

1 Organization for the Advancement of Structured Information Systems

# 2 **Business Transaction Protocol**

3 An OASIS Committee Specification

4 Version 1.0

5 3 June 2002

6

7

7 **Copyright and related notices**

8 Copyright © The Organization for the Advancement of Structured Information Standards  
9 (OASIS), 2002. All Rights Reserved.

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

19 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
20 successors or assigns.

21 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
22 **DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT**  
23 **LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL**  
24 **NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF**  
25 **MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.**

26

27 OASIS takes no position regarding the validity or scope of any intellectual property or other  
28 rights that might be claimed to pertain to the implementation or use of the technology described  
29 in this document or the extent to which any license under such rights might or might not be  
30 available; neither does it represent that it has made any effort to identify any such rights.  
31 Information on OASIS's procedures with respect to rights in OASIS specifications can be found  
32 at the OASIS website. Copies of claims of rights made available for publication and any  
33 assurances of licenses to be made available, or the result of an attempt made to obtain a general  
34 license or permission for the use of such proprietary rights by implementors or users of this  
35 specification, can be obtained from the OASIS Executive Director.

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

39

39 **Acknowledgements**

40 The members of the OASIS Business Transactions Technical Committee contributed to the  
41 development of this specification. The following were members of the committee for at least part  
42 of the time from July 2001 until the agreement of the specification are listed below. Some TC  
43 members changed their affiliation to OASIS members, but remained members of the TC; multiple  
44 affiliations are shown separated by semi-colons

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.
William 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 Inc; individual
Mark Potts	Individual; Talking Blocks
Pal Takacsi-Nagy	BEA Systems, Inc.
James Tauber	Bowstreet; mValent
Sazi Temel	BEA Systems, Inc.
Steve Viens	Individual
Jim Webber	Hewlett-Packard Co.
Steve White	SeeBeyond Inc.

45

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

- 47 Alex Ceponkus (alex@ceponkus.org)  
48 Sanjay Dalal (sanjay.dalal@bea.com)  
49 Tony Fletcher (tony.fletcher@choreology.com)  
50 Peter Furniss (peter.furniss@choreology.com)  
51 Alastair Green (alastair.green@choreology.com)  
52 Bill Pope (zpope@pobox.com)

53

54 We thank Rocky Stewart and Pal Takacsi-Nagy of BEA Systems Inc and Bill Pope for their  
55 efforts in chairing the Technical Committee, and Karl Best of OASIS for his guidance on the  
56 organization of the Committee's work.

57

*In memory of Ed Felt*

58

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

59

60

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

61

62

63

He was killed in the crash of the hijacked United Airlines flight 93 near Pittsburgh,

64

on 11 September 2001.

65

65 **Typographical and Linguistic Conventions and Style**

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

68 Cancel  
69 Participant  
70 Application Message

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

72 **Coordinator**

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

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

76 BEGIN  
77 CONTEXT  
78 RESIGN

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

80 BEGIN/atom

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

82 BEGIN/atom & CONTEXT

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

84 ENROL + VOTE

85 XML schemata and instances are given in Courier and are shaded:

86 `<btpp:begin> ... </btpp:begin>`

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

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

92

92	<b>Contents</b>	
93	Copyright and related notices.....	2
94	Acknowledgements .....	3
95	Typographical and Linguistic Conventions and Style .....	5
96	Contents .....	6
97	<b>Part 1. Purpose and Features of BTP .....</b>	<b>11</b>
98	1 Introduction .....	11
99	2 Deferred topics .....	12
100	2.1 Conformance.....	12
101	2.2 Interoperation.....	12
102	2.3 Security .....	12
103	2.4 Transaction coordinator migration.....	12
104	3 Development and Maintenance of the Specification .....	13
105	4 Structure of this specification .....	14
106	5 Conceptual Model .....	15
107	5.1 Concepts .....	15
108	5.1.1 Business transactions.....	17
109	5.1.2 External Effects .....	17
110	5.1.3 Two-phase outcome .....	18
111	5.1.4 Actors and roles.....	19
112	5.1.5 Superior:Inferior relationship .....	19
113	5.1.6 Business Transaction Trees .....	20
114	5.1.7 Atoms and Cohesions .....	22
115	5.1.8 Participants, Sub-Coordinators and Sub-Composers.....	23
116	5.2 Business transaction lifecycle .....	23
117	5.2.1 Business Transaction creation .....	23
118	5.2.2 Business Transaction propagation .....	25
119	5.2.3 Creation of Intermediates (Sub-Coordinators and Sub-Composers).....	26
120	5.2.4 “Checking” and context-reply .....	27
121	5.2.5 Message sequence .....	28
122	5.2.6 Control of inferiors .....	33
123	5.2.7 Evolution of Confirm-set.....	36
124	5.2.8 Confirm-set of intermediates .....	40
125	5.3 Optimisations and variations .....	43
126	5.3.1 Spontaneous prepared.....	43
127	5.3.2 One-shot .....	44
128	5.3.3 Resignation.....	45
129	5.3.4 One-phase confirmation .....	46
130	5.3.5 Autonomous cancel, autonomous Confirm and contradictions.....	46
131	5.4 Recovery and failure handling .....	47
132	5.4.1 Types of failure .....	47
133	5.4.2 Persistent information.....	48
134	5.4.3 Recovery messages.....	49
135	5.4.4 Redirection .....	50
136	5.4.5 Terminator:Decider failures and transaction timelimit .....	51
137	5.4.6 Contradictions and hazard .....	51
138	5.5 Relation of BTP to application and Carrier Protocols .....	53
139	5.6 Other elements .....	55

140	5.6.1	Identifiers .....	55
141	5.6.2	Addresses .....	56
142	5.6.3	Qualifiers .....	57
143	<b>Part 2. Normative Specification of BTP .....</b>	<b>59</b>	
144	6	Actors, Roles and Relationships.....	59
145	6.1	Relationships.....	59
146	6.2	Roles .....	61
147	6.2.1	Roles involved in the Outcome Relationships.....	62
148	6.2.2	Superior .....	62
149	6.2.3	Inferior.....	63
150	6.2.4	Enroller.....	63
151	6.2.5	Participant.....	64
152	6.2.6	Sub-coordinator.....	64
153	6.2.7	Sub-composer.....	65
154	6.3	Roles involved in the Control Relationships .....	65
155	6.3.1	Decider .....	65
156	6.3.2	Coordinator.....	66
157	6.3.3	Composer .....	66
158	6.3.4	Terminator .....	67
159	6.3.5	Initiator .....	68
160	6.3.6	Factory.....	68
161	6.4	Other roles .....	68
162	6.4.1	Redirector .....	68
163	6.4.2	Status Requestor .....	69
164	6.5	Summary of relationships .....	70
165	7	Abstract Messages and Associated Contracts .....	71
166	7.1	Addresses.....	71
167	7.2	Request/response pairs.....	72
168	7.3	Compounding messages .....	73
169	7.4	Extensibility.....	75
170	7.5	Messages.....	75
171	7.5.1	Qualifiers.....	75
172	7.6	Messages not restricted to outcome or Control Relationships.....	76
173	7.6.1	CONTEXT .....	76
174	7.6.2	CONTEXT_REPLY.....	77
175	7.6.3	REQUEST_STATUS.....	78
176	7.6.4	STATUS.....	79
177	7.6.5	FAULT .....	81
178	7.6.6	REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES.....	83
179	7.7	Messages used in the Outcome Relationships .....	83
180	7.7.1	ENROL.....	83
181	7.7.2	ENROLLED.....	84
182	7.7.3	RESIGN .....	85
183	7.7.4	RESIGNED .....	86
184	7.7.5	PREPARE .....	87
185	7.7.6	PREPARED .....	87
186	7.7.7	CONFIRM.....	89
187	7.7.8	CONFIRMED .....	89
188	7.7.9	CANCEL.....	90

189	7.7.10	CANCELLED.....	91
190	7.7.11	CONFIRM_ONE_PHASE.....	92
191	7.7.12	HAZARD.....	93
192	7.7.13	CONTRADICTION.....	95
193	7.7.14	SUPERIOR_STATE.....	95
194	7.7.15	INFERIOR_STATE.....	97
195	7.7.16	REDIRECT.....	99
196	7.8	Messages used in Control Relationships.....	100
197	7.8.1	BEGIN.....	100
198	7.8.2	BEGUN.....	101
199	7.8.3	PREPARE_INFERIORS.....	102
200	7.8.4	CONFIRM_TRANSACTION.....	103
201	7.8.5	TRANSACTION_CONFIRMED.....	105
202	7.8.6	CANCEL_TRANSACTION.....	106
203	7.8.7	CANCEL_INFERIORS.....	107
204	7.8.8	TRANSACTION_CANCELLED.....	108
205	7.8.9	REQUEST_INFERIOR_STATUSES.....	109
206	7.8.10	INFERIOR_STATUSES.....	110
207	7.9	Groups – combinations of related messages.....	112
208	7.9.1	CONTEXT & Application Message.....	112
209	7.9.2	CONTEXT_REPLY & ENROL.....	113
210	7.9.3	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED.....	114
211	7.9.4	CONTEXT_REPLY & ENROL & Application Message (& PREPARED).....	114
212	7.9.5	BEGUN & CONTEXT.....	115
213	7.9.6	BEGIN & CONTEXT.....	115
214	7.10	Standard qualifiers.....	115
215	7.10.1	Transaction timelimit.....	115
216	7.10.2	Inferior timeout.....	116
217	7.10.3	Minimum inferior timeout.....	117
218	7.10.4	Inferior name.....	117
219	8	State Tables.....	118
220	8.1	Status queries.....	119
221	8.2	Decision events.....	119
222	8.3	Disruptions – failure events.....	120
223	8.4	Invalid cells and assumptions of the communication mechanism.....	120
224	8.5	Meaning of state table events.....	120
225	8.6	Persistent information.....	124
226	8.7	Superior state table.....	127
227	8.8	Inferior state table.....	131
228	9	Persistent information.....	136
229	10	XML representation of Message Set.....	136
230	10.1	Field types.....	137
231	10.1.1	Addresses.....	137
232	10.1.2	Qualifiers.....	137
233	10.1.3	Identifiers.....	138
234	10.1.4	Message References.....	138
235	10.2	Messages.....	138
236	10.2.1	CONTEXT.....	138
237	10.2.2	CONTEXT_REPLY.....	138

238	10.2.3	REQUEST_STATUS.....	139
239	10.2.4	STATUS .....	139
240	10.2.5	FAULT.....	139
241	10.2.6	ENROL .....	140
242	10.2.7	ENROLLED .....	141
243	10.2.8	RESIGN .....	141
244	10.2.9	RESIGNED.....	141
245	10.2.10	PREPARE.....	142
246	10.2.11	PREPARED .....	142
247	10.2.12	CONFIRM .....	142
248	10.2.13	CONFIRMED.....	142
249	10.2.14	CANCEL.....	143
250	10.2.15	CANCELLED.....	143
251	10.2.16	CONFIRM_ONE_PHASE.....	143
252	10.2.17	HAZARD .....	144
253	10.2.18	CONTRADICTION.....	144
254	10.2.19	SUPERIOR_STATE.....	144
255	10.2.20	INFERIOR_STATE.....	145
256	10.2.21	REDIRECT .....	145
257	10.2.22	BEGIN .....	145
258	10.2.23	BEGUN.....	146
259	10.2.24	PREPARE_INFERIORS .....	146
260	10.2.25	CONFIRM_TRANSACTION .....	146
261	10.2.26	TRANSACTION_CONFIRMED .....	147
262	10.2.27	CANCEL_TRANSACTION .....	147
263	10.2.28	CANCEL_INFERIORS .....	147
264	10.2.29	TRANSACTION_CANCELLED.....	148
265	10.2.30	REQUEST_INFERIOR_STATUSES.....	148
266	10.2.31	INFERIOR_STATUSES .....	148
267	10.3	Standard qualifiers .....	149
268	10.3.1	Transaction timelimit .....	149
269	10.3.2	Inferior timeout .....	149
270	10.3.3	Minimum inferior timeout .....	149
271	10.3.4	Inferior name.....	149
272	10.4	Compounding of Messages.....	149
273	10.5	XML Schemas .....	150
274	10.5.1	XML schema for BTP messages.....	150
275	10.5.2	XML schema for standard qualifiers.....	163
276	11	Carrier Protocol Bindings.....	165
277	11.1	Carrier Protocol Binding Proforma.....	165
278	11.2	Bindings for request/response Carrier Protocols .....	166
279	11.2.1	Request/response exploitation rules.....	167
280	11.3	SOAP Binding .....	168
281	11.3.1	Example scenario using SOAP binding .....	170
282	11.4	SOAP + Attachments Binding.....	172
283	11.4.1	Example using SOAP + Attachments binding.....	173
284	12	Conformance .....	173
285	<b>Part 3. Glossary .....</b>		<b>176</b>
286	<b>Part 4. Annexes .....</b>		<b>184</b>

287	13	Informational annex A Node State Information Serialisation .....	184
288	13.1	NODE STATE INFORMATION .....	184
289	13.1.1	Abstract Format for Node State Information .....	184
290	13.1.2	Informal XML for Node State Information .....	186
291	13.1.3	XML schema for Node State Information .....	187
292			

## 293 Part 1. Purpose and Features of BTP

### 294 1 Introduction

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

300 The OASIS BTP Technical Committee began its work at an inaugural meeting in San Jose, Calif.  
301 on 13 March 2001, and this specification was endorsed as a Committee Specification by a  
302 unanimous vote on 16<sup>th</sup> May 2002.

303 BTP is designed to allow coordination of application work between multiple participants owned  
304 or controlled by autonomous organizations. BTP uses a two-phase outcome coordination protocol  
305 to ensure the overall application achieves a consistent result. BTP permits the consistent outcome  
306 to be defined *a priori* -- all the work is confirmed or none is -- (an atomic business transaction or  
307 atom) or for application intervention into the selection of the work to be confirmed (a cohesive  
308 business transaction or cohesion).

309 BTP's ability to coordinate between services offered by autonomous organizations makes it  
310 ideally suited for use in a Web Services environment. For this reason this specification defines  
311 communications protocol bindings which target the emerging Web Services arena, while  
312 preserving the capacity to carry BTP messages over other communication protocols. Protocol  
313 message structure and content constraints are schematized in XML, and message content is  
314 encoded in XML instances.

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

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

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

327

327 **2 Deferred topics**

328 Certain issues were considered in the development of this document, but final and complete  
329 resolutions were not included in this edition. These areas are potential subjects for future work of  
330 the BTP Technical Committee.

331 **2.1 Conformance**

332 The BT Technical Committee recognizes that the approach to conformance taken in this  
333 Committee Specification (see section “12 Conformance” in part 2) may not fully meet the  
334 needs of consumers of an eventual OASIS Standard. We plan to evaluate the conformance  
335 requirements along with comments from implementers and users with a mind to decreasing the  
336 number of conformance points. Comments on this subject will be appreciated.

337 **2.2 Interoperation**

338 BTP is an interoperation protocol: assuming unambiguous specification and faithful  
339 implementation any two independent implementations using an agreed carrier-protocol binding  
340 should exchange and process BTP messages (sometimes in association with application  
341 messages) in such a way that they are mutually intelligible, are processed in sequence and with  
342 consequences as defined in this specification to give effect to agreed business-defined  
343 coordinated updates in all parties participating in a transaction.

344 In its work the BT Technical Committee began discussion of the issues involved in testing  
345 interoperability between implementations of BTP 1.0. Such testing can only be effected when  
346 using an agreed application protocol and data, and a common carrier protocol. Implementations of  
347 the carrier protocol concerned (e.g. SOAP 1.1/HTTP 1.1) may themselves be non-interoperable,  
348 and that issue can only be addressed independently by the body or bodies responsible for  
349 establishing interoperability for such a carrier protocol.

350 **2.3 Security**

351 The BT Technical Committee has consciously deferred addressing integration with security  
352 standards or technology. BTP version 1.0 therefore assumes that all actors are within a trust  
353 domain. Comments on this topic are invited.

354 **2.4 Transaction coordinator migration**

355 Migration of the transaction coordination roles is an important feature for scalable transaction  
356 systems. The BT Technical Committee plans to examine this issue before moving to an OASIS  
357 standard. Please see the Informative Annex A for a first step in this direction.

358

358 **3 Development and Maintenance of the Specification**

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

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

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

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

370 If you have a question about the functionality of BTP, or wish to report an error or to suggest a  
371 modification to the specification, please send a message to (and, if you wish, subscribe to):

372 [business-transaction-comment@lists.oasis-open.org](mailto:business-transaction-comment@lists.oasis-open.org)

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

375 The main list of the committee is:

376 [business-transaction@lists.oasis-open.org](mailto:business-transaction@lists.oasis-open.org)

377

## 377 **4 Structure of this specification**

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

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

383 Part 1 contains

- 384 • Executive Summary
- 385 • This document structure description
- 386 • Conceptual Model

387 Part 2 contains the following sections:

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

- 414           • Conformance definitions: defines combinations of facilities (expressed as roles) that an  
415           implementation can declare it supports

416 Part 3 contains a glossary that provides succinct definitions of terms used in the rest of the  
417 document.

418 Part 4 contains an informational annex that defines a format for the serialised state information of  
419 a BTP node.

