727 </complexType>
5728 </element>
5729
5730 <element name="resign" substitutionGroup="btp:message">
5731 <complexType>
5732 <sequence>
5733 <element name="superior-identifier" type="btp:identifier"/>
5734 <element name="inferior-identifier" type="btp:identifier"/>
5735 <element name="response-requested" type="boolean"/>

```

```

5736         <element ref="btp:qualifiers" minOccurs="0"/>
5737         <element name="target-additional-information"
5738 type="btp:additional-information" minOccurs="0"/>
5739     </sequence>
5740     <attribute name="id" type="ID"/>
5741 </complexType>
5742 </element>
5743
5744 <element name="resigned" substitutionGroup="btp:message">
5745 <complexType>
5746 <sequence>
5747     <element name="inferior-identifier" type="btp:identifier"/>
5748     <element ref="btp:qualifiers" minOccurs="0"/>
5749     <element name="target-additional-information"
5750 type="btp:additional-information" minOccurs="0"/>
5751 </sequence>
5752 <attribute name="id" type="ID"/>
5753 </complexType>
5754 </element>
5755
5756 <element name="prepare" substitutionGroup="btp:message">
5757 <complexType>
5758 <sequence>
5759     <element name="inferior-identifier" type="btp:identifier"/>
5760     <element ref="btp:qualifiers" minOccurs="0"/>
5761     <element name="target-additional-information"
5762 type="btp:additional-information" minOccurs="0"/>
5763 </sequence>
5764 <attribute name="id" type="ID"/>
5765 </complexType>
5766 </element>
5767
5768 <element name="prepared" substitutionGroup="btp:message">
5769 <complexType>
5770 <sequence>
5771     <element name="superior-identifier" type="btp:identifier"/>
5772     <element name="inferior-identifier" type="btp:identifier"/>
5773     <element name="default-is-cancel" type="boolean"/>
5774     <element ref="btp:qualifiers" minOccurs="0"/>
5775     <element name="target-additional-information"
5776 type="btp:additional-information" minOccurs="0"/>
5777 </sequence>
5778 <attribute name="id" type="ID"/>
5779 </complexType>
5780 </element>
5781
5782 <element name="confirm" substitutionGroup="btp:message">
5783 <complexType>
5784 <sequence>
5785     <element name="inferior-identifier" type="btp:identifier"/>
5786     <element ref="btp:qualifiers" minOccurs="0"/>

```

```

5787         <element name="target-additional-information"
5788 type="btp:additional-information" minOccurs="0"/>
5789     </sequence>
5790     <attribute name="id" type="ID"/>
5791 </complexType>
5792 </element>
5793
5794 <element name="confirmed" substitutionGroup="btp:message">
5795     <complexType>
5796         <sequence>
5797             <element name="superior-identifier" type="btp:identifier"/>
5798             <element name="inferior-identifier" type="btp:identifier"/>
5799             <element name="confirmed-received" type="boolean"/>
5800             <element ref="btp:qualifiers" minOccurs="0"/>
5801             <element name="target-additional-information"
5802 type="btp:additional-information" minOccurs="0"/>
5803         </sequence>
5804         <attribute name="id" type="ID"/>
5805     </complexType>
5806 </element>
5807
5808 <element name="cancel" substitutionGroup="btp:message">
5809     <complexType>
5810         <sequence>
5811             <element name="inferior-identifier" type="btp:identifier"/>
5812             <element name="reply-address" type="btp:address"
5813 minOccurs="0"/>
5814             <element ref="btp:qualifiers" minOccurs="0"/>
5815             <element name="target-additional-information"
5816 type="btp:additional-information" minOccurs="0"/>
5817         </sequence>
5818         <attribute name="id" type="ID"/>
5819     </complexType>
5820 </element>
5821
5822 <element name="cancelled" substitutionGroup="btp:message">
5823     <complexType>
5824         <sequence>
5825             <element name="superior-identifier" type="btp:identifier"/>
5826             <element name="inferior-identifier" type="btp:identifier"
5827 minOccurs="0"/>
5828             <element ref="btp:qualifiers" minOccurs="0"/>
5829             <element name="target-additional-information"
5830 type="btp:additional-information" minOccurs="0"/>
5831         </sequence>
5832         <attribute name="id" type="ID"/>
5833     </complexType>
5834 </element>
5835
5836 <element name="confirm-one-phase" substitutionGroup="btp:message">
5837     <complexType>
5838         <sequence>

```

```

5839         <element name="inferior-identifier" type="btp:identifier"/>
5840         <element name="report-hazard" type="boolean"/>
5841         <element ref="btp:qualifiers" minOccurs="0"/>
5842         <element name="target-additional-information"
5843 type="btp:additional-information" minOccurs="0"/>
5844         </sequence>
5845         <attribute name="id" type="ID"/>
5846     </complexType>
5847 </element>
5848
5849 <element name="hazard" substitutionGroup="btp:message">
5850 <complexType>
5851 <sequence>
5852     <element name="superior-identifier" type="btp:identifier"/>
5853     <element name="inferior-identifier" type="btp:identifier"/>
5854     <element name="level">
5855         <simpleType>
5856             <restriction base="string">
5857                 <enumeration value="mixed"/>
5858                 <enumeration value="possible"/>
5859             </restriction>
5860         </simpleType>
5861     </element>
5862     <element ref="btp:qualifiers" minOccurs="0"/>
5863     <element name="target-additional-information"
5864 type="btp:additional-information" minOccurs="0"/>
5865 </sequence>
5866 <attribute name="id" type="ID"/>
5867 </complexType>
5868 </element>
5869
5870 <element name="contradiction" substitutionGroup="btp:message">
5871 <complexType>
5872 <sequence>
5873     <element name="inferior-identifier" type="btp:identifier"/>
5874     <element ref="btp:qualifiers" minOccurs="0"/>
5875     <element name="target-additional-information"
5876 type="btp:additional-information" minOccurs="0"/>
5877 </sequence>
5878 <attribute name="id" type="ID"/>
5879 </complexType>
5880 </element>
5881
5882 <element name="superior-state" substitutionGroup="btp:message">
5883 <complexType>
5884 <sequence>
5885     <element name="inferior-identifier" type="btp:identifier"/>
5886     <element name="status">
5887         <simpleType>
5888             <restriction base="string">
5889                 <enumeration value="active"/>
5890                 <enumeration value="prepared-received"/>

```

```

5891         <enumeration value="inaccessible"/>
5892         <enumeration value="unknown"/>
5893     </restriction>
5894 </simpleType>
5895 </element>
5896 <element name="response-requested" type="boolean"/>
5897 <element ref="btp:qualifiers" minOccurs="0"/>
5898 <element name="target-additional-information"
5899 type="btp:additional-information" minOccurs="0"/>
5900 </sequence>
5901 <attribute name="id" type="ID"/>
5902 </complexType>
5903 </element>
5904
5905 <element name="inferior-state" substitutionGroup="btp:message">
5906 <complexType>
5907 <sequence>
5908 <element name="superior-identifier" type="btp:identifier"/>
5909 <element name="inferior-identifier" type="btp:identifier"/>
5910 <element name="status">
5911 <simpleType>
5912 <restriction base="string">
5913 <enumeration value="active"/>
5914 <enumeration value="inaccessible"/>
5915 <enumeration value="unknown"/>
5916 </restriction>
5917 </simpleType>
5918 </element>
5919 <element name="response-requested" type="boolean"/>
5920 <element ref="btp:qualifiers" minOccurs="0"/>
5921 <element name="target-additional-information"
5922 type="btp:additional-information" minOccurs="0"/>
5923 </sequence>
5924 <attribute name="id" type="ID"/>
5925 </complexType>
5926 </element>
5927
5928 <element name="redirect" substitutionGroup="btp:message">
5929 <complexType>
5930 <sequence>
5931 <element name="superior-identifier" type="btp:identifier"
5932 minOccurs="0"/>
5933 <element name="inferior-identifier" type="btp:identifier"
5934 />
5935 <element name="old-address" type="btp:address"
5936 maxOccurs="unbounded"/>
5937 <element name="new-address" type="btp:address"
5938 maxOccurs="unbounded"/>
5939 <element ref="btp:qualifiers" minOccurs="0"/>
5940 <element name="target-additional-information"
5941 type="btp:additional-information" minOccurs="0"/>
5942 </sequence>

```

```

5943         <attribute name="id" type="ID"/>
5944     </complexType>
5945 </element>
5946
5947
5948     <element name="begin" substitutionGroup="btp:message">
5949         <complexType>
5950             <sequence>
5951                 <element name="transaction-type" type="btp:superior-type"/>
5952                 <element ref="btp:qualifiers" minOccurs="0"/>
5953                 <element name="target-additional-information"
5954 type="btp:additional-information" minOccurs="0"/>
5955                 <element name="reply-address" type="btp:address"
5956 minOccurs="0"/>
5957             </sequence>
5958             <attribute name="id" type="ID"/>
5959         </complexType>
5960     </element>
5961
5962     <element name="begun" substitutionGroup="btp:message">
5963         <complexType>
5964             <sequence>
5965                 <element name="decider-address" type="btp:address"
5966 minOccurs="0" maxOccurs="unbounded"/>
5967                 <element name="transaction-identifier"
5968 type="btp:identifier" minOccurs="0"/>
5969                 <element name="inferior-handle" type="btp:identifier"
5970 minOccurs="0"/>
5971                 <element name="inferior-address" type="btp:address"
5972 minOccurs="0" maxOccurs="unbounded"/>
5973                 <element ref="btp:qualifiers" minOccurs="0"/>
5974                 <element name="target-additional-information"
5975 type="btp:additional-information" minOccurs="0"/>
5976             </sequence>
5977             <attribute name="id" type="ID"/>
5978         </complexType>
5979     </element>
5980
5981     <element name="prepare-inferiors" substitutionGroup="btp:message">
5982         <complexType>
5983             <sequence>
5984                 <element name="transaction-identifier"
5985 type="btp:identifier"/>
5986                 <element name="inferiors-list" minOccurs="0">
5987                     <complexType>
5988                         <sequence>
5989                             <element name="inferior-handle"
5990 type="btp:identifier" maxOccurs="unbounded"/>
5991                         </sequence>
5992                     </complexType>
5993                 </element>
5994                 <element ref="btp:qualifiers" minOccurs="0"/>

```

```

5995         <element name="target-additional-information"
5996 type="btp:additional-information" minOccurs="0"/>
5997         <element name="reply-address" type="btp:address"
5998 minOccurs="0"/>
5999     </sequence>
6000     <attribute name="id" type="ID"/>
6001 </complexType>
6002 </element>
6003
6004     <element name="confirm-transaction" substitutionGroup="btp:message">
6005     <complexType>
6006     <sequence>
6007         <element name="transaction-identifier"
6008 type="btp:identifier"/>
6009         <element name="inferiors-list" minOccurs="0">
6010         <complexType>
6011         <sequence>
6012             <element name="inferior-handle"
6013 type="btp:identifier" maxOccurs="unbounded"/>
6014         </sequence>
6015         </complexType>
6016     </element>
6017     <element name="report-hazard" type="boolean"/>
6018     <element ref="btp:qualifiers" minOccurs="0"/>
6019     <element name="target-additional-information"
6020 type="btp:additional-information" minOccurs="0"/>
6021     <element name="reply-address" type="btp:address"
6022 minOccurs="0"/>
6023     </sequence>
6024     <attribute name="id" type="ID"/>
6025 </complexType>
6026 </element>
6027
6028     <element name="transaction-confirmed" substitutionGroup="btp:message">
6029     <complexType>
6030     <sequence>
6031         <element name="transaction-identifier"
6032 type="btp:identifier"/>
6033         <element ref="btp:qualifiers" minOccurs="0"/>
6034         <element name="target-additional-information"
6035 type="btp:additional-information" minOccurs="0"/>
6036     </sequence>
6037     <attribute name="id" type="ID"/>
6038 </complexType>
6039 </element>
6040
6041     <element name="cancel-transaction" substitutionGroup="btp:message">
6042     <complexType>
6043     <sequence>
6044         <element name="transaction-identifier"
6045 type="btp:identifier"/>
6046         <element name="report-hazard" type="boolean"/>

```