## 420 **5 Conceptual Model**

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

### 425 **5.1 Concepts**

426 BTP is designed to make minimal assumptions about the implementation structure and the  
427 properties of the **Carrier Protocols**. This allows BTP to be bound to more than one Carrier  
428 Protocol. BTP implementations built in quite different ways should be able to interoperate if they  
429 are bound to the same Carrier Protocol. This flexibility requires that much of the text is abstract  
430 and may be difficult to visualise in the absence of a particular implementation pattern or Carrier  
431 Protocol. To aid understanding some possible implementation examples are presented in the  
432 following text.

#### 433 **Example Core**

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

- 438 1. *Manufacturer A*'s production scheduling system sends an Order message to a  
439 *Supplier*.
- 440 2. The *Supplier*'s order processing system sends back an order confirmation with the  
441 details of the order.
- 442 3. *Manufacturer A* orders delivery from a *Shipper* for the ordered parts.
- 443 4. The *Shipper* evaluates the request and based on its truck schedule it sends back a  
444 positive or negative reply.
- 445 5. Some shipments need to be insured based on their value, where they are shipped  
446 from, and method of transportation. *Manufacturer A* sends an Order message to an  
447 *Insurer* when this is necessary.
- 448 6. The *Insurer* responds with a bid or a no-bid response.

449

Problems have arisen with some of these interactions.

450

- Manufacturer A had ordered parts from a supplier and contacted shipper M about delivering the goods. Shipper M was busy and agreed to the contract but only for a scheduled delivery the day after the parts were needed. By the time this was addressed it was too late to schedule alternate shipping.

451

452

453

454

- There were communications problems with supplier Z that resulted in an order not being confirmed. The shipper arrived to pick up the order and supplier Z knew nothing about it.

455

456

457

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

458

459

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.

460

461

462

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 5-1. 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.

463

464

465

466

467

468

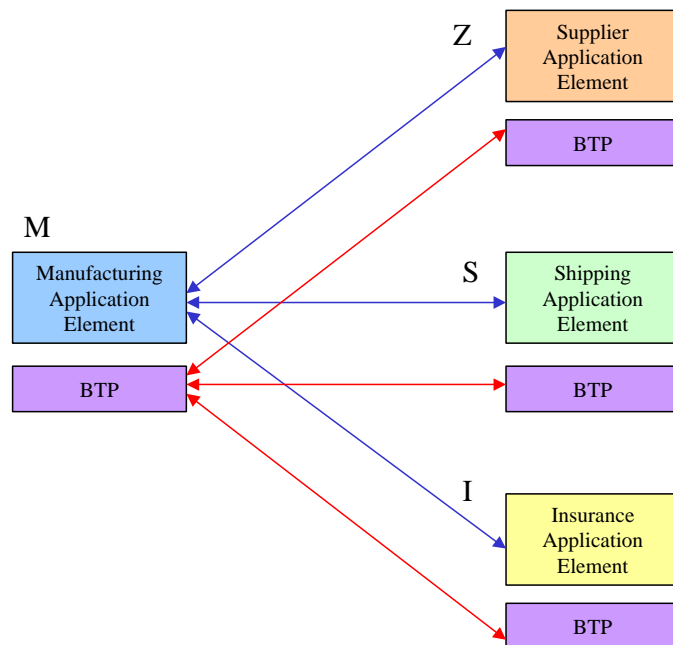
469

470

471

472

473



474

475

Figure 5-1 – Manufacturer Example

## 476 5.1.1 Business transactions

477 A **Business Transaction** can be defined as a consistent change in the state of a business  
478 relationship between two or more **parties**. A business relationship is any distributed state held by  
479 the parties which is subject to contractual constraints agreed by those parties. For example, an  
480 master purchasing agreement, which permits the placing of orders for components by known  
481 buying organizations allows a buyer and a seller to create and subsequently exchange meaningful  
482 information about the creation and processing of an order. Such agreements (and the consequent  
483 specification of shared or canonical data formats and of the messages that carry those formats,  
484 and their permitted sequences, all of which are needed for an automated implementation of an  
485 agreement) stem from business negotiations and are specific to a particular trading or information  
486 exchange community (group of potential parties). This definition of a business relationship is  
487 deliberately silent on the nature of the “business” transacted between the parties: it might be  
488 trading for profit, verification of authorizations for expenditure or loans, consistent publication  
489 (replication) of government ordinances to multiple sites, or any other computerized interaction  
490 where the parties require high confidence of consistent delivery or processing of data. In each  
491 party or site where business relationship state resides an **Application System** must exist which  
492 can maintain that state and communicate it as needed to other parties. The **Business Transaction**  
493 **Protocol** (BTP) assists the Application Systems of the various parties to bring about consistent  
494 and coordinated changes in the relationship as viewed from each party. BTP assumes that for a  
495 given Business Transaction, state changes occur, or are desired, in computer systems controlled  
496 by some set of parties, and that these changes are related in some application-defined manner.  
497 BTP assumes that the parties involved in a Business Transaction have distinct and autonomous  
498 Application Systems, which do not require knowledge of each others’ implementation or internal  
499 state representations in volatile or persistent storage. Access to such loosely coupled Application  
500 Systems is assumed to occur only through service interfaces.

501 Thus the state changes that BTP is concerned with are only those affecting the immediate  
502 business relationship. Although these externally visible changes will typically correspond to  
503 internal state changes of the parties, use of BTP does not itself imply any constraints or  
504 requirements on the internal state.<sup>1</sup>

## 505 5.1.2 External Effects

506 BTP coordinates the state changes caused by the exchange of **Application Messages**. These state  
507 changes are part of the **Contract** between BTP-using parties. In the manufacturing example, an  
508 interaction between the manufacturer and the supplier might involve the supplier receiving the  
509 order (an Application Message), checking to ensure that it had enough product on hand, reserving  
510 the product in the manufacturer’s name and replying. When the manufacturer agrees to the  
511 purchase (assuming the shipping and insurance are also reserved), BTP messages are sent to  
512 confirm the purchase. In this case, the supplier is offering a **BTP-enabled service** – the  
513 Application Element and its supporting BTP Elements together offer this service.

---

<sup>1</sup> 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.

514 In general, to be able to satisfy such contracts a BTP-enabled **service** must support in some  
 515 manner provisional or tentative state changes (the transaction's **Provisional Effect**) and  
 516 completion either through confirmation (**Final Effect**) or cancellation (**Counter-effect**). The  
 517 meaning of provisional, final, and Counter-effect are specific to the application and to the  
 518 implementation of the application. In the example, the reservation of the order is the Provisional  
 519 Effect, the completion of the purchase is the Final Effect.

520 Some of the implementation approaches are shown in Table 1. From the perspective of BTP and  
 521 the initiator application, all these are considered equivalent. Outside of BTP the underlying  
 522 business relationship (or Contract) between the parties can constrain the degree to which the  
 523 effects are visible.

524 **Table 1 Some alternatives for Provisional, Final and Counter-Effects**

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

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

528 Especially in the compensation approach, if the changes are cancelled, the Counter-effect may be  
 529 a precise inversion or removal of provisional changes, or it may be the processing of operations  
 530 that in some way compensate for, make good, alleviate or supplement their effect. There may be  
 531 side-effects of various kinds from a Counter-effected operation – such as levying of cancellation  
 532 charges or the record of the operation may be visible, but marked as cancelled. The possibility of  
 533 these side-effects is considered to be part of the overarching Contract.

### 534 **5.1.3 Two-phase outcome**

535 The BTP protocol coordinates the transitions into and out of the event states described above by  
 536 sending messages between the transaction parties. This involves a two-phase exchange. First the  
 537 Application Elements exchange messages that determine the characteristics and cause the  
 538 performance of the Provisional Effect; then a separate message, to the BTP Element, asking for  
 539 the performance of the final or the counter effect.

540 In general, the Application Elements in the systems involved having first communicated the  
541 Application Messages, each system that has to make changes in its own state:

- 542 • determines whether it is able achieve its Provisional Effect and then ensure it will be  
543 able either to **Cancel** (Counter-effect) its operation or to **Confirm** (give Final Effect to)  
544 its operation, whichever is subsequently instructed, and
- 545 • reports its ability to Confirm-or-cancel (its preparedness) to a central coordinating  
546 entity.

547 And, after receiving these reports, the coordinating entity:

- 548 • determines which of the systems should be instructed to Confirm and which should be  
549 instructed to Cancel
- 550 • informs each system whether it should Confirm or Cancel (the “outcome”).by sending  
551 a message to its BTP Element

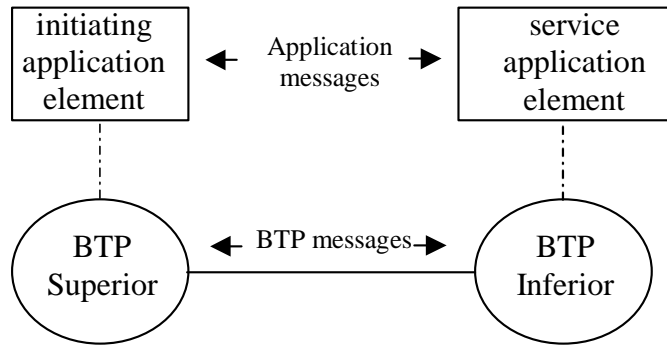
552 When there is more than one system that has to make changes such a two-phase exchange  
553 mediated by a coordinator is required to achieve a consistent outcome for a set of operations.  
554 The two-phases of the BTP protocol ensure that either the entire attempted transaction is  
555 abandoned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

#### 556 5.1.4 Actors and roles

557 BTP centres on the bilateral relationship between the computer systems of the coordinating entity  
558 and those of one of the parties in the overall Business Transaction. For each bilateral relationship  
559 in a Business Transaction, a software agent within the coordinating entity’s systems plays the  
560 BTP Role of Superior and a software agent within the systems of the party play the BTP Role of  
561 Inferior. The concept “**Role**” refers strictly to the participation in a particular relationship in a  
562 particular Business Transaction. The software agent performing a Role is termed an **Actor**. An  
563 Actor is distinguished from other Actors by being distinguishably addressable. The same Actor  
564 may perform multiple roles in the same Business Transaction (including the case where a  
565 Superior is also an Inferior), and may also perform the same or different roles in multiple  
566 Business Transactions, either concurrently or consecutively.

#### 567 5.1.5 Superior:Inferior relationship

568 A basic case of a single Superior:Inferior relationship, including the association with Application  
569 Elements, is illustrated in Figure 5-2. In many cases, including the manufacturer supply example,  
570 the Application Element associated with the superior will directly initiate the application  
571 exchanges –as does the manufacturer’s application **Client** to the supplier’s server, for example –  
572 but this is not invariably the case. It is possible that the first direct communication between the  
573 Application Elements is from one associated with an inferior to the one associated with the  
574 superior – for example, with an application that requested quotes by advertising the identity and  
575 location of the Superior along with invitation to quote; incoming quotes would be the first direct  
576 Application Message exchanged. In all cases the topmost Application Element in a tree or subtree  
577 will be aware of the Business Transaction first. How the identity of the transaction and the  
578 address of the BTP Superior are communicated to the secondary Application Element is a matter  
579 for the **Application Protocol** and not strictly part of BTP, although it will commonly be done by  
580 associating a BTP CONTEXT message with Application Messages..



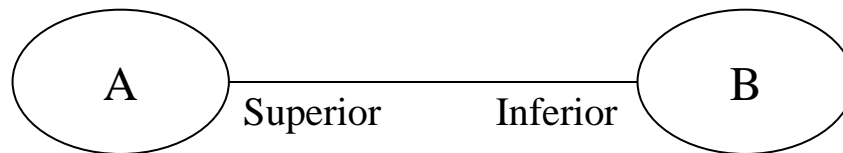
581  
582

**Figure 5-2 Basic Superior:Inferior relationship for BTP**

583 An Inferior is associated with some set of application activities that create effects within the  
 584 party, for a given Business Transaction. As stated above, commonly, though not invariably, this  
 585 application activity within the party will be a result of some operation invocations from elsewhere  
 586 (shown as the “initiating Application Element” in Figure 5-2), associated with the Superior to an  
 587 Application Element associated with the Inferior (shown as “Service Application Element”). This  
 588 second Application Element determines what activities the Inferior is responsible for, and then  
 589 the Inferior is responsible for reporting to the Superior whether the associated operations’  
 590 Provisional Effect can be confirmed/cancelled – this is called “becoming prepared”, because the  
 591 Inferior has to remain prepared to receive whichever order eventually arrives (subject to various  
 592 exceptions and exclusions, detailed below).

593 **5.1.6 Business Transaction Trees**

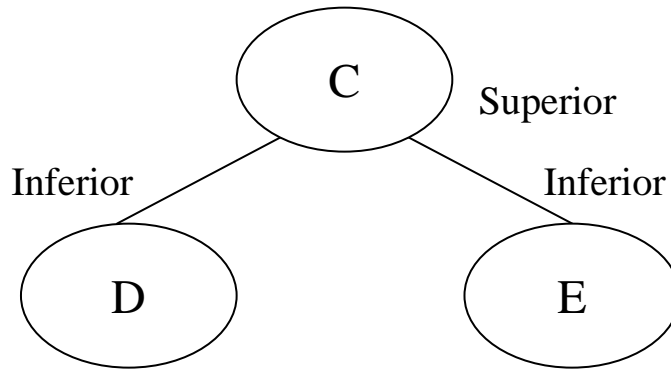
594 There are many patterns in which the service provider participants involved in a Business  
 595 Transaction may be arranged in respect of the two-phase exchange and the determination of  
 596 which are eventually confirmed. The simplest is shown in Figure 5-3 involving only two parties –  
 597 one (B) making itself subject to the decision of Confirm-or-Cancel made by the other (A). This  
 598 basic bilateral relationship, in which one side makes itself inferior to the other, is the building  
 599 block used in all Business Transaction patterns. In this simplest case, the “coordination” by the  
 600 superior, A, is just that A can be sure whether the operations at the inferior, B were eventually  
 601 cancelled or confirmed.



602  
603

**Figure 5-3 Simple two-party Business Transaction**

604 In the next simplest case, as in Figure 5-4, a bilateral, Superior:Inferior relationship appears  
 605 twice, with two Inferiors, D and E, both making themselves inferior to a single Superior, C. From  
 606 the perspective of either D or E, they are in the same position as B in the previous case –they are  
 607 unaware of and unaffected (directly) by each other. It is only within C that there is any linkage  
 608 between the Confirm-or-Cancel outcomes that apply to D and E.

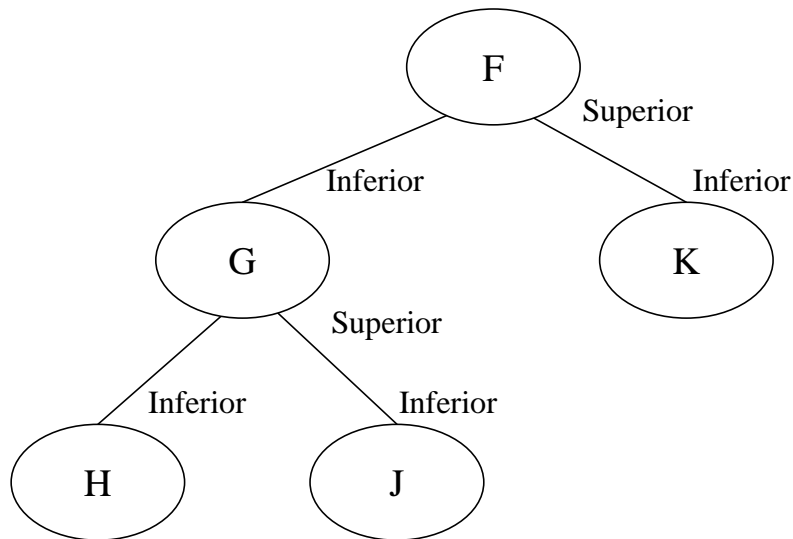


609

610

**Figure 5-4 Business Transaction with two inferiors**

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



618

619

**Figure 5-5 Business Transaction with an Intermediate (interposition)**

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

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

## 630 5.1.7 Atoms and Cohesions

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

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

- 641 a) the Application Element initiated the Business Transaction (causing the creation  
642 of the Superior), and instructed that all Inferiors of the Superior should Confirm  
643 or all should Cancel; the Superior is an **Atom Coordinator**;
- 644 b) the Application Element initiated the Business Transaction, but deferred the  
645 choice of which Inferiors should Confirm until later, allowing it (the Application  
646 Element) to choose some subset to be confirmed, others to Cancel; the Superior  
647 is a **Cohesion Composer**;
- 648 c) the Application Element was itself involved in an existing Business Transaction,  
649 and the Superior in this relationship is the Inferior in another one; this  
650 Application Element instructed that all Inferiors of this Superior should Confirm,  
651 but only if confirmation is instructed from above or all should Cancel; the  
652 Superior is an (atomic) **Sub-coordinator**;
- 653 d) the Application Element was itself involved in an existing Business Transaction,  
654 and the Superior in this relationship is the Inferior in another one; this  
655 Application Element deferred the choice of which Inferiors should be candidates  
656 to Confirm until later, allowing it (the Application Element) to choose some  
657 subset to be confirmed, given that confirmation is instructed from above, others  
658 to Cancel; the Superior is a (cohesive) **Sub-composer**.