```

6047         <element ref="btp:qualifiers" minOccurs="0"/>
6048         <element name="target-additional-information"
6049 type="btp:additional-information" minOccurs="0"/>
6050         <element name="reply-address" type="btp:address"
6051 minOccurs="0"/>
6052     </sequence>
6053     <attribute name="id" type="ID"/>
6054 </complexType>
6055 </element>
6056
6057     <element name="cancel-inferiors" substitutionGroup="btp:message">
6058     <complexType>
6059     <sequence>
6060         <element name="transaction-identifier"
6061 type="btp:identifier" minOccurs="0"/>
6062         <element name="inferiors-list">
6063         <complexType>
6064         <sequence>
6065             <element name="inferior-handle"
6066 type="btp:identifier" maxOccurs="unbounded"/>
6067         </sequence>
6068         </complexType>
6069     </element>
6070     <element ref="btp:qualifiers" minOccurs="0"/>
6071     <element name="target-additional-information"
6072 type="btp:additional-information" minOccurs="0"/>
6073     <element name="reply-address" type="btp:address"
6074 minOccurs="0"/>
6075     </sequence>
6076     <attribute name="id" type="ID"/>
6077 </complexType>
6078 </element>
6079
6080     <element name="transaction-cancelled" substitutionGroup="btp:message">
6081     <complexType>
6082     <sequence>
6083         <element name="transaction-identifier"
6084 type="btp:identifier"/>
6085         <element ref="btp:qualifiers" minOccurs="0"/>
6086         <element name="target-additional-information"
6087 type="btp:additional-information" minOccurs="0"/>
6088     </sequence>
6089     <attribute name="id" type="ID"/>
6090 </complexType>
6091 </element>
6092
6093     <element name="request-inferior-statuses"
6094 substitutionGroup="btp:message">
6095     <complexType>
6096     <sequence>
6097         <element name="target-identifier" type="btp:identifier"/>
6098         <element name="inferiors-list" minOccurs="0">

```

```

6099         <complexType>
6100             <sequence>
6101                 <element name="inferior-handle"
6102 type="btp:identifier" maxOccurs="unbounded"/>
6103             </sequence>
6104         </complexType>
6105     </element>
6106     <element ref="btp:qualifiers" minOccurs="0"/>
6107     <element name="target-additional-information"
6108 type="btp:additional-information" minOccurs="0"/>
6109     <element name="reply-address" type="btp:address"
6110 minOccurs="0"/>
6111     </sequence>
6112     <attribute name="id" type="ID"/>
6113 </complexType>
6114 </element>
6115
6116     <element name="inferior-statuses" substitutionGroup="btp:message">
6117         <complexType>
6118             <sequence>
6119                 <element name="responders-identifier"
6120 type="btp:identifier"/>
6121                 <element name="status-list">
6122                     <complexType>
6123                         <sequence>
6124                             <element name="status-item" maxOccurs="unbounded">
6125                                 <complexType>
6126                                     <sequence>
6127                                         <element name="inferior-handle"
6128 type="btp:identifier"/>
6129                                         <element name="status">
6130                                             <simpleType>
6131                                                 <restriction base="string">
6132                                                     <enumeration value="active"/>
6133                                                     <enumeration value="resigned"/>
6134                                                     <enumeration value="preparing"/>
6135                                                     <enumeration value="prepared"/>
6136                                                     <enumeration value="autonomously-confirmed"/>
6137                                                     <enumeration value="autonomously-cancelled"/>
6138                                                     <enumeration value="confirming"/>
6139                                                     <enumeration value="confirmed"/>
6140                                                     <enumeration value="cancelling"/>
6141                                                     <enumeration value="cancelled"/>
6142                                                     <enumeration value="cancel-contradiction"/>
6143                                                     <enumeration value="confirm-contradiction"/>
6144                                                     <enumeration value="hazard"/>
6145                                                     <enumeration value="invalid"/>
6146                                                 </restriction>
6147                                             </simpleType>
6148                                         </element>
6149                                         <element ref="btp:qualifiers" minOccurs="0"/>
6150                                     </sequence>

```

```

6151         </complexType>
6152     </element>
6153 </sequence>
6154 </complexType>
6155 </element>
6156     <element ref="btp:qualifiers" minOccurs="0"/>
6157     <element name="target-additional-information"
6158 type="btp:additional-information" minOccurs="0"/>
6159 </sequence>
6160     <attribute name="id" type="ID"/>
6161 </complexType>
6162 </element>
6163
6164
6165 </schema>
6166

```

XML schema for standard qualifiers

```

6167
6168
6169 <?xml version="1.0"?>
6170 <schema
6171     xmlns="http://www.w3.org/2001/XMLSchema"
6172     targetNamespace="urn:oasis:names:tc:BTP:1.0:qualifiers"
6173     xmlns:btpq="urn:oasis:names:tc:BTP:1.0:qualifiers"
6174     xmlns:btp="urn:oasis:names:tc:BTP:1.0:core"
6175     elementFormDefault="qualified">
6176
6177
6178     <element name="transaction-timelimit"
6179 substitutionGroup="btp:qualifier">
6180         <complexType>
6181             <complexContent>
6182                 <extension base="btp:qualifier-type">
6183                     <sequence>
6184                         <element name="timelimit"
6185 type="nonNegativeInteger"/>
6186                     </sequence>
6187                 </extension>
6188             </complexContent>
6189         </complexType>
6190     </element>
6191
6192     <element name="inferior-timeout" substitutionGroup="btp:qualifier">
6193         <complexType>
6194             <complexContent>
6195                 <extension base="btp:qualifier-type">
6196                     <sequence>
6197                         <element name="timelimit"
6198 type="nonNegativeInteger"/>
6199                         <element name="intended-decision">
6200                             <simpleType>
6201                                 <restriction base="string">
6202                                     <enumeration value="confirm"/>

```

```

6203         <enumeration value="cancel"/>
6204     </restriction>
6205 </simpleType>
6206 </element>
6207 </sequence>
6208 </extension>
6209 </complexContent>
6210 </complexType>
6211 </element>
6212
6213 <element name="minimum-inferior-timeout"
6214 substitutionGroup="btp:qualifier">
6215 <complexType>
6216 <complexContent>
6217 <extension base="btp:qualifier-type">
6218 <sequence>
6219 <element name="minimum-timeout"
6220 type="nonNegativeInteger"/>
6221 </sequence>
6222 </extension>
6223 </complexContent>
6224 </complexType>
6225 </element>
6226
6227 <element name="inferior-name" substitutionGroup="btp:qualifier">
6228 <complexType>
6229 <complexContent>
6230 <extension base="btp:qualifier-type">
6231 <sequence>
6232 <element name="inferior-name" type="string"/>
6233 </sequence>
6234 </extension>
6235 </complexContent>
6236 </complexType>
6237 </element>
6238
6239 </schema>
6240

```

6240
6241

6242 **Carrier Protocol Bindings**

6243

6244 The notion of bindings is introduced to act as the glue between the BTP messages and an
6245 underlying transport. A binding specification must define various particulars of how the BTP
6246 messages are carried and some aspects of how the related application messages are carried.

6247 This document specifies two bindings: a SOAP binding and a SOAP + Attachments binding.
6248 However, other bindings could be specified by the Oasis BTP technical committee or by a
6249 third party. For example, in the future a binding might exist to put a BTP message directly on
6250 top of HTTP without the use of SOAP, or a closed community could define their own
6251 binding. To ensure that such specifications are complete, the Binding Proforma defines the
6252 information that must be included in a binding specification.
6253

6254 **Carrier Protocol Binding Proforma**

6255

6256 A BTP carrier binding specification should provide the following information:
6257

6258 **Binding name:** A name for the binding, as used in the “binding name” field of BTP
6259 addresses (and available for declaring the capabilities of an implementation). Binding
6260 specified in this document, and future revisions of this document have binding names that are
6261 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).
6262 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in
6263 this document use numbers to identify the version of the binding, not the version(s) of the
6264 carrier protocol.

6265

6266 **Binding address format:** This section states the format of the “binding address” field of a
6267 BTP address for this binding. For many bindings, this will be a URL of some kind; for other
6268 bindings it may be some other form
6269

6270 **BTP message representation:** This section will define how BTP messages are represented.
6271 For many bindings, the BTP message syntax will be as specified in the XML schema defined
6272 in this document, and the normal string encoding of that XML will be used.
6273

6274

6275 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that
6276 are not related to application messages are sent in either direction between Superior and
6277 Inferior. (i.e. those messages sent directly between BTP actors). This mapping need not be
6278 symmetric (i.e. Superior to Inferior may differ to some degree to Inferior to Superior). The
6279 mapping may define particular rules for particular BTP messages, or messages with particular
6280 parameter values (e.g. the FAULT message with “fault-type” “CommunicationFailure” will
6281 typically not be sent as a BTP message). The mapping states any constraints or requirements
6282 on which BTP may or must be bundled together by compounding.

6282

6283 **Mapping for BTP messages related to application messages:** This section will define how
6284 BTP messages that are related to application messages are sent. A binding specification may

6285 defer details of this to a particular application (e.g. a mapping specification could just say
6286 “the CONTEXT may be carried as a parameter of an application invocation”). Alternatively,
6287 the binding may specify a general method that represents the relationship between application
6288 and BTP messages.

6289

6290 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly
6291 but are treated as implicit in carrier-protocol mechanisms, application messages or other BTP
6292 messages. This may depend on particular parameter values of the BTP messages or the
6293 application messages.

6294

6295 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
6296 protocol and of BTP shall be defined. This may include definition of which carrier protocol
6297 exceptions are equivalent to a FAULT/communication-failure message.

6298

6299 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.
6300 If BTP addresses with different bindings are be considered to match (for purposes of
6301 identifying the peer Superior/Inferior and redirection), this should be specified here.

6302

6303 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are
6304 imposed by use of this binding should be listed. This would include limitations on which
6305 messages can be sent, which event sequences are supported and restrictions on parameter
6306 values. Such limitations may reduce the usefulness of an implementation, but may be
6307 appropriate in certain environments.

6308

6309 **Other:** Other features of the binding, especially any that will potentially affect interoperability
6310 should be specified here. This may include restrictions or requirements on the use or support
6311 of optional carrier parameters or mechanisms [or use of standard or other qualifiers](#).

6312

6313 **Bindings for request/response carrier protocols**

6314

6315 BTP does not generally follow a request/response pattern. In particular, on the outcome
6316 relationship either side may initiate a message – this is an essential part of the presume-abort
6317 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
6318 messages, especially in the control relationship, that do have a request/response pattern.
6319 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The
6320 specification of a binding specification to a request/response carrier protocol needs to state
6321 what rules apply – which messages can be carried by requests, which by responses. The
6322 simplest rule is to send all BTP messages on requests, and let the carrier responses travel back
6323 empty. This would be inefficient in use of network resources, and possibly inconvenient
6324 when used for the BTP request/response pairs.

6325

6326 This section defines a set of rules that allow more efficient use of the carrier, while allowing
6327 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the
6328 carrier response. These rules are specified in this section to enable binding specifications to
6329 reference them, without requiring each binding specification to repeat similar information.
6330 These rules also allow the receiver of a message between Superior and Inferior (in either

6331 direction) on a carrier protocol request to send any reply message on the carrier response –
6332 the “sender-address” field is implicitly considered to be that of the sender of the carrier
6333 request.

6334

6335 A binding to a request/response carrier is not required to use these rules. It may define other
6336 rules.

6337

6338 Request/response exploitation rules

6339

6340 These rules allow implementations to use the request and response of the carrier protocol
6341 efficiently, and, when a BTP request/response exchange occurs, to either treat the
6342 request/response exchanges of the carrier protocol and of BTP independently, if both sides
6343 wish, or allow either side to map them closely.

6344

6345 Under these rules, an implementation sending a BTP request (i.e. a message, other than
6346 CONTEXT, which has “reply-address” as a parameter in the abstract message definition), can
6347 ensure that it and the reply map to a carrier request/response by supplying no value for the
6348 “reply-address”. An implementation receiving such a request is required to send the BTP
6349 response on the carrier response.

6350

6351 Conversely, if an implementation does supply a “reply-address” value on the request, the
6352 receiver has the option of sending the BTP response back on the carrier response, or sending
6353 it on a new carrier request.

6354

6355 Within the outcome relationship, apart from ENROL, there is no “reply-address”, and the
6356 parties normally know each other’s “superior-address” and “inferior-address”. However,
6357 these messages have a “sender-address”, which is used when the receiver does not have
6358 knowledge of the peer. In this case, the “sender-address” is treated as the “reply-address” of
6359 the other messages – if the field is absent in a message on a carrier request, the “sender-
6360 address” is implicitly that of the request sender. Any message for the peer (including the three