659 In the atomic case, the two-phase outcome exchange means a Superior acting as an atomic  
660 Coordinator or sub-coordinator will treat any Inferior which cannot prepare to Cancel/Confirm as  
661 having veto power, causing the Superior to instruct all its Inferiors to Cancel. A Business  
662 Transaction whose topmost Superior is atomic is an **Atomic Business Transaction**, or **Atom** –  
663 the superior is the Atom Coordinator.

664 In the cohesion case, with the Superior acting as a cohesive Composer or Sub-Composer, the  
665 controlling Application Element will determine the implications of an Inferior's failure to be  
666 prepared to Confirm-or-Cancel; the Application Element may Cancel some or all other Inferiors,  
667 do other application work, which may involve new Inferiors or may just accept the cancellation of  
668 that one Inferior and carry on. A Business Transaction whose topmost Superior is cohesive is a  
669 **Cohesive Business Transaction**, or **Cohesion** – the Superior is the Cohesion Composer.

670 For a Cohesion, the set of Inferiors that eventually Confirm is called the **Confirm-set**. The term is  
671 also used to mean the set of Inferiors that have been chosen to (potentially) Confirm before the  
672 final outcome is decided – if the Cohesion is eventually cancelled, then Confirm-set cancels. (See  
673 section “5.2.7 Evolution of Confirm-set”). The Confirm-set of an Atom is all of the Inferiors.

674 If the Superior is itself an Inferior, its own action of becoming prepared, and reporting this to its  
675 own Superior will depend on the receipt of prepared reports from its Inferiors. If it is atomic (i.e.  
676 is a sub-coordinator), it will only **Become Prepared** if all Inferiors reported preparedness to it; if  
677 it is cohesive (i.e. is a sub-composer), the controlling Application Element will determine whether  
678 the set of Inferiors that have reported as prepared is sufficient.

679 If the Superior is not an Inferior, the determination of when, if and, for a Cohesion, what it should  
680 Confirm depends on the controlling application. This “top-most” Superior has a different  
681 relationship to the controlling application to that of an Inferior to its Superior: an Inferior reports  
682 that it is prepared to the Superior, which instructs it whether to Cancel or to Confirm; the top-  
683 most Superior is asked by the Application Element to attempt to Confirm, but, dependent on the  
684 preparedness of its Inferiors, the top-most Superior makes the final decision. Consequently the  
685 top-most Superior is termed the **Decider**; the Application Element that asks it to Confirm is the  
686 **Terminator**.

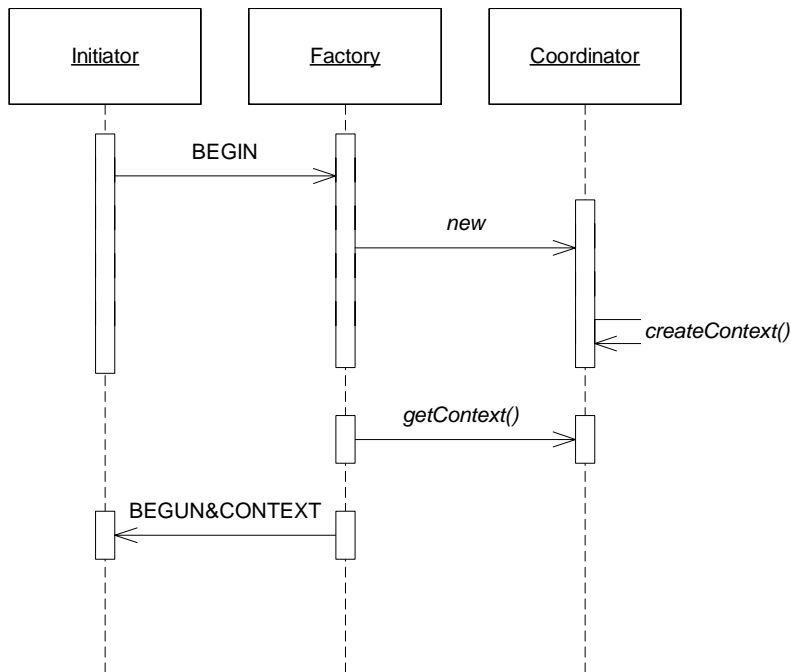
#### 687 **5.1.8 Participants, Sub-Coordinators and Sub-Composers**

688 An Inferior may directly be responsible for applying the Confirm-or-Cancel decision to some  
689 application effects, or may in turn be a BTP Superior to which others will enrol. If it only handles  
690 application effects it is called a **Participant**, in the latter case it is called a **Sub-coordinator** or a  
691 **Sub-composer**, depending on whether it is atomic or cohesive with respect to its own future  
692 Inferiors. (If an Inferior is both responsible for application effects, and is a BTP Superior, it is not  
693 considered a Participant, according to the strict definitions, though informally it may be referred  
694 to as such.) The Superior is unaware, via the BTP exchanges, whether the Inferior is a Participant,  
695 Sub-coordinator or Sub-composer. This specification does not define messages or interfaces for  
696 the creation of Participants or for the Application Element to tell the Participant what the  
697 application effects are or how they are to be confirmed or cancelled as necessary. (Although out-  
698 of-scope for this specification, one or more APIs could be standardised.)

### 699 **5.2 Business transaction lifecycle**

#### 700 **5.2.1 Business Transaction creation**

701 This section describes in some detail how a BTP Business Transaction is created. The interaction  
702 diagram in Figure 5-6 also shows this sequence. The messages shown in lower-case italics  
703 (between Factory and Coordinator) represent interactions that are not specified in BTP.



**Figure 5-6 – Creation of a Business Transaction**

704

705

706 A Business Transaction is started at the initiative of an Application Element, which causes the  
 707 creation of a Coordinator or Composer. Any Inferiors participating in this transaction will enrol  
 708 with this Superior. BTP defines abstract messages (BEGIN, BEGUN) to request this but the  
 709 equivalent function can also be achieved using proprietary means, especially if the Factory or  
 710 Coordinator is an internal component of the initiating application. If the BTP messages are used,  
 711 the Application Element performs the Role of Initiator and sends BEGIN to a Factory. The  
 712 BEGIN message identifies whether a Coordinator (for an Atom) or a Composer (for a Cohesion)  
 713 is desired. The Factory, after the creation of the new Coordinator or Composer, replies with  
 714 related BEGUN and CONTEXT messages. "Related" means they are sent together in a manner  
 715 that has semantic significance; how this is represented is determined by the binding in use. The  
 716 Coordinator's or Composer's creation is the establishment of a new instance of a BTP Role. It  
 717 may involve only the assignment of a new identifier within an existing Actor (which may also be  
 718 performing the Factory Role, for example). Alternatively a new Actor with a distinct address  
 719 may be instantiated. These and other alternatives are implementation choices, and BTP ensures  
 720 other Actors are unaffected by the choice made.

721 The BEGUN message provides the addressing and identification information needed for a  
 722 Terminator to access the new Coordinator or Composer as Decider; the Application Element  
 723 performing the Initiator Role may itself act as Terminator, or may pass this information to some  
 724 other Application Element.

725 Whether this interoperable BTP Initiator:Factory relationship or some other mechanism is used to  
 726 initiate the Business Transaction, a CONTEXT is made available. This identifies the Coordinator  
 727 or Composer as a Superior – containing both addressing information and the identification of the  
 728 relevant state information. The CONTEXT is also marked as to whether or not this Superior will  
 729 behave atomically with respect to its Inferiors (i.e. is it a Coordinator or Composer).

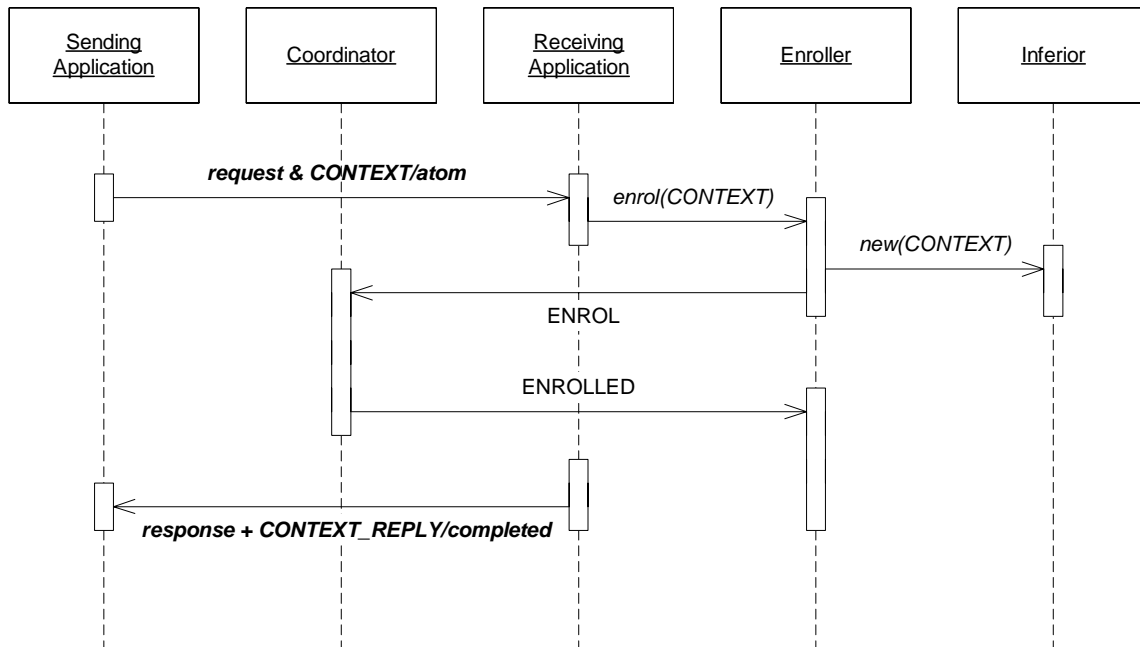
## 730 5.2.2 Business Transaction propagation

731 The propagation of the Business Transaction from one party to another, to establish the  
732 Superior:Inferior relationships involves the transmission of the CONTEXT. This is commonly in  
733 association with, or related to, one or more Application Messages between the parties. In a typical  
734 case, an Application Message is sent from the Application Element that performed the Initiator  
735 Role (the “sending application” in Figure 5-2) to some other element (the receiving application).  
736 The CONTEXT is sent with the Application Message in such a way that the Application  
737 Elements understand that work performed as a result of the Application Message is to be the  
738 subject of a Confirm-or-Cancel decision of the Superior.<sup>2</sup> The receiving Application Element  
739 causes the creation of an Inferior (which, as for the Superior may involve just assignment on a  
740 new identifier, or instantiation of an new Actor) and ensures the new Inferior is enrolled with the  
741 Superior identified in the received CONTEXT, using an ENROL message sent to the Superior  
742 using the address in that CONTEXT.

743 Figure 5-7 shows a sequence diagram of the propagation of a Business Transaction. It is assumed  
744 the transaction has already been created, and thus the Application Element and Coordinator exist.  
745 The diagram shows the Enroller as a distinct Role, with non-standardised interactions between the  
746 Application Element, the Enroller and the new Inferior. The Enroller Role may in fact be  
747 performed by the Application Element, by the Inferior or by a distinct entity. At least the  
748 Superior-identifier and Superior-address from the CONTEXT has to be passed the Enroller and to  
749 the Inferior so they can communicate with the Coordinator (whose identifier and address these  
750 are).

---

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

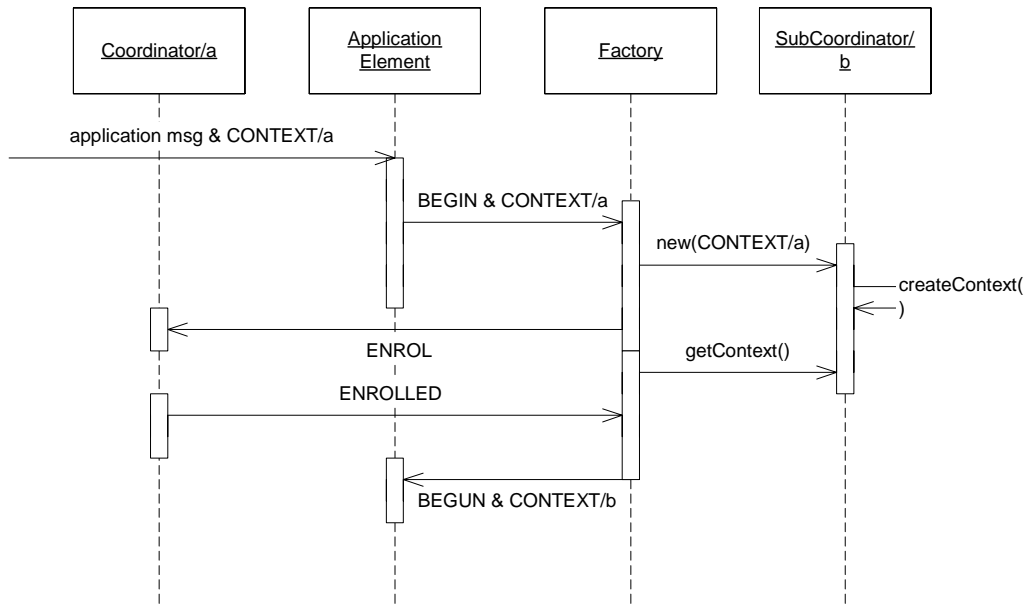


751  
752

**Figure 5-7 Sequence diagram of propagation**

753 **5.2.3 Creation of Intermediates (Sub-Coordinators and Sub-Composers)**

754 If the new Inferior is to be a Sub-coordinator or Sub-composer, this can be created using a non-  
 755 standard mechanism or the Initiator:Factory relationship can be used again. Figure 5-8 shows a  
 756 sequence diagram, using the latter mechanism. The Application Element, having received an  
 757 Application Message and a CONTEXT from some Superior – shown as a Coordinator/a in the  
 758 diagram - wants to create the new Inferior and acting in the Initiator Role, issues BEGIN to the  
 759 Factory, but the CONTEXT for the original Superior (Coordinator/a) is “related” to the BEGIN.  
 760 The Factory is responsible for enrolling the new Sub-coordinator or Sub-composer as an Inferior  
 761 of the Superior identified by the received CONTEXT. The reply from the Factory is a related  
 762 BEGUN and CONTEXT – this being the CONTEXT for the new Sub-coordinator (‘b’) or Sub-  
 763 sub-composer as a Superior. The Sub-coordinator/Sub-composer is not a Decider, as its decision is  
 764 subordinated to the outcome received from the Superior. For a Sub-coordinator, further control by  
 765 the application is primarily a matter of relating the new CONTEXT to appropriate application  
 766 activity. For a Sub-composer, there is also a requirement for the application to determine which of  
 767 the Inferiors of the Sub-composer must have reported they are prepared before the Sub-composer  
 768 can report that it is itself prepared to its own Superior, and then which of these Inferiors are to be  
 769 ordered to Confirm if the Sub-composer is ordered to Confirm. This specification does not  
 770 provide an interface or interoperable message to control this; like the relationship between  
 771 Application Element and Participant, it is left to the implementation or independent  
 772 standardisation.



773

774

**Figure 5-8 – Creation of a Sub-coordinator**

775 The creation of a new Inferior and establishment of a Superior:Inferior relationship does not  
 776 always imply that the BTP Actors are under the control of different business parties or  
 777 Application Elements. In particular, an Application Element may begin a Cohesion, then create  
 778 and enrol (atomic) Sub-coordinators as Inferiors of the Composer, then associate a different Sub-  
 779 coordinator's CONTEXT with each of several aspects of the application work, transmitting that  
 780 CONTEXT with the Application Messages for that aspect to the other parties in the Business  
 781 Transaction. Those parties can then create Participants (or other Inferiors) that are enrolled with  
 782 the appropriate Sub-coordinator. Later, the Application Element (as Terminator, or its equivalent)  
 783 can choose which of the Cohesion Composers' Inferiors to Cancel and which to Confirm. By  
 784 interposing its own atomic Sub-coordinator the initiating Application Element can indicate to the  
 785 other parties that some associated set of application work will be confirmed or cancelled as a unit.  
 786 This may allow the receiving parties to share information between **Application Operations** and  
 787 to make one Participant responsible for applying the outcome to several operations.