6361 messages mentioned, FAULT but also any other valid message in the Superior:Inferior
6362 relationship) may be sent on the carrier response. Apart from this, both sides are permitted to
6363 treat the carrier request/response exchanges as opportunities for sending messages to the
6364 appropriate destination.

6365

6366 The rules:

6367

6368 a) A BTP actor **may** bundle one or more BTP messages and related groups that
6369 have the same binding address for their target in a single btp:messages and
6370 transmit this btp:messages element on a carrier protocol request. There is no
6371 restriction on which combinations of messages and groups may be so bundled,
6372 other than that they have the same binding address, and that this binding address
6373 is usable as the destination of a carrier protocol request.

6374

6375 b) A BTP actor that has received a carrier protocol request to which it has not yet
6376 responded, and which has one or more BTP messages and groups whose binding

- 6377 address for the target matches the origin of the carrier request **may** bundle such
6378 BTP messages in a single `btm:messages` element and transmit that on the carrier
6379 protocol response.
6380
- 6381 c) A BTP actor that has received, on a carrier protocol request, one or more BTP
6382 messages or related groups that require a BTP response and for which no “reply-
6383 address” was supplied, **must** bundle the responding BTP message and groups in a
6384 `btm:messages` element and transmit this element on the carrier protocol response
6385 to the request that carried the BTP request.
6386
- 6387 d) A BTP actor that has received, on a carrier protocol request, one or more BTP
6388 messages or related groups that, as abstract messages, have a “sender-address”
6389 parameter but no “reply-address” was supplied and does not have knowledge of
6390 the peer address, **must** bundle the responding BTP message and groups in a
6391 `btm:messages` element and transmit this element on the carrier protocol response
6392 to the request that carried the BTP request. If the actor does have knowledge of
6393 the peer address it **may** send one or messages for the peer in the carrier protocol
6394 response, regardless of whether the binding address of the peer matches the
6395 address of the carrier protocol requestor.
6396
- 6397 e) Where only one message or group is to be sent, it shall be contained within a
6398 `btm:messages` element, as a bundle of one element.
6399
- 6400 f) A BTP actor that receives a carrier protocol request carrying BTP messages that
6401 do have a “reply-address”, or which initiate processing that produces BTP
6402 messages whose target binding address matches the origin of the request, **may**
6403 freely choose whether to use the carrier protocol response for the replies, or to
6404 send back an “empty carrier protocol response”, and send the BTP replies in a
6405 separately initiated carrier protocol request. The characteristics of an “empty
6406 carrier protocol response” shall be stated in the particular binding specification.
6407
- 6408 g) A BTP actor that sends BTP messages on a carrier protocol request **must** be able
6409 to accept returning BTP messages on the corresponding carrier protocol response
6410 and, if the actor has offered an address on which it will receive carrier requests,
6411 must be able to accept “replying” BTP messages on a separate carrier protocol
6412 request.
6413

6414 SOAP Binding

6415
6416 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
6417 specification, using the SOAP literal messaging style conventions. If no application message
6418 is sent at the same time, the BTP messages are contained within the SOAP Body element. If
6419 application messages are sent, the BTP messages are contained in the SOAP Header element.
6420

6421 **Binding name:** soap-http-1
6422

6423
6424
6425
6426
6427
6428
6429
6430
6431
6432
6433
6434
6435
6436
6437
6438
6439
6440
6441
6442
6443
6444
6445
6446
6447
6448
6449
6450
6451
6452
6453
6454
6455
6456
6457
6458
6459
6460
6461
6462

6463
6464
6465

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

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

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

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

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

- a) an empty HTTP response
- b) an HTTP response containing an empty SOAP Envelope
- c) an HTTP response containing a SOAP Envelope containing a single, empty btp:messages element.

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

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.

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.

6466 Only CONTEXT and ENROL messages are related (&) to application messages. If there is
6467 only one CONTEXT or one ENROL message present in the SOAP Header, it is assumed to
6468 be related to the whole of the application message in the SOAP Body. If there are multiple
6469 CONTEXT or ENROL messages, any relation of these BTP messages shall be indicated by
6470 application specific means.

6471 Note 1 – An application protocol could use references to the ID values of the
6472 BTP messages to indicate relation between BTP CONTEXT or ENROL
6473 messages and the application message.

6474 Note 2 -- However indicated, what the relatedness means, or even whether it
6475 has any significance at all, is a matter for the application.

6476
6477 **Implicit messages:** A SOAP FAULT, or other communication failure received in response to
6478 a SOAP request that had a CONTEXT in the SOAP Header shall be treated as if a
6479 CONTEXT_REPLY/repudiated had been received. See also the discussion under “other”
6480 about the SOAP mustUnderstand attribute.

6481
6482 **Faults:** A SOAP FAULT or other communication failure shall be treated as
6483 FAULT/communication-failure.

6484
6485 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
6486 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
6487 http-1” if the binding address and additional information fields match.

6488
6489 **Limitations on BTP use:** None

6490
6491 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
6492 attribute. The SOAPAction HTTP header is left to be application specific when there are
6493 application messages in the SOAP Body, as an already existing web service that is being
6494 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
6495 header shall ~~be omitted~~contain no value when the SOAP message carries only BTP messages
6496 in the SOAP Body.

6497
6498 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP
6499 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to
6500 determine whether any enrolments are necessary and replies with CONTEXT_REPLY as
6501 appropriate. The sender of the CONTEXT (and related application message) can use this to
6502 ensure that the application work is performed as part of the business transaction, assuming the
6503 receiver’s SOAP implementation supports the mustUnderstand attribute. If mustUnderstand if
6504 false, a receiver can ignore the CONTEXT (if BTP is not supported there), and no
6505 CONTEXT_REPLY will be returned. It is a local option on the sender (client) side whether
6506 the absence of a CONTEXT_REPLY is assumed to be equivalent to aCONTEXT_REPLY/ok
6507 (and the business transaction allowed to proceed to confirmation).
6508

6509 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to
6510 enforce these requirements.

6511 Example scenario using SOAP binding

6512

6513 The example below shows an application request with CONTEXT message sent from
6514 client.example.com (which includes the Superior) to services.example.com (Service).

6515

6516

6517

6518

6519

6520

6521

6522

6523

6524

6525

6526

6527

6528

6529

6530

6531

6532

6533

6534

6535

6536

6537

6538

6539

6540

6541

6542

6543

6544

6545

6546

6547

6548

6549

6550

6551

6552

6553

6554

6555

6556

6557

6558

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  soap:encodingStyle="">
  <soap:Header>
    <ctp:messages xmlns:ctp="urn:oasis:names:tc:BTP:1.0:core">
      <ctp:context superior-type="atom">
        <ctp:superior-address>
          <ctp:binding>soap-http-1</ctp:binding>
          <ctp:binding-
address>http://client.example.com/soaphandler</ctp:binding-
address>
          <ctp:additional-information>ctpengine</ctp:additional-
information>
        </ctp:superior-address>
        <ctp:superior-
identifier>http://example.com/1001</ctp:superior-identifier>
        <ctp:qualifiers>
          <ctpq:transaction-timelimit
xmlns:ctpq="urn:oasis:names:tc:BTP:1.0:qualifiers"><ctpq:timelimit
>1800</ctpq:timelimit></ctpq:transaction-timelimit>
        </ctp:qualifiers>
      </ctp:context>
    </ctp:messages>
  </soap:Header>
  <soap:Body>
    <ns1:orderGoods
xmlns:ns1="http://example.com/2001/Services/xyzgoods">
      <custID>ABC8329045</custID>
      <itemID>224352</itemID>
      <quantity>5</quantity>
    </ns1:orderGoods>
  </soap:Body>
</soap:Envelope>
```

6559 The example below shows CONTEXT_REPLY and a related ENROL message sent from
6560 services.example.com to client.example.com, in reply to the previous message. There is no
6561 application response, so the BTP messages are in the SOAP Body. The ENROL message
6562 does not contain the target-additional-information, since the grouping rules for
6563 CONTEXT_REPLY & ENROL omit the "target-address" (the receiver of this example
6564 remembers the superior address from the original CONTEXT)

```
6565
6566 <soap:Envelope
6567     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
6568     soap:encodingStyle="">
6569
6570     <soap:Header>
6571     </soap:Header>
6572
6573     <soap:Body>
6574
6575         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core">
6576             <btp:related-group>
6577                 <btp:context-reply>
6578                     <btp:target-additional-information>btpengine</btp:target-
6579 additional-information>
6580                     <btp:superior-
6581 identifier>http://example.com/1001</btp:superior-identifier>
6582                     <completion-status>related</completion-status>
6583                     </btp:context-reply>
6584
6585                     <btp:enrol response-requested="false">
6586                         <btp:target-additional-
6587 information>btpengine</btp:target-additional-information>
6588                         <btp:superior-
6589 identifier>http://example.com/1001</btp:superior-identifier>
6590                         <btp:inferior-address>
6591                             <btp:binding>soap-http-1</btp:binding>
6592                             <btp:binding-address>
6593                                 http://services.example.com/soaphandler
6594                             </btp:binding-address>
6595                         </btp:inferior-address>
6596                         <btp:inferior-identifier>
6597                             http://example.com/AAAB
6598                         </btp:inferior-identifier>
6599                     </btp:enrol>
6600
6601                 </btp:related-group>
6602
6603             </btp:messages>
6604
6605         </soap:Body>
6606
6607     </soap:Envelope>
6608
6609
```

6610
6611
6612
6613
6614
6615
6616
6617
6618
6619
6620
6621
6622
6623
6624
6625
6626
6627
6628
6629
6630
6631
6632
6633
6634
6635
6636
6637
6638
6639
6640
6641
6642
6643
6644
6645
6646
6647
6648
6649
6650
6651
6652
6653
6654
6655

SOAP + Attachments Binding

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

Binding name: soap-attachments-http-1

Binding address format: as for soap-http-1

BTP message representation: As for soap-http-1

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

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

Implicit messages: As for soap-http-1.

Faults: As for soap-http-1.

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

Limitations on BTP use: None

Other: As for soap-http-1

Example using SOAP + Attachments binding

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

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

```

6656
6657 <?xml version='1.0' ?>
6658 <soap:Envelope
6659     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
6660     soap:encodingStyle=" " >
6661
6662     <soap:Header>
6663
6664         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core">
6665             <btp:context superior-type="atom">
6666                 <btp:superior-address>
6667                     <btp:binding>soap-http-1</btp:binding>
6668                     <btp:binding-address>
6669                         http://client.example.com/soaphandler
6670                     </btp:binding-address>
6671                     </btp:superior-address>
6672                     <btp:superior-
6673 identifier>http://example.com/1001</btp:superior-identifier>
6674                 </btp:context>
6675             </btp:messages>
6676
6677         </soap:Header>
6678
6679         <soap:Body>
6680             <orderGoods href="cid:anotherID"/>
6681         </soap:Body>
6682
6683     </soap:Envelope>
6684
6685     --MIME_boundary
6686     Content-Type: text/xml
6687     Content-ID: anotherID
6688
6689         <ns1:orderGoods
6690 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
6691             <custID>ABC8329045</custID>
6692             <itemID>224352</itemID>
6693             <quantity>5</quantity>
6694         </ns1:orderGoods>
6695
6696
6697     --MIME_boundary--
6698
6699

```

6700 **Conformance**

6701
6702 A BTP implementation need not implement all aspects of the protocol to be useful. The level
6703 of conformance of an implementation is defined by which roles it can support using the
6704 specified messages and carrier protocol bindings for interoperation with other
6705 implementations.

6706
6707
6708
6709
6710
6711
6712
6713
6714
6715
6716
6717

An ~~partially conformant~~ implementation may implement the functionality of some roles in a non-interoperable way – usually combining pairs of roles, such as Terminator and Decider; ~~giving that implementation's users comparable proprietary functionality.~~ Such an implementation is conformant in respect of the roles it does implement in accordance with this specification.

An implmentation can state which aspects of the BTP specification it conforms to in terms of which Roles it supports. Since most Roles cannot usefully be supported in isolation, (The following ~~Roles and~~ Role Groups ~~are can be~~ used to ~~define~~ describe implementation capabilities: conformance:

Role Group	Roles
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

6718

6719
6720
6721
6722
6723
6724
6725
6726
6727
6728
6729
6730
6731
6732
6733
6734
6735
6736
6737
6738
6739
6740
6741
6742

The Role Groups occupy different positions within a business transaction tree and thus require presence of implementations supporting other Role Groups:

Initiator/Terminator uses control relationship to Atomic Hub or Cohesive Hub to initiate and control Atoms or Cohesions. Initiator/Terminator would typically be a library linked with application software.

Atomic Hub and Cohesive Hub would often be standalone servers.

Cohesive Superior and Atomic Superior would provide the equivalent of Initiator/Terminator functionality by internal or proprietary means.

Cohesive Hubs, Atomic Hubs, Cohesive Superior and Atomic Superior use outcome relationships to Participants and to each other.