#### 788 5.2.4 "Checking" and context-reply

789 In BTP, enrolment is at the initiative of an Application Element that has received or has access to  
 790 the CONTEXT which creates an Inferior (BTP uses a "pull" paradigm for enrolment). An  
 791 Application Element in possession of a CONTEXT can choose, perhaps constrained by an  
 792 overarching business and application understanding, whether and how many Inferiors to create  
 793 and enrol. Consequently, in general, an Application Element which propagates a CONTEXT to  
 794 another (via whatever mechanisms it choose), cannot be sure how many Inferiors will be enrolled  
 795 as a result. Without further controls, there would be a possibility that an Application Element  
 796 receiving a CONTEXT might attempt to enrol an Inferior with a Superior after the Superior had  
 797 been asked to Confirm, or even had completed confirmation. In such a case application work that  
 798 should have been part of a confirmed Atomic Business Transaction could be cancelled, violating  
 799 the atomicity in a manner that will not be apparent to the application.

800 To avoid this, whenever a CONTEXT is transmitted to another party by or on behalf of the  
801 application, the transmission of the CONTEXT itself can be replied to with a  
802 CONTEXT\_REPLY message – this is required for an Atom, allowed for a Cohesion. An  
803 Application Element that has received a BTP CONTEXT is able, because it knows the Superior’s  
804 identification and address in the CONTEXT, to enrol Inferiors (Figure 5-9).<sup>3</sup> Replying with  
805 CONTEXT\_REPLY means that the sender (the earlier receiver of a CONTEXT) will not enrol  
806 any more Inferiors. Consequently the sender of a CONTEXT can keep track of whether there are  
807 any outstanding (un-replied to) CONTEXTs that could be used for an enrolment and can avoid  
808 requesting or permitting confirmation until everything is safe. This check is required for an Atom,  
809 but is not always essential when the CONTEXT is for a Cohesion. For a Cohesion, it is a matter  
810 for the controlling application whether all would-be Inferiors must be enrolled before a  
811 confirmation decision can be made; or whether it is acceptable to proceed to confirmation at some  
812 point in time with the already enrolled Inferiors (or a subset thereof), accepting the automatic  
813 cancellation of any late arrivals.

814 CONTEXT\_REPLY can also indicate that attempted enrollments failed. This can occur if the  
815 Enroller is unable to contact the Superior, but it able to return a CONTEXT\_REPLY to where-  
816 ever the CONTEXT came from.

## 817 5.2.5 Message sequence

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

- 820 • **Outcome Relationships** : the Superior:Inferior relationship (i.e. between BTP Actors  
821 within the Transaction Tree) and the Enroller:Superior relationship used in establishing  
822 it
- 823 • **Control Relationships** : the application:BTP Actor relationships that create the nodes  
824 of the Transaction Tree (Initiator:Factory) and drive the completion  
825 (Terminator:Decider).

826 The Outcome Relationships and the messages used in them an essential part of BTP. For the  
827 Control Relationships, it would be possible to achieve the same general function using non-  
828 standardised messages or API mechanisms. There are other distinguishable relationships between  
829 roles defined by BTP that are not standardised in this specification.

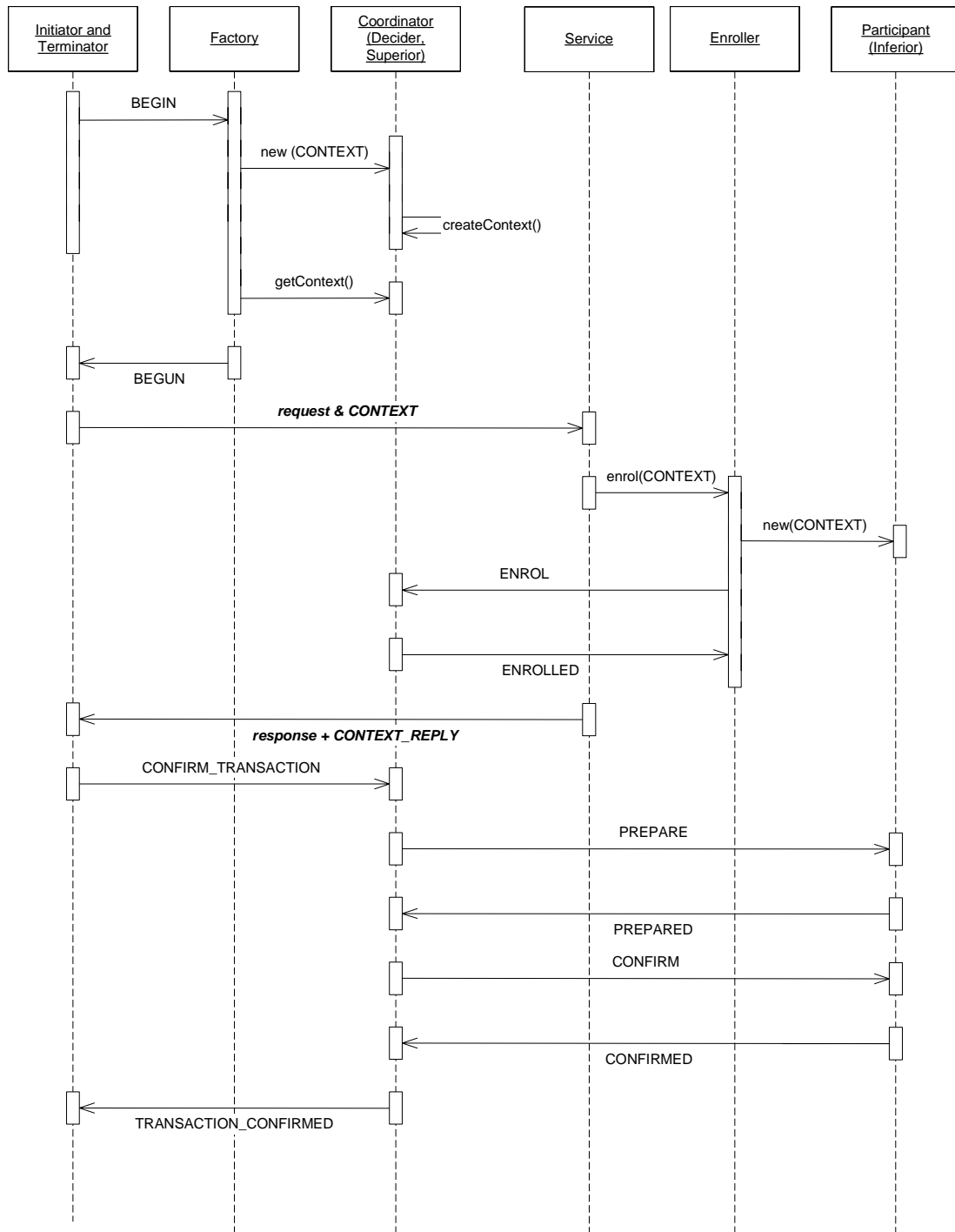
830 Figure 5-9 shows the message exchange for the conventional progression of a simple transaction  
831 to confirmation with a single Superior:Inferior relationship, assuming the standard Control  
832 Relationship. Two Application Elements using a request/response Application Message exchange  
833 are involved – the first is represented as the Initiator and Terminator, the second as the Service  
834 and Enroller. The Decider/Superior is shown as a Coordinator, but with only one Inferior there  
835 would be no difference with a Cohesion Composer. The Factory:Coordinator events are non-  
836 standardised, but represent interactions that must occur in some form. There are other interactions

---

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

837 between the various application groups – Initiator-Terminator and Participant-Enroller-Service  
838 that are not shown – in particular the Service:Participant relationship.

839 The message sequence is shown is the “conventional” sequence, with all messages explicitly  
840 present and sent separately. There are several variations and optimisations possible – these are  
841 discussed below.



842

843

**Figure 5-9 A conventional message sequence for a simple transaction**

844 Note that CONTEXT has a “related” (&) relationship to BEGUN and to the application request

845 (although in the latter case the meaning of this is defined by the application, not by BTP. The

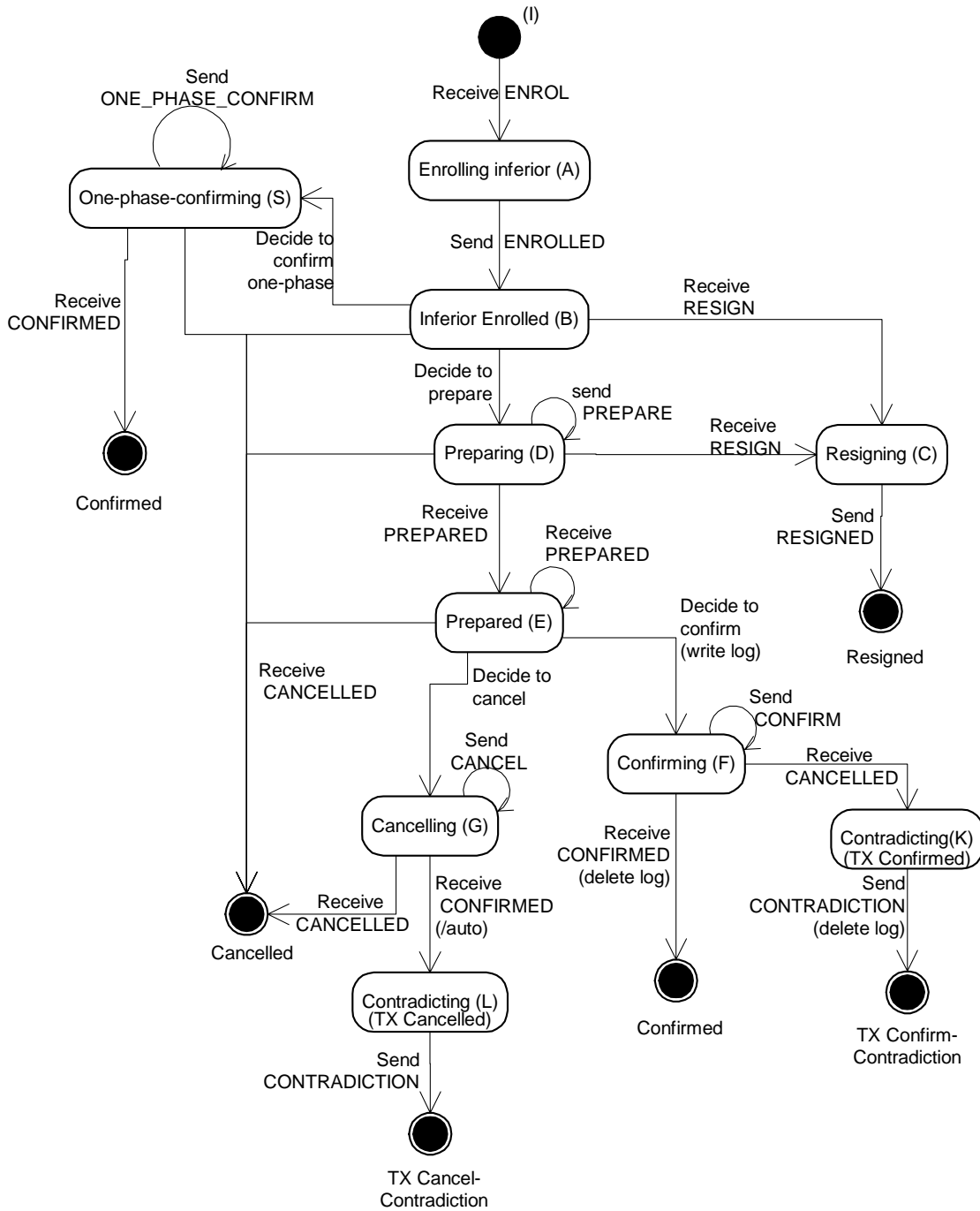
846 response + CONTEXT\_REPLY has no semantic significance, and could be sent separately;

847 provided the CONTEXT\_REPLY is not sent until the ENROLLED has returned.

848 The progression of a single instance of the central outcome (Superior:Inferior) relationship can  
849 also be presented as a set of state transitions. The normative part of the specification includes  
850 state tables for the Superior side of such a relationship and for the Inferior. Since a single  
851 Superior (Coordinator, Composer, Sub-coordinator, Sub-composer) can have multiple Inferiors,  
852 each Superior will have multiple instances of the “Superior state”. How these link together is  
853 discussed below in the section “5.2.7 Evolution of Confirm-set”, but the state transitions for  
854 the individual Superior:Inferior relationships include “decision events” which constrain the  
855 behaviour of the **Business Transaction Tree Node** as a whole, and thus define the semantics of  
856 the BTP messages.

857 The normative state tables distinguish some states that differ only in which messages can be  
858 received and thus allow for a level of error checking. The progress of the Outcome Relationship  
859 can be followed without dropping to such a detailed level, and the state diagrams shown here  
860 aggregate some of the states that are distinguished in the state tables. The single letters in  
861 parentheses in the diagrams correspond to the state names used in the tables. For simplicity, the  
862 state diagrams do not include the events leading to the sending of a HAZARD message – the  
863 detection and recording of a “problem” – meaning that the Inferior is unable to cleanly Confirm  
864 or cleanly Cancel the operations it is responsible for. As is specified in the state tables, such a  
865 problem can be detected in most states, and reported with a HAZARD message.

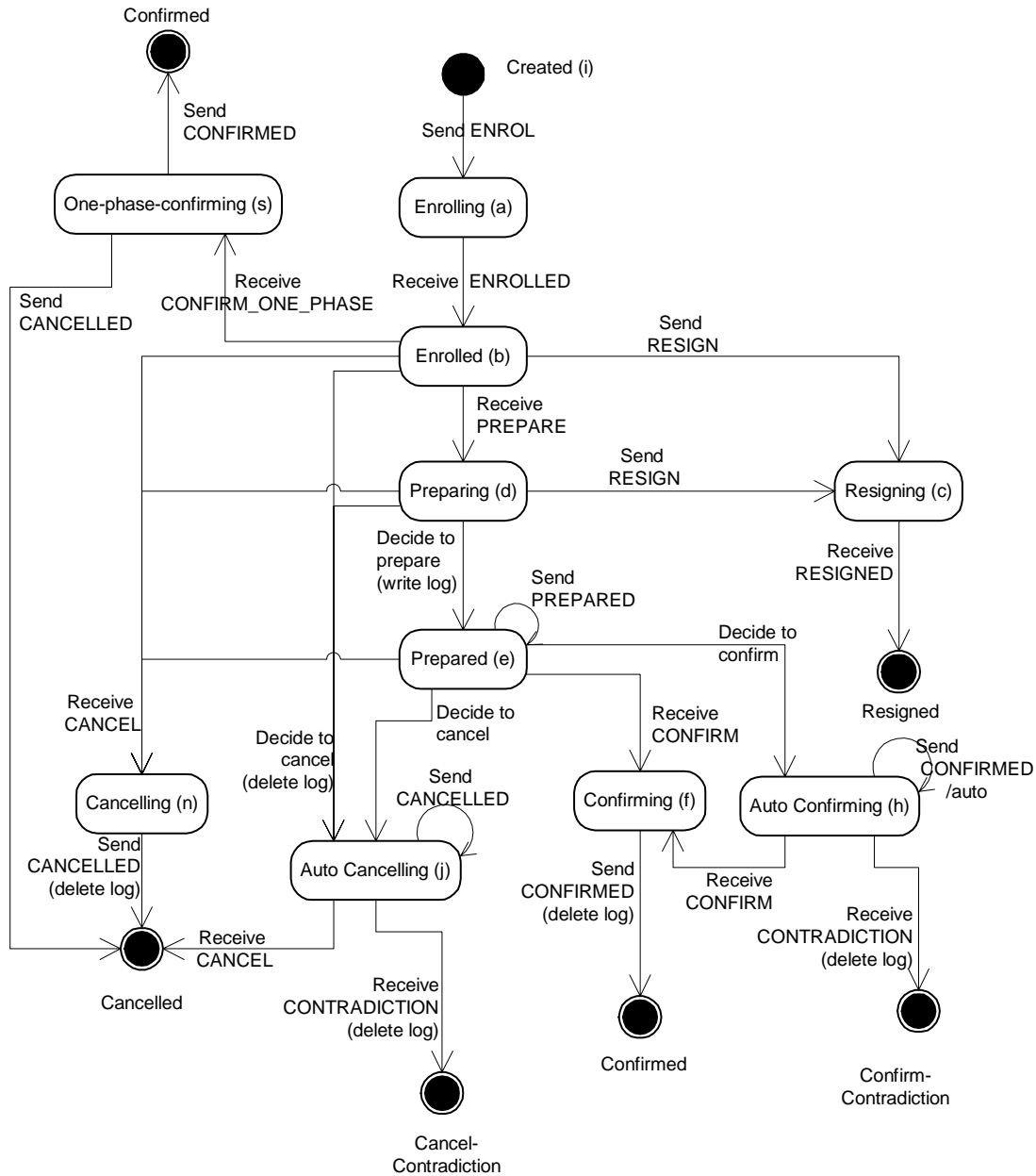
866 It should be noted that, with some exceptions, the transmission of a message **from** a Superior or  
867 Inferior does not cause a state change at that side. State changes are normally caused either by the  
868 receipt of a message from the **Peer**, or by a “decision event” – which may be an internal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