Participants will establish outcome relationships to implementations of any of the other Role Groups except Initiator/Terminator. A Participant “covers” a resource or application work of some kind. It should be noted that a Participant is unaffected by whether it is enrolled in an Atom or Cohesion – it gets only a single outcome.

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

6743

6744

6745

6746

6747

6748

6749

6750

6751

6752

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

6752
6753
6754
6755
6756
6757

~~Part 3. Appendices~~ Part 3. Glossary

The glossary is the subject of issue 4

~~A. Glossary~~

Actor

An entity that executes procedures, a software agent. (See also BTP Actor)

Address

An identifier for an endpoint.

Application

An actor, which uses the Business Transaction Protocol (in the context of this specification).

Also, a group of such actors, which may be distributed, that perform a common purpose.

(When used in phrases such as “determined by the Application”, it is not relevant to BTP whether this is determined by the owner of a single system or is explicitly part of the contract that defines the distributed collaborative application. When it is necessary to distinguish the responsibilities of a single party, the term “Application element” is used.)

Application element

An actor that communicates, using application protocols, with other application elements, as part of an overall distributed application. A single system may contain more than one application element.

Application Endpoint

An endpoint of an application message.

Application Message

A message produced by an application element and consumed by an application element.

Application Operation

An operation, which is started when an application message arrives.

Appropriate

In accordance with a pertinent contract or specification.

Atom

A set of participants, which are the direct inferiors of a node (which may have only one member), all of which will receive instructions that will result in a homogeneous outcome. That is they will be issued instructions to all confirm or all cancel. (Transitively, a set of operations whose effect is capable of counter effect.)

Atomic Business Transaction

A complete business transaction that follows the atom rules for every node in the transaction tree over space and time, so that all the participants in the transaction will receive instructions that will result in a homogeneous outcome. That is they will be issued instructions to all confirm or all cancel. (Transitively, a set of operations whose effect is capable of counter effect.)

Become prepared

Ensure that of a set of procedures is capable of being successfully instructed to cancel or to confirm.

BTP Actor

A software entity, or agent, that is able to take part in Business Transaction Protocol exchanges i.e. that sends or receives BTP messages. A BTP Actor may be capable of only playing a single role, or of playing several different roles concurrently and / or sequentially. A BTP Actor may be involved in one, or more, transactions, concurrently and / or sequentially.

BTP element

A BTP actor that supports an application element (or elements) but is not itself concerned with application messages or semantics.

(Business) Application Protocol

The messages, their meanings and their permitted sequences used to effect a change in the state of a business relationship.

(Business) application system

A system that contains one, or more, business applications, and resources such as volatile and persistent storage for business state information. It may also contain other things such as an operating system and BTP elements.

Business relationship agreement

The contract and / or set of agreements that govern and constrain a business relationship between two, or more, parties.

Business relationship

A *business relationship* is any distributed state held by the parties, which is subject to contractual constraints agreed by those parties.

Business Transaction Protocol (BTP)

The messages, their meanings and their permitted sequences defined in this specification. Its purpose is to provide the interactions (or signalling) required to coordinate the effects of application protocol to achieve a business transaction.

BTP-Address

A compound address consisting of 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.

Business transaction

A set of state changes that occur, or are desired, in computer systems controlled by some set of parties, and these changes are related in some application defined manner. A *business transaction* is subject to, and a part of, a *business relationship*. (BTP assumes that the parties involved in a *business transaction* have distinct and autonomous application systems, which do not require knowledge of each others’ implementation or internal state representations in volatile or persistent storage. Access to such loosely coupled systems is assumed to occur only through service interfaces.)

Cancel

Process a counter effect for the current effect of a set of procedures. There are a number of different ways that this may be achieved in practice.

Carrier Protocol

A protocol, which defines how the transmission of BTP messages occur.

Carrier Protocol Address (CPA)
Client

The address of an endpoint for a particular carrier protocol.

An actor, which sends application messages to services.

Cohesion

A set of participants, which are the direct inferiors of a node that may receive instructions that may result in different outcomes for each participant. That is they will be issued instructions to confirm or cancel according to the application logic. Participants may resign or be instructed to cancel until the confirm set is fixed. Once the confirm set for a cohesion is fixed, then all participants in the confirm set are treated atomically. That is they will all be instructed to confirm unless one, or more, cancel in which case all will be instructed to cancel. All participants not in the confirm set will be instructed to cancel.

Cohesive Business Transaction

A complete business transaction for which at least one node over space and time follows the cohesion rules. The other nodes in the transaction tree of a cohesive business transaction may follow either the cohesion rules or the atom rules.

Confirm

Ensure that the effect of a set of procedures is completed. There are a number of different ways that this may be achieved in practice.

Context

Information pertinent to a single transaction, or branch of a transaction.

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.

Control relationship

The application element:BTP element relationships that create the nodes of the transaction tree (Initiator:Factory) and drive the completion (Terminator:Decider).

Coordinator

A BTP actor, which is the top 'node' of a transaction and decides the outcome of its immediate branches according to the atom rules defined in this specification. It has a lifetime, which is coincident with that of the atom. A coordinator can issue instructions to prepare, cancel and confirm. These instructions take the form of BTP messages. A coordinator is identified by its transaction-identifier. A coordinator must also have a BTP Address to which participants can send BTP messages.

Counter effect

An appropriate effect intended to counteract a prior effect.

Counter effect contract

The contract, which governs the relationship between the effect and the counter effect of a procedure. In the absence of any other overriding contracts the counter effect contract is the promise that the **Counter effect** will attempt so far as is possible to reverse or cancel the **Effect** such that an observer (on completion of the **Counter effect**) is unaware that the **Effect** ever occurred, but this attempt cannot be guaranteed to succeed.

Decider

The top node of a transaction tree, a composer or a coordinator (so called because the Terminator can only request confirmation – the Decider makes the final determination). The term can always be interpreted as “Composer or Coordinator”.

It is the role at the other end of a control relationship to a Terminator.

Delivery parameter

A parameter of an abstract message that is concerned with the transmission of the message to its target or the transmission of an immediate reply.. Distinguished from Payload parameter.

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 counter effect of the effect, and this is known as a counter effect contract. An effect is **Completed** when the change inducing processing of the set of procedures is finished.

Endpoint

A sender or receiver.

Enroller

The BTP Actor role that informs a superior of the existence of an inferior.

Factory

The BTP Actor role that creates transaction contexts and deciders.

Inappropriate

In violation of a pertinent contract or specification.

Ineffectual

Describes a set of procedures, which has no effect.

Inferior

The end of end of a BTP node to BTP node relationship governed by the outcome protocol that is topologically further from the top of the transaction tree.

<u>Inferior-Address</u>	<u>The address used to communicate with an actor playing the role of an Inferior.</u>
<u>Inferior-identifier</u>	<u>A globally unambiguous identification of a particular Inferior within a single transaction (represented as an URI or equivalent).</u>
<u>Initiator</u>	<u>The BTP Actor role (an application element) that starts a transaction.</u>
<u>Intermediate</u>	<u>A node that is a sub-composer or a sub-coordinator. An alternative term to interposed.</u>
<u>Interposed</u>	<u>A node that is a sub-composer or a sub-coordinator. An alternative term to intermediate.</u>
<u>Message</u>	<u>A datum, which is produced and then consumed.</u>
<u>Node</u>	<u>A logical entity that is associated with a single transaction. A node is a composer, a coordinator, a sub-coordinator, a sub-composer, or a participant.</u>
<u>Operation</u>	<u>A procedure, which is started by a receiver when a message arrives at it.</u>
<u>Outcome</u>	<u>A decision to either cancel or confirm.</u>
<u>Outcome relationship</u>	<u>The Superior:Inferior relationship (i.e. between BTP actors within the transaction tree) and the Enroller:Superior relationship used in establishing it.</u>
<u>Participant</u>	<u>A participant is part of an application system that also contains one, or more, applications, which manipulate resources. It is a role of a BTP Actor that is (or is equivalent to) a set of procedures, which is capable of receiving instructions from another BTP Actor to prepare, cancel and confirm. These signals are used by the application(s) to determine whether to effect (confirm) or counter effect (cancel) the results of application operations. A participant must also have a BTP Address, to which these instructions will be delivered, in the form of BTP messages. A participant is identified by an inferior-identifier.</u>

<u>Payload parameter</u>	<u>A parameter of an abstract message that is will be received and processed or retained by the receiving BTP actor. The various identifier parameters are considered Payload parameters . Distinguished from Delivery parameter.</u>
<u>Peer</u>	<u>The other party in a two-party relationship, as in Superior to Inferior, or Sender to Receiver.</u>
<u>Provisional Effect</u>	<u>The changes induced by the incomplete or complete processing of a set of procedures by an actor, which are subject to later completion or counter-effecting. The provisional effect may or may not be observable by other actors.</u>
<u>Receiver</u>	<u>The consumer of a message.</u>
<u>Relationship parties</u>	<u>The legal entities that enter into an agreement that forms the basis of the relationship.</u>
<u>Responders-identifier</u>	<u>An identifier carried in a BTP message that can be interpreted as transaction-identifier, a superior-identifier, or an inferior-identifier according to the nature of the role in a BTP actor that is responding to a received message.</u>
<u>Role</u>	<u>The participation of a software agent in a particular relationship in a particular business transaction. The software agent performing a role is termed an Actor.</u>
<u>Sender</u>	<u>The producer of a message.</u>
<u>Service</u>	<u>An actor (an application element), which on receipt of application messages, may start an appropriate application operation. For example, a process that advertises an interface allowing defined RPCs (remote procedure calls) to be invoked by a remote client.</u>
<u>Status requestor</u>	<u>The BTP Actor role that requests the status of another BTP actor.</u>

Sub-composer

An actor, which is not the top 'node' of a transaction. It receives an outcome from its superior and decides the outcome of its immediate branches according to the cohesive rules defined in this specification. It has a lifetime, which is coincident with that of the cohesion. A sub-composer can issue instructions to prepare, cancel and confirm on individual branches. These instructions take the form of BTP messages. A sub-composer must also have at least one BTP Address to which lower nodes can send BTP messages.

Sub-coordinator

An actor, which is not the top 'node' of a transaction. It receives an outcome from its superior and propagates the outcome to its immediate branches according to the atom rules defined in this specification. It has a lifetime, which is coincident with that of this atom. A sub-coordinator can issue instructions to prepare, cancel and confirm. These instructions take the form of BTP messages. A sub-coordinator must also have at least one BTP Address to which lower nodes can send BTP messages.

Superior

The BTP role that will accept enrolments of Inferiors and subsequently inform the Inferior of the Outcome applicable to it.

A Superior will be one of Composer, Coordinator, Sub-composer, or Sub-coordinator.

A Superior is considered to be a Superior even if it currently has no enrolled Inferiors.

Superior-address

The set of BTP-addresses used to communicate with an actor playing the role of a Superior.

Superior-identifier

A globally unambiguous identifier of a particular Superior within a particular transaction (represented as an URI or equivalent).

Target-identifier

An identifier carried in a BTP message that can be interpreted as transaction-identifier, a superior-identifier, or an inferior identifier according to the nature of the role in a BTP actor that receives this identifier.

Terminator

A BTP role performed by an Application element communicating with a Decider to control the completion of the Business Transaction. Frequently will be identical to the Initiator, but distinguished because the control of the Business Transaction can be passed between Application elements.

Transaction

A complete unit of work as defined by an application. A transaction starts when a part of the distributed transaction first initiates some work that is to be a part of a new transaction. The transaction tree may grow and shrink over time and (logical) space. A transaction completes when all the participants in a transaction have completed (that is have replied to their confirm or cancel instruction).

Transaction tree

A pattern of BTP nodes that provides the coordination of a distributed application transaction. There is single top node (a Decider) that interacts with the initiating application (which is a part of a distributed application). The Decider node has one, or more outcome relationships with other BTP nodes (sub-composer, sub-coordinator, or participant nodes). Any intermediate nodes (Sub-composer or Sub-coordinator nodes) have exactly one relationship up the tree in which they act as Inferior, and one, or more, relationships down the tree in which they act as Superior. Participants are leaves of the tree. That is they have exactly one relationship up the tree in which they act as Inferior and no down tree relationships.

Transaction-identifier

A globally unambiguous identifier for a particular a Decider (represented as an URI or equivalent). A Decider is the top 'node' of the transaction and thus this identifier also unambiguously identifies the transaction. Often identical to the Superior-identifier of the Decider in its role as Superior, though the protocol does not require this.

Transmission

The passage of a message from a sender to a receiver.

6758