876

877

**Figure 5-10 State diagram for Superior side of a Superior:Inferior relationship**

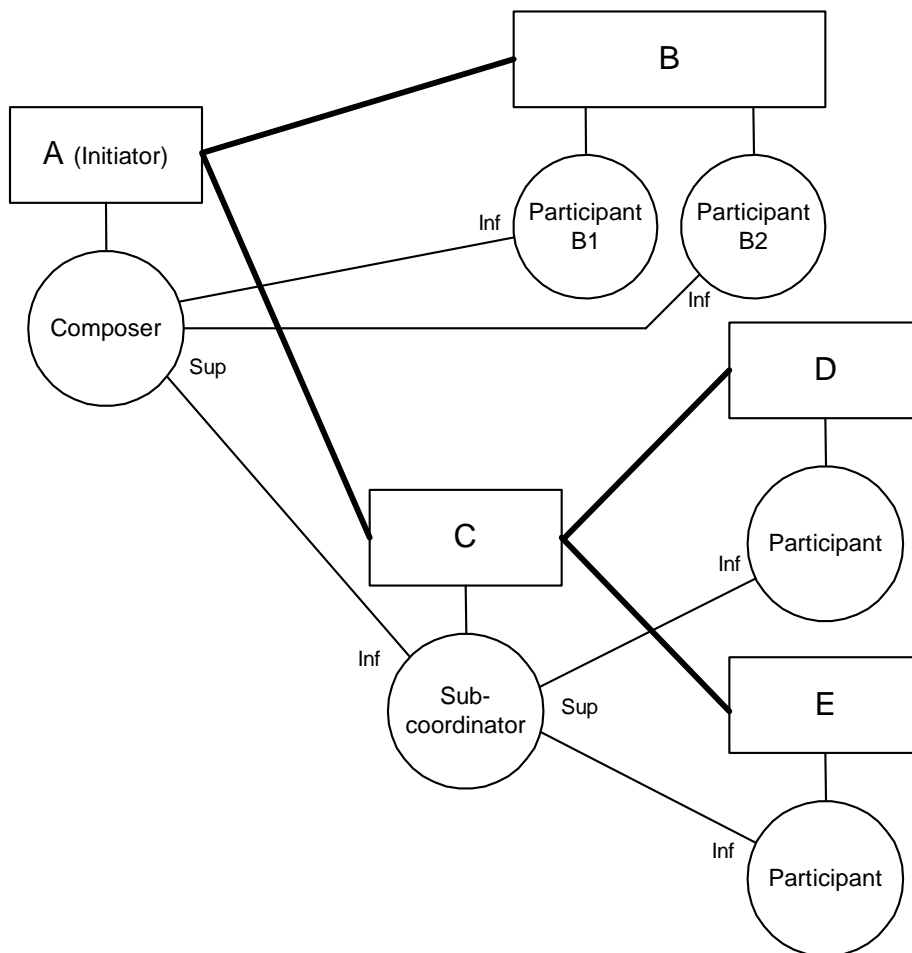


878

879 **Figure 5-11 State diagram for Inferior side of Superior:Inferior relationship**

880 **5.2.6 Control of inferiors**

881 In the case as shown in Figure 5-12, where the CONTEXT has been propagated from one  
 882 Application Element (A) to others (B, C, and from C to D,E), the determination of whether to  
 883 create and enrol Inferiors is, in general, up to the receiving Application Element – this is an aspect  
 884 of the fundamental autonomy of the parties involved in a Business Transaction. This autonomy  
 885 may be constrained in particular situations, by inter-party agreement or where the Application  
 886 Elements are in fact under common control.



887

888

**Figure 5-12 Transaction Tree showing various application:Participant relationships**

889 The relationship between the Application Messages and either the propagated CONTEXT or the  
 890 ENROL message(s) sent to the Superior is strictly part of the Application Protocol (or the  
 891 application-with-BTP combination protocol). However defined, this allows the Superior-side  
 892 Application Element to be aware of what application work will be confirmed or cancelled under  
 893 the control of an Inferior. However, from the perspective of the Superior, and the Application  
 894 Element controlling it, the Inferior is opaque – it is not in general possible for the Superior or its  
 895 controlling Application Element to determine whether an Inferior is a Sub-composer or Sub-  
 896 coordinator (i.e. has Inferiors of its own) or is a Participant, with no further BTP relationships.  
 897 Thus, if the Inferior is a Sub-composer or Sub-coordinator, the Superior has no visibility or  
 898 control of its “grand-children” – the Inferiors of its Inferior (thus, in Figure 5-12, the Composer at  
 899 A is unaware of D and E)

900 The opacity of an Inferior does not however apply to the control exercised by the immediately  
 901 controlling Application Element. An Application Element, acting as Terminator to a Decider (i.e.  
 902 to a Composer or Coordinator), can be aware of and distinguish the different Inferiors enrolled  
 903 with that Decider (i.e. Inferiors enrolled with the Decider in its Role as Superior). (E.g. in Figure  
 904 5-12, Application Element A knows of the Inferiors at C, B1 and B2) This is especially the case  
 905 for a Cohesion Composer, where the Terminator will be able to control which of the enrolled  
 906 Inferiors of the Composer are eventually confirmed – more exactly, the application will have

907 control of the Confirm-set for the Cohesion. For an Atom Coordinator, visibility of the Inferiors is  
908 useful but less important, since no selection can be made among which will be in the Confirm-set  
909 – for an Atom, all Inferiors are ipso facto members of the Confirm-set.

910 For this control of the Inferiors to be useful, the Terminator Application Element will need to be  
911 able to associate particular parts of the application work with each Inferior. In a traditional  
912 transaction system, users do not need to see participants, but they see services or objects. What  
913 participants are enlisted with a transaction on behalf of those services and objects is not really of  
914 interest to the user. When it comes to commit or rollback the transaction, it acts on the transaction  
915 and not on the individual participants.

916 In BTP that is still the case if we work purely with atoms. While an Atomic Coordinator knows  
917 its participants it cannot pick and choose among them. In contrast, a Cohesive Terminator must  
918 have significant, detailed knowledge and visibility of both the identities of its inferiors and  
919 association of parts of the application work with each Inferior. The user must be able to identify  
920 which participants to cancel/prepare/confirm. This identification can be achieved by various  
921 means. Taking the case of an Application Element controlling a Cohesion Composer:

922 a) The Application Element can create an Atom Sub-coordinator as an immediate  
923 Inferior of the Cohesion Composer and propagate the Sub-coordinator's CONTEXT  
924 associated with Application Messages concerned with the particular part of the  
925 application work; any Inferiors (however many there may be) enrolled with Sub-  
926 coordinator can be assumed to be responsible for (some of) that part of the  
927 application, and the Terminator Application Element can just deal with the immediate  
928 Inferior of the Composer that it created.

929 b) The Application Element can propagate the Composer's own CONTEXT, and the  
930 receiving Application Element can create its own Inferior (or Inferiors) which will be  
931 responsible for some part of the application, and send ENROL(s) to the Composer (as  
932 Superior). Application Messages concerned with that part of the application are  
933 associated, directly or indirectly, with each ENROL, and the Terminator Application  
934 Element can thus determine what each Inferior is responsible for.

935 In both cases, the means by which the Application Message and the BTP CONTEXT or ENROL  
936 are associated are ultimately application-specific, and there are several ways this can be done.

937 • At the abstract message level, BTP defines the concept of transmitting “related” BTP and  
938 Application Messages – particular bindings to Carrier Protocols can specify interoperable  
939 ways to represent this relatedness (e.g. the BTP message can be in a “header” field of the  
940 Carrier Protocol, the Application Message in the body).

941 • An Application Message may contain fields that identify or point to the BTP message  
942 (e.g. the “inferior-identifier” from the ENROL may be a field of the Application  
943 Message).

944 • BTP messages, including CONTEXT and ENROL, can carry “qualifiers” – extension  
945 fields that are not core parts of BTP or are not defined by BTP at all. The standard  
946 qualifier “inferior-name” or application-specific qualifiers can be used to associate  
947 application information and the BTP message. The qualifiers received from the Inferiors

948 on ENROL are visible to the Terminator application on the INFERIOR\_STATUSES  
949 message. The application design will need to ensure that the Terminator can determine  
950 which parts of the application work are associated with each Inferior.

951 *NOTE -- For example, a service receiving an invocation associated with a Cohesion*  
952 *CONTEXT, but where the application design meant that there would be no more*  
953 *than one Inferior enrolled as a result of that invocation, could be required to include*  
954 *information identifying the service and the invocation in the “inferior-name”*  
955 *qualifier on the consequent ENROL. These qualifiers would be visible to the*  
956 *Terminator on INFERIOR\_STATUSES, allowing the Terminator to determine which*  
957 *“inferior-identifiers” to include in the “inferiors-list” parameter of the*  
958 *CONFIRM\_TRANSACTION which defines which Inferiors are to be confirmed.*  
959 *Among other alternatives, the “inferior-identifier” itself could be a field of the*  
960 *application response – this would also be applicable where there could be multiple*  
961 *Inferiors enrolled as a consequence of one invocation for the Terminator to choose*  
962 *between.*

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

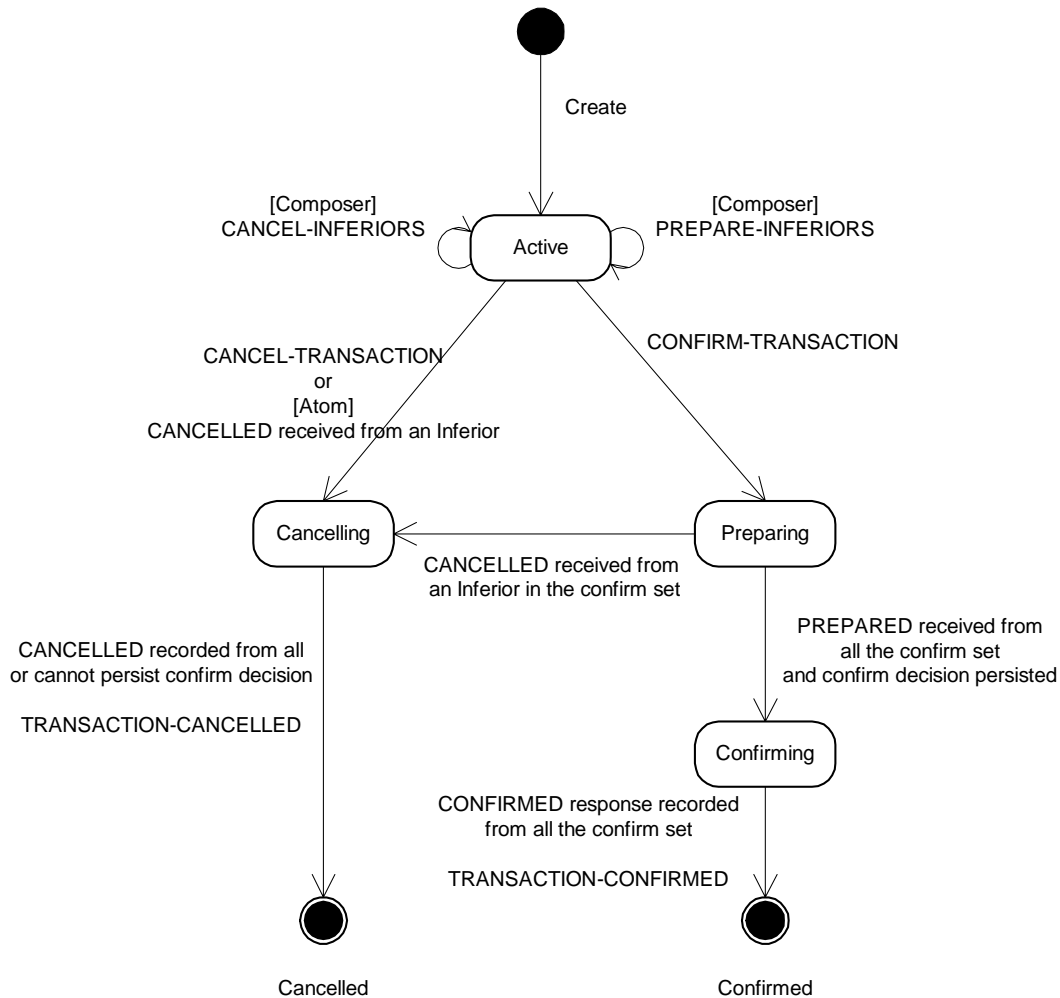
### 965 **5.2.7 Evolution of Confirm-set**

966 As mentioned above, the set of Inferiors of a Cohesion that will eventually Confirm is called the  
967 Confirm-set. The determination of the Confirm-set is made by the controlling application, but is  
968 affected by events from the Inferiors themselves. If the standard Control Relationship is used, the  
969 control of the Cohesion Composer is expressed by the Terminator:Decider exchanges, and the  
970 progressive determination of the Confirm-set (its evolution) is effectively the event sequence for  
971 the Terminator:Decider relationship.

972 An Atom also has a Confirm-set, but this always includes all the Inferiors and so does not evolve  
973 in the same way as Cohesion’s. With some exceptions, the Terminator:Decider relationship is the  
974 same for Atom Coordinators as for Cohesion Composers; this section deals with both, noting the  
975 exceptions.

976 The event sequence for a Composer or Coordinator is summarised in the state diagram in Figure  
977 5-13. The step-by-step description refers to “Composer”, but should be read as referring to  
978 Coordinators as well, unless stated otherwise.

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



981

982

**Figure 5-13 State diagram for a Composer or Coordinator (i.e. Decider)**

983

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

984

985

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

986

987

- Inferiors may resign - RESIGN is received from an Inferior (see section 5.3.3 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).

988

989

990

991

- 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).

992

993

994

- PREPARED may be received; there is no immediate effect

995                   • The Terminator may issue PREPARE\_INFERIORS to the Composer (as Decider)  
996                   for some subset of the Inferiors; PREPARE is sent to each and any of the Inferiors  
997                   in the subset, excluding any from RESIGN, CANCELLED or PREPARED has been  
998                   received; the sending of PREPARE will induce the Inferiors to reply with  
999                   PREPARED, CANCELLED or RESIGN; when replies have been received from all,  
1000                  the Composer (as Decider) replies to the Terminator with INFERIOR\_STATUSES,  
1001                  reporting the replies received (which may in fact have been received before the  
1002                  PREPARE\_INFERIORS). PREPARE\_INFERIORS is not issued to Atom  
1003                  Coordinators.

1004                  • The Terminator may issue CANCEL\_INFERIORS to the Composer (as Decider) for  
1005                  some subset of the Inferiors; CANCEL is sent to each and any of the Inferiors in the  
1006                  subset, excluding any from RESIGN or CANCELLED has been received; the  
1007                  sending of CANCEL will normally induce the Inferiors to reply with CANCELLED  
1008                  – there are some exception cases; when replies have been received from all, the  
1009                  Composer (as Decider) replies to the Terminator with INFERIOR\_STATUSES,  
1010                  reporting the replies received. CANCEL\_INFERIORS is not issued to Atom  
1011                  Coordinators. CANCEL\_INFERIORS may be issued for an Inferior regardless of  
1012                  whether PREPARED has been received from it.

1013                  • The Terminator may issue REQUEST\_INFERIOR\_STATUSES to the Composer  
1014                  (as Decider) for all or some subset of the Inferiors; the Composer immediately  
1015                  replies with INFERIOR\_STATUSES, reporting the current state of the Inferiors as  
1016                  known to the Superior.

1017       Eventually, the Terminator issues one of the completion messages – CANCEL\_TRANSACTION  
1018       or CONFIRM\_TRANSACTION. These messages have a flag that determines whether the  
1019       Terminator wishes to be informed of contradictory and heuristic decisions or hazards within the  
1020       transaction – this affects when the reply from the Composer (as Decider) is sent to the  
1021       Terminator. (See section “5.3.5 Autonomous cancel, autonomous Confirm and contradictions”  
1022       for details on contradictory and heuristic cases).

1023       If the message is CANCEL\_TRANSACTION, CANCEL is sent to all Inferiors that it has not  
1024       already been sent to, and from which neither RESIGN or CANCELLED have been received. If  
1025       the Terminator indicates it does not want to be informed of contradictions, the Composer will  
1026       immediately reply with TRANSACTION\_CANCELLED. Otherwise, if and when CANCELLED  
1027       or RESIGN has been received from all Inferiors, the Composer replies to the Terminator with  
1028       TRANSACTION\_CANCELLED; but if HAZARD or CONFIRMED is received from any  
1029       Inferior, the reply is INFERIOR\_STATUSES, identifying which Inferior(s) had problems.

1030       If the completion message is CONFIRM\_TRANSACTION, the inferiors-list parameter of the  
1031       message defines the Confirm-set. If the parameter is absent (which it must be for an Atom  
1032       Coordinator), then all Inferiors (excluding only those that have resigned) are the Confirm-set;  
1033       otherwise the Confirm-set is only the Inferiors identified in the inferiors-list parameter (less any  
1034       from which RESIGN has been received). The processing to arrive at the Confirm decision is:

1035                  • If at the point of receiving CONFIRM\_TRANSACTION or at any point before making  
1036                  the Confirm decision (see below), CANCELLED is received, then the transaction is  
1037                  cancelled and processing continues as if CANCEL\_TRANSACTION had been received.

- 1038 • If there any Inferiors **not** in the Confirm-set from which neither CANCELLED or  
1039 RESIGN has been received, CANCEL is sent to them (this cannot happen for Atom  
1040 Coordinators)
- 1041 • If initially or later, there is exactly one Inferior in the Confirm-set, and either PREPARE  
1042 has not been sent to it, or PREPARED has been received from it, then at implementation  
1043 or configuration option, CONFIRM\_ONE\_PHASE can be sent to that Inferior. This  
1044 delegates the Confirm decision to the Inferior
- 1045 • If at any point, RESIGN is received from an Inferior, it is immediately removed from  
1046 the Confirm-set (this may trigger the decision making)
- 1047 • If there are any Inferiors in the Confirm-set from which none of PREPARED,  
1048 CANCELLED has been received and to which PREPARE has not yet been sent,  
1049 PREPARE is sent to that Inferior
- 1050 • If initially or later, PREPARED has been received from all Inferiors in the Confirm-set,  
1051 the Composer *makes the Confirm decision*; it persists (or attempts to persist) information  
1052 identifying the Inferiors in the Confirm-set; if this fails, the transaction is cancelled and  
1053 processing continues as if CANCEL\_TRANSACTION had been received; if the  
1054 information is persisted, the Confirm decision has been made.

1055 When the Confirm decision is made, CONFIRM is sent to all the Inferiors in the Confirm-set.  
1056 And, if on the CONFIRM\_TRANSACTION the Terminator indicated it did not wish to be  
1057 informed of contradictions, TRANSACTION\_CONFIRMED is sent to the Terminator.

1058 If the Terminator indicated it wanted to be informed of contradictions, the Composer replies to it  
1059 with TRANSACTION\_CONFIRMED if and when CONFIRMED has been received from all the  
1060 Inferiors in the Confirm-set and CANCELLED or RESIGN has been received from any other  
1061 Inferiors. If other replies (CANCELLED from a Confirm-set Inferior, CONFIRMED from other  
1062 Inferiors, HAZARD from any) are received, the reply to the Terminator is  
1063 INFERIOR\_STATUSES, identifying which Inferior(s) had problems.

1064 Figure 5-14 shows an example message sequence for a Composer with three Inferiors. The  
1065 Terminator (Application Element) chooses to prepare Inferiors 1 and 3 explicitly – the numbers in  
1066 parentheses on the Terminator:Composer messages represent the inferior-identifiers in the  
1067 “inferior-list” parameters. Both 1 and 3 prepare successfully, but the Terminator then decides to  
1068 make 1 and 2 the Confirm-set; that is, if the transaction confirms only 1 and 2 are confirmed. The  
1069 Terminator issues CONFIRM\_TRANSACTION to the Composer. A PREPARED message has  
1070 not been received from Inferior 2 yet, so the Composer issues PREPARE to it, and waits for the  
1071 PREPARED. At the same time, it sends CANCEL to Inferior 3, which has been excluded from  
1072 the Confirm-set by the CONFIRM\_TRANSACTION. After the PREPARED is received from  
1073 Inferior 2, the Composer makes the Confirm decision and issues CONFIRM to the Inferiors, and  
1074 waits for the CONFIRMED messages before reporting to the Terminator. The  
1075 CONFIRM\_TRANSACTION in this case did not ask for reporting of hazards (see below) – if it  
1076 had not, the TRANSACTION\_CONFIRMED would have been sent at the same time as the  
1077 CONFIRM messages.

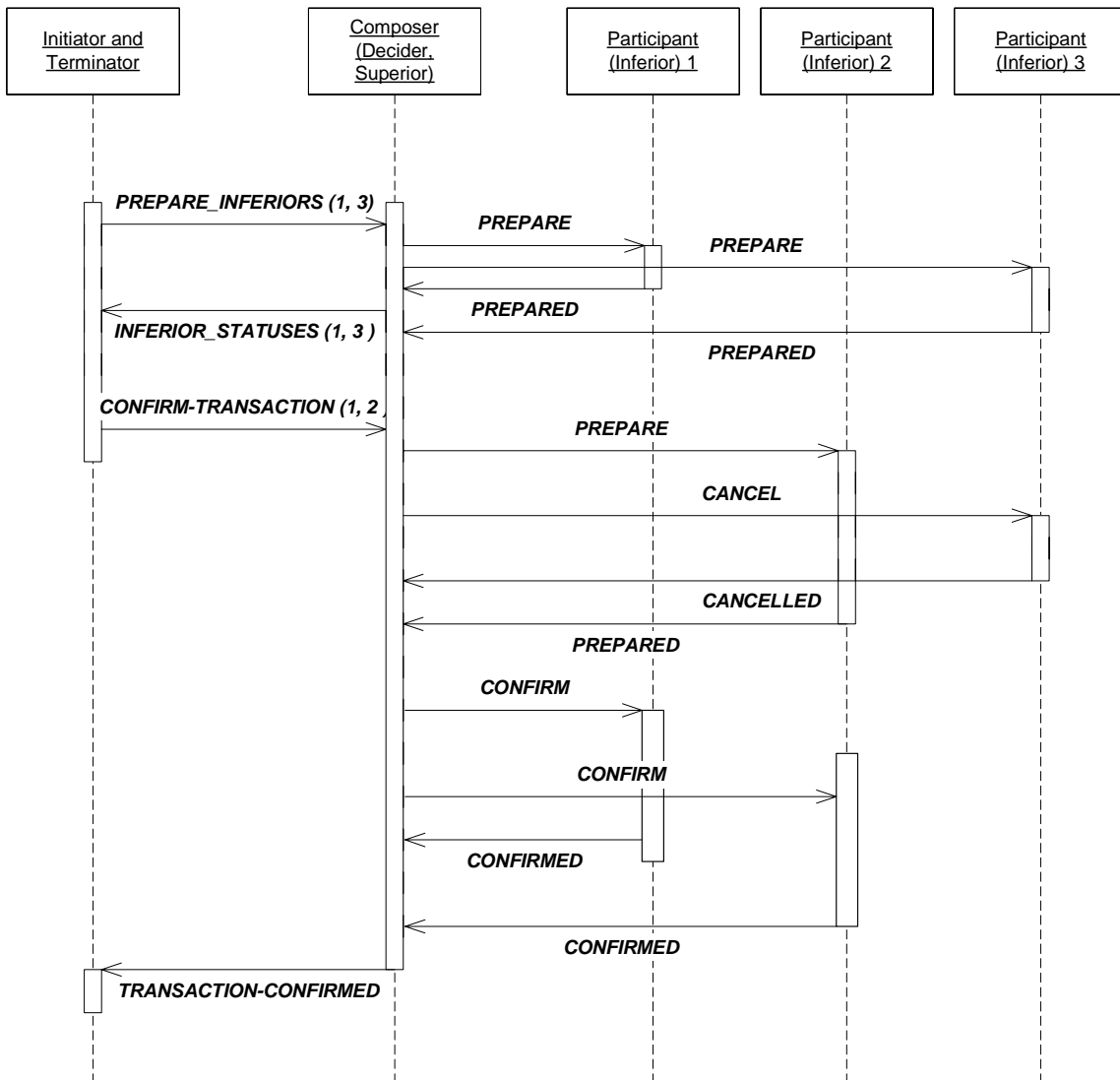


Figure 5-14 Termination sequence for a composer

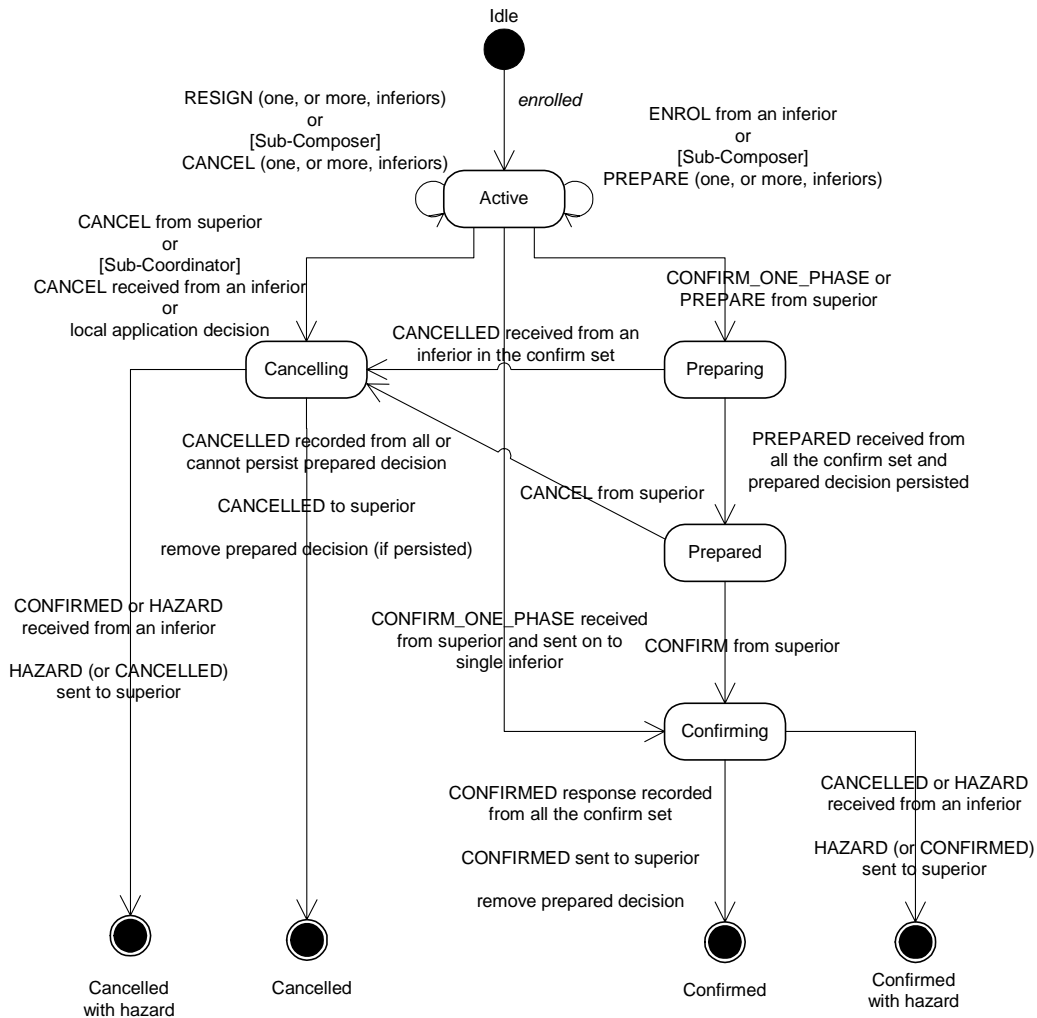
1078

1079 **5.2.8 Confirm-set of intermediates**

1080 An Intermediate, that is a Superior that is also an Inferior, also has a Confirm-set, but this is  
 1081 controlled rather differently to the top-most Superior (Decider) described above.

1082 As an Inferior, the interface between the application and BTP Elements is not fully defined in this  
 1083 specification. However, within the standard Control Relationship, issuing BEGIN with a related  
 1084 CONTEXT to a Factory will cause the creation of a Sub-coordinator or Sub-composer (depending  
 1085 on whether the BEGIN parameter asked for atomic or cohesive behaviour). Initially, of course,  
 1086 the new Intermediate has no Inferiors – however, unlike a Participant (in the strict sense of the  
 1087 term), it has a “superior-address” to which ENROL can be sent to enrol Inferiors. This address is  
 1088 a field of the new CONTEXT.

1089 Figure 5-15 is a state diagram for a Sub-composer or Sub-coordinator.



1090

1091

**Figure 5-15 State diagram for Sub-coordinator or Sub-composer**

1092

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

1093

1094

- ENROL messages can be received, adding a new Inferior

1095

- Inferiors may resign - RESIGN is received from an Inferior. The Inferior is immediately removed from the set of Inferiors

1096

1097

- CANCELLED may be received from an Inferior

1098

- PREPARED may be received from an Inferior

1099

In some circumstances, receipt of an incoming message allows an Intermediate to determine that a state change for the whole Transaction Tree Node takes place. The Intermediate is able to send messages to its Superior at its own initiative (whereas a Decider can only respond to a received message from the Terminator), so the receipt of a message from an Inferior can trigger the

1100

1101

1102

1103 sending of messages. This is especially the case if the Intermediate knows (from application  
1104 knowledge, perhaps involving received or sent CONTEXT\_REPLY messages) that there will be  
1105 no further enrolments. In particular:

1106       • If CANCELLED is received from an Inferior, and this is a Sub-coordinator, the Sub-  
1107 coordinator can itself Cancel - CANCEL is sent to other Inferiors, and CANCELLED to  
1108 the Superior

1109       • If RESIGN is received from the only Inferior and there will be no other enrolments, the  
1110 Intermediate can itself resign, sending RESIGN to the Superior

1111       • If PREPARED is received from the Inferior, it is known there will be no other  
1112 enrolments and this is a Sub-coordinator, the Sub-coordinator can Become Prepared  
1113 (assuming successful persistence of the appropriate information) and send PREPARED  
1114 to the Superior.

1115 For a Sub-composer, application logic will invariably be involved in determining what effect a  
1116 CANCELLED and PREPARED from an Inferior have – though in a real implementation, this  
1117 logic may be delegated to the BTP-support software.

1118 The Intermediate may initiate cancellation or the two-phase outcome exchange, either as a result  
1119 of receiving the corresponding message (CANCEL, PREPARE) from the Superior, or triggered  
1120 by its own controlling Application Element. For a Sub-composer, this may be partial - a Sub-  
1121 composer might be instructed by the Application Element to Cancel some Inferiors and send  
1122 PREPARE to others. Receipt of PREPARE from the Superior will often have a similar effect to a  
1123 Decider receiving CONFIRM\_TRANSACTION – PREPARE is propagated to all Inferiors that  
1124 have not indicated they are PREPARED. However, exactly what happens on receiving PREPARE  
1125 will depend on the application – receipt of the PREPARE may be visible to the Application  
1126 Element and cause it to initiate further application activity (perhaps causing enrolment of new  
1127 Inferiors) before it is determined whether to propagate PREPARE, and with a Sub-composer,  
1128 some of the Inferiors may be instructed to Cancel instead.

1129 Assuming the Intermediate does not Cancel as a whole (in which case CANCEL would be sent to  
1130 all Inferiors), the Intermediate will at some point attempt to Become Prepared. If it is a Sub-  
1131 coordinator, this will require that PREPARED has been received from all Inferiors. For a Sub-  
1132 composer, application logic will determine from which Inferiors PREPARED is required, with  
1133 the others being cancelled. In either case, the Intermediate will persist the information about the  
1134 Inferiors that are to be in the Confirm-set and about the Superior, if this persisting is successful,  
1135 send PREPARED to its own Superior.

1136 If CANCEL is subsequently received from the Superior, this is propagated to all the Inferiors and  
1137 the persistent information removed (or effectively removed as far as recovery is concerned). It is  
1138 not important which order this is done in, since the recovery sequence will ensure that a cancel  
1139 outcome is eventually delivered anyway.

1140 If CONFIRM is received from the Superior (which can only be after sending PREPARED to the  
1141 Superior), this is likewise propagated to the Inferiors. For a Sub-coordinator, CONFIRM is  
1142 invariably sent to all Inferiors. However, for a Sub-composer it is possible further application  
1143 logic intervenes and some of the Inferiors are rejected from the Confirm-set at this late stage.

1144 (This can only occur when the application work, as defined by the Contract to the Superior, can  
1145 be performed by some sub-set of the Inferiors.) The Intermediate may, but is not required to,  
1146 change the persistent information to reflect the Confirm outcome (though a Sub-composer that  
1147 selects only some Inferiors probably will need to re-write the information to ensure the correct  
1148 subset are confirmed despite possible failures). If the information is not changed, then, on  
1149 recovery, the Intermediate will find itself to be in a prepared state and will interrogate the  
1150 Superior to re-determine the outcome. If the information is changed, a recovered Intermediate can  
1151 immediately continue with ordering confirmation to its Inferiors.

1152 If CONFIRM\_ONE\_PHASE is received from the Superior, either before or after the Intermediate  
1153 has Become Prepared, the effect is very similar to a Decider receiving  
1154 CONFIRM\_TRANSACTION. If there is only one Inferior, the CONFIRM\_ONE\_PHASE may  
1155 be propagated to that Inferior. Otherwise, the Intermediate behaves as a Decider, making a  
1156 Confirm decision if it can.

1157 If one or more Inferiors make contradictory autonomous decisions, or HAZARD is received from  
1158 an Inferior, the Intermediate may report this to the Superior using HAZARD. However, BTP does  
1159 not require this. Since the Superior may be owned and controlled by a different organisation,  
1160 there may be business reasons not to report such problems.

## 1161 **5.3 Optimisations and variations**

### 1162 **5.3.1 Spontaneous prepared**

1163 As described above, before a Superior can order confirmation to an Inferior, the Inferior must  
1164 become “prepared”, meaning that it is ready to Confirm or to Cancel as it so ordered and send the  
1165 PREPARED message as a report of this. In the conventional message sequence, as shown above,  
1166 the Inferior attempts to Become Prepared when it receives a PREPARE message from the  
1167 Superior. The PREPARE in turn is sent by the Superior when it receives an appropriate request  
1168 from its controlling application (or from its own Superior, if there is one). The application  
1169 controlling the Superior will request the sending of PREPARE when it determines that no further  
1170 application work associated with this Inferior (or, perhaps with the whole Business Transaction)  
1171 will occur.

1172 However, for some applications, the Application Element controlling the Inferior will know that  
1173 the application work for which the Inferior will be responsible is complete before a PREPARE is  
1174 sent from the Superior. In fact, because the Application Element has autonomy in determining  
1175 how application work is to be allocated to Inferiors, it is possible for the Inferior-side Application  
1176 Element to know the work is complete **for a particular Inferior** when Superior-side Application  
1177 Element will be sending more message to the Inferior-side. (The future work will, probably,  
1178 require the enrollment of additional Inferiors.)

1179 BTP consequently allows the Application Element controlling an Inferior to cause the Inferior to  
1180 Become Prepared, and to send PREPARED to the Superior without PREPARE having been  
1181 received from the Superior. From the perspective of the BTP Superior the Inferior sends  
1182 PREPARED spontaneously. Apart from this, a spontaneous PREPARED message is the same as,  
1183 and has the same effect and implications as one induced by a PREPARE message.

### 1184 5.3.2 One-shot

1185 In the “conventional” message sequence shown above and assuming the Initiator, Terminator and  
1186 Coordinator on the one side, and “Service”, Enroller and Participant on the other are located  
1187 within their respective parties, there are eight messages passed in one direction or the other  
1188 between the two parties. There are four round-trip exchanges: the application request and  
1189 response exchange, the ENROL/ENROLLED exchange (going in the opposite direction and  
1190 overlapped with the application exchange), then PREPARE/PREPARED and the  
1191 CONFIRM/CONFIRMED. However, if the application exchange is a single request/response, it  
1192 is possible to reduce these eight to two round-trips– the first of which merges the first three of the  
1193 conventional sequence. The fundamental two-phase nature of BTP (or any coordination  
1194 mechanism) means there have to be at least two round trips – one before the Confirm-or-Cancel  
1195 decision is made at the Superior, one after. This merging of the exchanges is termed “one-shot”,  
1196 as it requires only one exchange to take the relationship from non-existent to waiting for the  
1197 Confirm-or-Cancel decision.

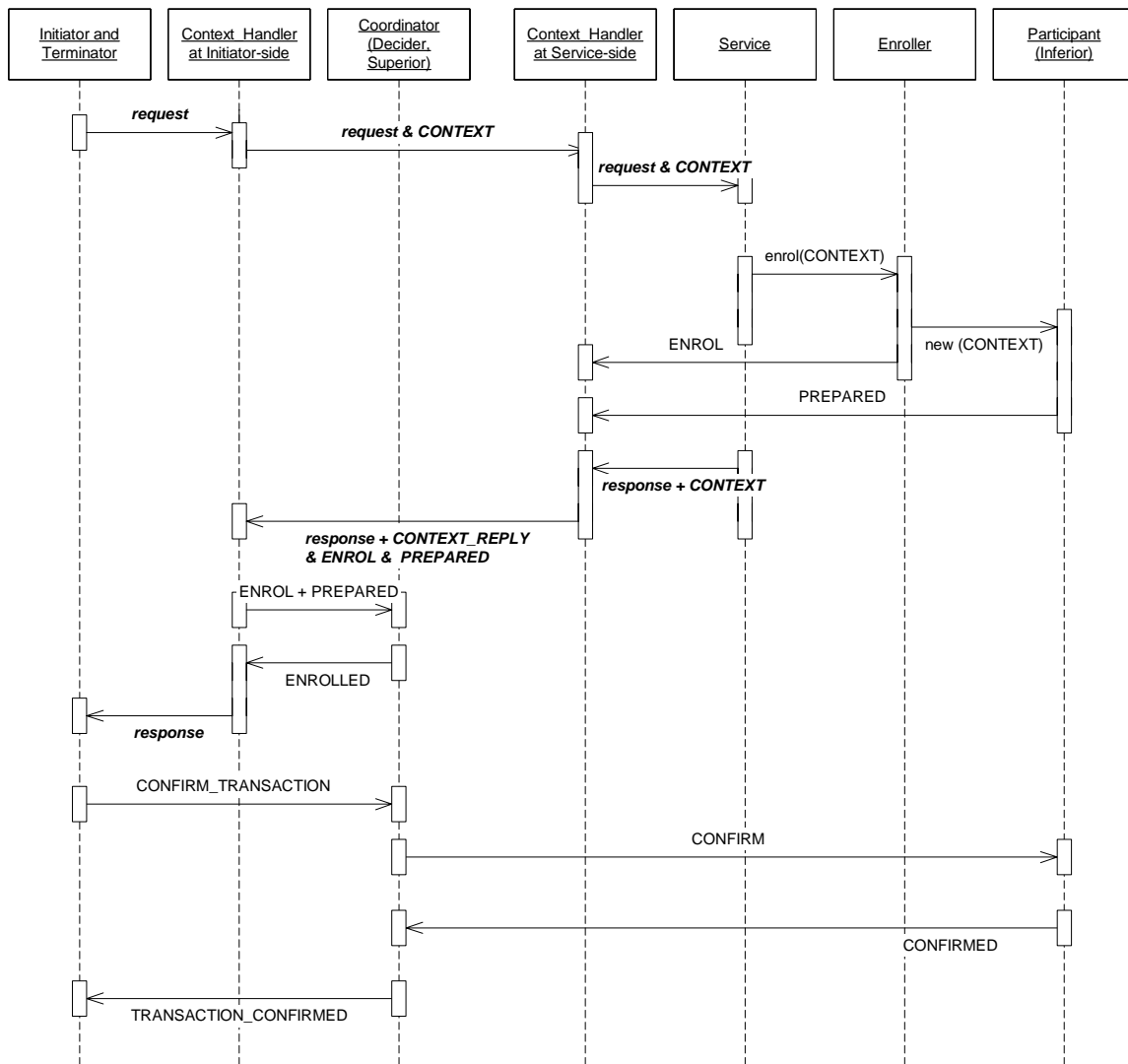
1198 Figure 5-16 shows a typical “one-shot” message sequence. The diagram distinguishes an  
1199 additional aspect of the Application Elements, labelled “context-handler”. This is not a Role in  
1200 the BTP model, but is used only to distinguish a set of responsibilities and actions. In a real  
1201 implementation these might be performed by the user application itself, or might be performed by  
1202 the BTP-supporting infrastructure on the path between the Application Elements. (Figure 5-9  
1203 could be redrawn to show the context-handlers, but to no particular benefit) As in the  
1204 conventional case, the CONTEXT is sent related to the application request (the creation of the  
1205 CONTEXT by the Factory is not shown and is the same as the conventional case). The “context-  
1206 handler” is aware of the sending of the CONTEXT.

1207 On the responder (service side), however, when the Application Element creates the Inferior, the  
1208 ENROL is not sent immediately, but retained. The application performs the “Provisional Effect”  
1209 implied by the received message and the Inferior becomes prepared and issues a PREPARED  
1210 message, which is also retained. When the application response is available, it is sent with the  
1211 retained messages and the CONTEXT\_REPLY (which indicates that the related ENROL will  
1212 complete the enrolments implied by the earlier transmission of the CONTEXT).

1213 When this group of messages is received by the context-handler on the Client side, the contained  
1214 ENROL and PREPARED messages are forwarded to the Superior (whose address was on the  
1215 original CONTEXT and so is known to the context-handler). An ENROLLED message is sent  
1216 back to the context-handler, assuring it that the enrolment was successful and the application can  
1217 progress. If enrollment fails and the Business Transaction is atomic, confirmation must be  
1218 prevented – this responsibility falls on the context-handler and the Client application, since the  
1219 failure of the enrolment implies that Superior itself is inaccessible. If enrolment fails and the  
1220 Business Transaction is a Cohesion, the appropriate response is a matter for the application.

1221 With “one-shot”, if there are multiple Inferiors created as a result of a single Application  
1222 Message, there is an ENROL and PREPARED message for each one sent related with the  
1223 CONTEXT\_REPLY. If an operation fails, a CANCELLED message may be sent instead of a  
1224 PREPARED – if the Superior is atomic, this will ensure it cancels, if cohesive, the Client  
1225 application will be aware of this and behave appropriately.

1226 Whether the “one-shot” mechanism is used is determined by the implementation on the  
 1227 responding (Inferior) side. This may be subject to configuration and may also be constrained by  
 1228 the application or by the binding in use.



1229

1230

**Figure 5-16 A message sequence showing the “one-shot” optimisation**

1231

### 5.3.3 Resignation

1232

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.

1236

1237

1238

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.

1239

#### 1240 5.3.4 One-phase confirmation

1241 If a Coordinator or Composer that has been requested to Confirm has only one (remaining)  
1242 Inferior in the Confirm-set, it may delegate the Confirm-or-Cancel decision to that Inferior, just  
1243 requesting it to Confirm rather than performing the two-phase exchange. This is done by sending  
1244 the CONFIRM\_ONE\_PHASE message. Unlike the two-phase exchange (PREPARED received,  
1245 CONFIRM sent), it is possible with CONFIRM\_ONE\_PHASE for a failure to occur that leads to  
1246 the original Coordinator or Composer (and its controlling Application Element – the Terminator)  
1247 being uncertain whether the outcome was confirmation or cancellation.

#### 1248 5.3.5 Autonomous cancel, autonomous Confirm and contradictions

1249 As described above, BTP does not require a Participant, while it is responsible for holding  
1250 application resources such that can be confirmed or cancelled, to use any particular mechanism  
1251 for maintaining this state. A Participant that “becomes prepared” may choose to let the  
1252 “Provisional Effect” be identical to the “Final Effect”, and hold a compensating “counter effect”  
1253 ready to implement cancellation; or it may make the Provisional Effect effectively null, and only  
1254 perform the real application work as the Final Effect if confirmed; or the “Provisional Effect”  
1255 may involve performance of the application work and locking application data against other  
1256 access; or other patterns, as may be constrained or permitted by the application.

1257 Although a Participant is not required to lock data (as would be the case with some other  
1258 transaction specifications) on becoming prepared, it is nevertheless in a state of doubt, and this  
1259 doubt may have application or business implications. Accordingly it is recognised that a  
1260 Participant (or, rather the business party controlling the Application Element and the Participant)  
1261 may need to limit the promise made by sending PREPARED, and retain the right to apply its own  
1262 decision to Confirm or Cancel to the Participant and the application effects it is responsible for.  
1263 This is described as an “autonomous” decision. It is closely analogous to the heuristic decisions  
1264 recognised in other transaction specifications. The only difference is the conceptual one that  
1265 heuristic decisions are typically considered to occur only as a result of rare and unpredictable  
1266 failure, whereas BTP recognises that the right to take an autonomous decision may be critical to  
1267 the willingness of a business party to be involved in the Business Transaction at all. BTP  
1268 therefore allows Participants (and all Inferiors) to indicate that there are limits on how long they  
1269 are willing to promise to remain in the prepared state, and that after that time they may invoke  
1270 their right of taking an autonomous decision.

1271 Taking an autonomous decision will of course run the risk of breaking the intended consistency of  
1272 outcome across the Business Transaction, if the autonomous decision of the Inferior contradicts  
1273 the decision (for this Inferior) made by the Superior. The Superior will have received the  
1274 PREPARED message and thus be permitted to make a Confirm decision (directly, or through  
1275 exchanges with a Terminator Application Element or with its own Superior). An Inferior taking  
1276 an autonomous decision informs the Superior by sending CONFIRMED or CANCELLED, as  
1277 appropriate, without waiting for an outcome order from the Superior. This may cross the outcome  
1278 message from the Superior, or the Superior may not make its decision till later. If the decisions  
1279 agree, the normal CONFIRM or CANCEL message is sent. In the case of CANCEL, this  
1280 completes the relationship – the CANCEL and CANCELLED messages acknowledge each other,  
1281 regardless of which travels first. In the case of CONFIRM, another CONFIRMED message is  
1282 needed.

1283 If the Superior's decision is contradicted by the autonomous decision, the Superior may need to  
1284 record this, report it to management systems or inform the Terminator application or its own  
1285 Superior. When this has been done (details are implementation-specific, but may be constrained  
1286 by the application), the Superior sends a CONTRADICTION message to the Inferior. If an  
1287 outcome message was sent earlier (crossing the announcement of the autonomous decision), the  
1288 Inferior will already know there was a contradiction, but the receipt of the CONTRADICTION  
1289 message informs the Inferior that the Superior knows and has done whatever it considers  
1290 necessary to cope.

1291 As mentioned, BTP allows an Inferior to inform the Superior, with a qualifier on the PREPARED  
1292 message, that the promise to remain in the prepared state will expire. In turn this allows the  
1293 application on the Superior side to avoid risking a contradictory decision by making and sending  
1294 its own decision in time. The Superior side can also indicate, with another qualifier, a minimum  
1295 time for which it expects the prepared promise to remain valid.

1296

1297 As well as deliberate and forewarned autonomous decisions, BTP recognises that failures and  
1298 exceptional conditions may force unplanned autonomous decisions. In the protocol sequence  
1299 these are treated exactly like planned autonomous decisions – if they contradict, the Superior will  
1300 be informed and a CONTRADICTION message sent to the Inferior.

1301 Autonomous decisions, planned or unplanned, are equivalent to the heuristic decisions of other  
1302 transaction systems. The term is avoided in BTP since it may carry implications that it only  
1303 occurs in an unplanned manner.

## 1304 **5.4 Recovery and failure handling**

### 1305 **5.4.1 Types of failure**

1306 BTP is designed to ensure the delivery of a consistent decision for a Business Transaction to the  
1307 parties involved, even in the event of failure. Failures can be classified as:

1308 **Communication failure:** messages between BTP Actors are lost and not delivered. BTP  
1309 assumes the Carrier Protocol ensures that messages are either delivered correctly (without  
1310 corruption) or are lost, but does not assume that all losses are reported nor that messages  
1311 sent separately are delivered in the order of sending.

1312 **Network Node failure (system failure, site failure):** a machine hosting one or more  
1313 BTP Actors stops processing and all its volatile data is lost. BTP assumes a site fails by  
1314 stopping – it either operates correctly or not at all, it never operates incorrectly.

1315 Communication failure may become known to a BTP implementation by an indication from the  
1316 lower layers or may be inferred (or suspected) by the expiry of a timeout. Recovery from a  
1317 communication failure requires only that the two Actors can again send messages to each other  
1318 and continue or complete the progress of the Business Transaction.

1319 A Network Node failure is distinguished from communication failure because there is loss of  
1320 volatile state. To ensure consistent application of the decision of a Business Transaction, BTP  
1321 requires that some state information will be persisted despite Network Node failure. Exactly what

1322 real events correspond to Network Node failure but leave the persistent information undamaged is  
1323 a matter for implementation choice, depending on application requirements; however, for most  
1324 application uses, power failure should be survivable (an exception would be if the data  
1325 manipulated by the associated operations was volatile). In all cases, there will be some level of  
1326 event sufficiently catastrophic to lose persistent information and the ability to recover—  
1327 destruction of the computer or bankruptcy of the organisation, for example.

1328 Recovery from Network Node failure involves recreating an accessible communications endpoint  
1329 in a Network Node that has access to the persistent information for incomplete transactions. This  
1330 may be a recreation of the original Actor using the same addresses; or using a different address;  
1331 or there may be a distinct recovery entity, which can access the persistent data, but has a different  
1332 address; other implementation approaches are possible. The recovered, and possibly relocated  
1333 Actor may or may not be capable of performing new application work Restoration of the Actor  
1334 from persistent information will often result in a partial loss of state, relative to the volatile state  
1335 reached before the failure. In some states, there may be total loss of knowledge of the Business  
1336 Transaction, including particular Superior:Inferior relationships. After recovery from Network  
1337 Node failure, the implementation behaves much as if a communication failure had occurred.

#### 1338 5.4.2 Persistent information

1339 BTP **requires** that certain state information is persisted – these are information that records an  
1340 Inferior’s decision to be prepared, a Superior’s decision to Confirm and an Inferior’s autonomous  
1341 decision . Requiring the first two to be persistent ensures that a consistent decision can be reached  
1342 for the Business Transaction and that it is delivered to all involved BTP Nodes, despite failure.  
1343 Requiring an Inferior’s autonomous decision to be persistent allows BTP to ensure that, if the  
1344 autonomous decision is contradictory (i.e. opposite to the decision at the Superior), the  
1345 contradiction will be reported to the Superior, despite failures.

1346 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the active  
1347 state (unlike many transaction protocols, where a communication or node failure in active state  
1348 would invariably cause rollback of the transaction). Recovery in the active state may require that  
1349 the application exchange is resynchronised as well – BTP does not directly support this, but  
1350 allows continuation of the Business Transaction if the application desires it. Apart from the  
1351 (optional) recovery in active state, BTP follows the well-known presume-abort model – it is only  
1352 **required** that information be persisted when decisions are made (and not, for example, on  
1353 enrolment). This means that on recovery one side may have persistent information while the other  
1354 does not. This occurs, among other cases, when an Inferior has decided to be prepared but the  
1355 Superior never confirmed (so the decision is “presumed” to be cancelled), and when the Superior  
1356 did Confirm, the Inferior applied the confirmation and removed its persistent information but the  
1357 acknowledgement message (CONFIRMED) was never received by the Superior.

1358 Information to be persisted when an Inferior decides to be prepared has to be sufficient to re-  
1359 establish communication with the Superior, to apply a Confirm decision and to apply a Cancel  
1360 decision. It will thus need to include the addressing and identification information for the  
1361 Superior. The information needed to apply the Confirm or Cancel decision will depend on the  
1362 application and the associated operations.

1363 A Superior must persist the corresponding information to allow it to re-establish communication  
1364 with the Inferior – that is the addressing and identification information for the Inferior. When it

1365 must persist this information depends on its position within the Transaction Tree. If it is the top of  
1366 the tree – i.e. it is the Decider for the Business Transaction -- it need only persist this information  
1367 if and when it makes a decision to Confirm (and, for a Cohesion, only if this Inferior is in the  
1368 Confirm-set). A Superior that is an intermediate in the tree – i.e. it is an Inferior to some other  
1369 Superior – must persist the information about each of its own Inferiors as part of (or before)  
1370 persisting its own decision to be prepared. For such an intermediate, the “decision to confirm” as  
1371 Superior is made when either CONFIRM is received from its Superior or it makes an autonomous  
1372 decision to Confirm. If CONFIRM is received, the persistent information may be changed to  
1373 show the Confirm decision, but alternatively, the receipt of the CONFIRM can be treated as the  
1374 decision itself and the CONFIRM message propagated to the Inferiors without changing the  
1375 persistent information. If the persistent information is left unchanged and there is a node failure,  
1376 on recovery the entity (as an Inferior) will be in a prepared state, and will rediscover the Confirm  
1377 decision (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

1378 Since BTP messages may carry application-specified qualifiers, and the BTP messages may be  
1379 repeated if they are lost in transit (see next section), the persistent information may need to  
1380 include sufficient to recreate the qualifiers, to allow them to be resent with their carrying BTP  
1381 message. This applies both to qualifiers on PREPARED (which would be persisted by the  
1382 Inferior) and on CONFIRM (which would be persisted by the Superior).

1383 In some cases, an implementation may not need to make an active change to have a persistent  
1384 record of a decision, provided that the implementation will restore itself to the appropriate state  
1385 on recovery. For example, an implementation that, as Inferior, always used the default-is-cancel  
1386 mechanism, and recorded the timeout (to Cancel) in the persistent information on becoming  
1387 prepared, and always updated or removed that record when it applied a Confirm instruction could  
1388 treat the presence of an expired record as effectively a record of an autonomous Cancel decision.

### 1389 5.4.3 Recovery messages

1390 Once the Superior:Inferior relationship has entered the completion phase – BTP does not  
1391 generally use special messages in recovery, but merely permits the resending of the previous  
1392 message – thus, for example, PREPARE, PREPARED, CANCEL, CONFIRM can all be sent  
1393 repeatedly. Resending the previous message means a possible loss of the original message may be  
1394 invisible to the receiver. The trigger for this re-sending is implementation dependent – a reported  
1395 communication failure, a timeout expiry while waiting for a reply, the re-establishment of  
1396 communications or the general restoration of function after a node failure are all possible triggers.  
1397 An incoming repetition of the last message received, if it has already been replied to (e.g.  
1398 receiving PREPARE after PREPARED has been sent), should normally trigger a resending of the  
1399 last message sent – since that sent message may have got lost.<sup>4</sup>

1400 While in the active phase – i.e. prior to entering completion – there is no appropriate last message  
1401 that can be sent. However, for active-phase recovery there needs to be some way for the BTP  
1402 Actors to determine that the Peer is still there and still aware of the Superior:Inferior relationship.

---

<sup>4</sup> 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.

1403 In this case, the peers can interrogate each other using the INFERIOR\_STATE or  
1404 SUPERIOR\_STATE messages, informing the Peer of their own state and requesting a response –  
1405 which may be the opposite message, or one of the main BTP messages (which perhaps had been  
1406 lost). If it is another SUP|INFERIOR\_STATE message, that reply does not ask for a response.  
1407 Receiving a SUP|INFERIOR\_STATE messages that asks for a response does not require an  
1408 immediate response – especially if an implementation is waiting to determine a decision (perhaps  
1409 because it is itself waiting for a decision from elsewhere), an implementation may choose not to  
1410 reply until it wishes too.

1411 The SUP|INFERIOR\_STATE messages are also used as replies when the receiver of **any** of the  
1412 Superior:Inferior message has determined that there is no corresponding state information – the  
1413 targeted Superior or Inferior does not exist (or is known to have completed and is no longer an  
1414 active entity). The SUP|INFERIOR\_STATE messages with a status of “unknown” is the  
1415 indication that the state information does not exist.

1416 The SUP|INFERIOR\_STATE messages are also available as replies to any Superior:Inferior  
1417 message in the (transient, one hopes) case where, after failure an implementation cannot currently  
1418 determine whether the persistent information exists or not, or what its state is, and so cannot give  
1419 a definitive answer. The SUP|INFERIOR\_STATE messages with a status of “inaccessible” is the  
1420 indication that the existence of state information cannot be determined. The receiver of such a  
1421 message should normally treat it as a “retry later” suggestion.

#### 1422 5.4.4 Redirection

1423 As described above, BTP uses the presume-abort model for recovery. A corollary of this is that  
1424 there are cases where one side will attempt to re-establish communication when there is no  
1425 persistent information for the relationship at the far-end, because that side either never reached a  
1426 state where the state was persisted, or had been persisted, but then progressed to remove the state  
1427 information. In such cases, it is important the side that is attempting recovery can distinguish  
1428 between unsuccessful attempts to connect to the holder of the persistent information and when the  
1429 information no longer exists. If the Peer information does not exist, the side that is attempting  
1430 recovery can draw appropriate conclusions (that the Peer either was never prepared, never  
1431 confirmed or has already completed) and complete its part of the transaction; if it merely fails to  
1432 get through, it is stuck in attempting recovery.

1433 Two mechanisms are provided to assist implementation flexibility while allowing completion of  
1434 Superior:Inferior relationships when only one side has any persistent information. The  
1435 mechanisms are:

- 1436 • Address fields which provide the address that will be used by the Peer to send messages  
1437 to an Actor (effectively a “callback address”) can be a set of addresses, which are  
1438 alternatives, one of which is chosen as the target address for the future message. If the  
1439 sender of that message finds the address does not work, it can try a different alternative.
- 1440 • The REDIRECT message can be used to inform the Peer that an address previously  
1441 given is no longer valid and to supply a replacement address (or set of addresses).  
1442 REDIRECT can be issued either as a response to receipt of a message or spontaneously.

1443 The two mechanisms can be used in combination, with one or more of the original set of  
1444 addresses just being a redirector, which does not itself ever have direct access to the state  
1445 information for the transaction, but will respond to any message with an appropriate REDIRECT.

1446 REDIRECT as a message is only used on the Superior:Inferior relationship, where each side  
1447 holds the address of the other. On the other relationships (e.g. Terminator:Decider), one side (e.g.  
1448 Terminator) has the address of the other, and initiates all the message exchanges. However, the  
1449 entity whose address is known to the other may itself move - e.g. if a Coordinator, which will be  
1450 both Decider and Superior changes its address as a Superior, it will probably change its address as  
1451 a Decider too. In this case, a FAULT reply to a misdirected message can be used, assuming there  
1452 is some entity available at, or on the path to the old address that understands BTP sufficiently to  
1453 provide the redirection information.

1454 Some implementations, in which a single addressable entity with one, constant address deals with  
1455 all transactions, distinguishing them by identifier, will not need to supply “backup” addresses  
1456 (and would only use REDIRECT if permanently migrated).

#### 1457 **5.4.5 Terminator:Decider failures and transaction timelimit**

1458 BTP does not provide facilities or impose requirements on the recovery of Terminator:Decider  
1459 relationships, other than allowing messages to be repeated. A Terminator may survive failures (by  
1460 retaining knowledge of the Decider’s address and identifier), but this is an implementation option.  
1461 Although a Decider (if it decides to Confirm) will persist information about the Confirm decision,  
1462 it is not required, after failure, to remain accessible using the address it originally gave to the  
1463 Initiator (and used by the Terminator). Any such recovery is an implementation option.

1464 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and thus  
1465 no way of detecting that a Terminator has failed. The Decider always has the right to initiate  
1466 cancellation, but if the application (Terminator) and the Decider have different views about how  
1467 long a “long time” is, then either the Decider might wait unnecessarily for a completion request  
1468 (e.g. CONFIRM\_TRANSACTION) that will never arrive, or it might initiate cancellation while  
1469 the application is still active. To avoid these irritations, a standard qualifier “Transaction  
1470 timelimit” can be used (by the Initiator) to inform the Decider when it can assume the Terminator  
1471 will not request confirmation and so it (the Decider) should initiate cancellation.

#### 1472 **5.4.6 Contradictions and hazard**

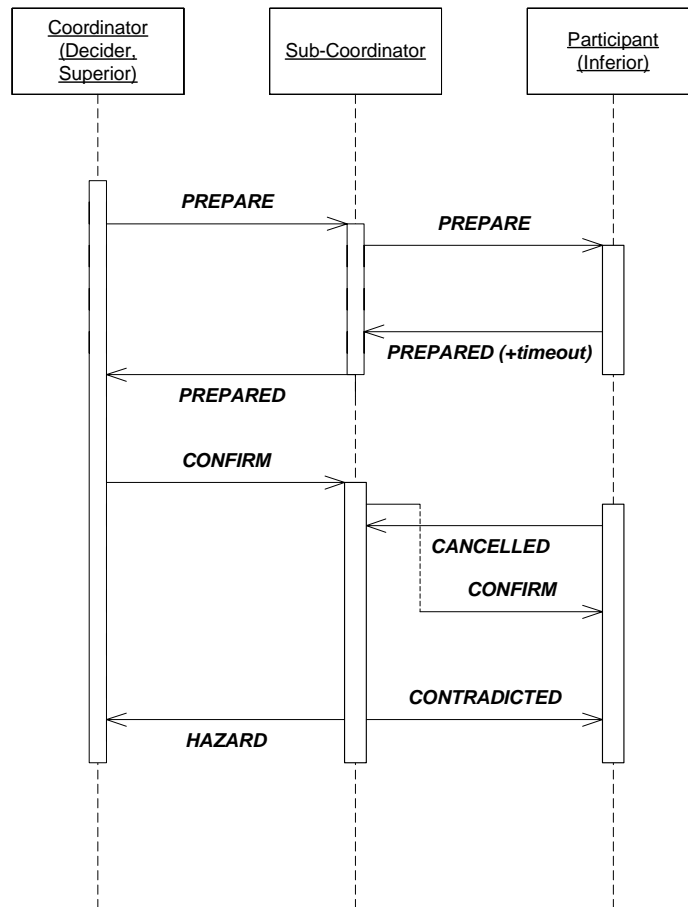
1473 As described above (see “5.3.5 Autonomous cancel, autonomous Confirm and contradictions”),  
1474 in some circumstances an Inferior may apply a decision that is contradictory to the decision of the  
1475 Superior. This can occur in a semi-planned manner, when the Inferior has announced a timeout on  
1476 the PREPARED message but no outcome message has been received, or as a result of an  
1477 exceptional condition that forces the Inferior to break the promise implicit in PREPARED,  
1478 regardless of timers. In both cases, this is considered an autonomous decision by the Inferior. An  
1479 autonomous decision, of itself, does not imply a contradiction – it only results in a contradiction if  
1480 the decision is opposite to that of the Superior (in the case of a cohesive Superior, opposite to the  
1481 decision that applies to this Inferior).

1482 In order to ensure that a contradiction is detected despite node and communication failures, it is  
1483 required that information about the taking of the autonomous decision be persisted until a BTP

1484 message received from the Superior indicates either that there was no contradiction (the decisions  
1485 were in line – CANCEL is received after an autonomous Cancel or CONFIRM is received after  
1486 an autonomous Confirm) or that the Superior is aware of the contradiction (CONTRADICTION  
1487 is received). Note that the Inferior will become aware of the fact of the contradiction when it  
1488 receives the “wrong” message, but must retain the record of its own decision until it receives the  
1489 CONTRADICTION message, which tells it the Superior knows too.

1490 The Superior’s action on becoming aware of the contradiction is not determined by this  
1491 specification. In particular, if the Superior is a Sub-coordinator or Sub-composer, it is not  
1492 required by this specification to report the contradiction to its own Superior (which may, for  
1493 example, be controlled by a different organisation). The Superior may report the problem to  
1494 management systems or record it for manual repair. However, BTP does provide mechanisms to  
1495 report the contradiction to the next higher Superior (if there is one) or to the Terminator  
1496 Application Element.

1497 A contradiction occurring in an Inferior will usually mean the immediate Superior has a “mixed”  
1498 condition – some of the application work it was responsible for has confirmed, some has  
1499 cancelled (and contrary to any Cohesion Confirm-set selection). If the Superior is a Sub-  
1500 coordinator or Sub-composer, it can report the mixed condition to its own Superior with the  
1501 HAZARD message. If the Superior is the top-most in the tree, it can report the problem with the  
1502 INFERIOR\_STATUSES message, which will detail the state of all the Inferiors. Figure 5-17  
1503 shows a message sequence in a Transaction Tree with two levels. The Participant makes an  
1504 autonomous Cancel decision, but the Coordinator decides to Confirm. The Confirm decision from  
1505 the Coordinator, passed on by the Sub-coordinator crosses with the CANCELLED message from  
1506 the Participant. The Participant waits for the CANCELLED from the Sub-coordinator, which  
1507 chooses to report the problem with HAZARD to the Coordinator.



1508

1509

**Figure 5-17 Message sequence showing contradiction, reported with HAZARD**

1510 If a Sub-coordinator or Sub-composer having sent (or attempted to send) the outcome message to  
 1511 its Inferiors, is temporarily unable to get a response (CONFIRMED or CANCELLED), it may  
 1512 either wait until a response does come back or choose to reply to its own Superior with a  
 1513 HAZARD message indicating that a contradiction is “possible”. If it does choose to send  
 1514 HAZARD, it is required to persist a record of this until it receives a CONTRADICTION message  
 1515 from the Superior, or a message from the Inferior indicating there was no contradiction in fact.

1516 HAZARD is also used to indicate that it has become impossible to cleanly and consistently  
 1517 achieve either a confirmed or a cancelled state for the application work. In this case, there is can  
 1518 be no guarantee that the problem will be reliably reported – especially because it may be the  
 1519 inability to persist information that is the cause of the problem.

1520 **5.5 Relation of BTP to application and Carrier Protocols**

1521 BTP messages are communicated between Actors in two distinguishable circumstances:

- 1522 a) in establishing and progressing the outcome and Control Relationships between BTP
- 1523 Actors, and between Application Elements and BTP Actors – Initiator:Factory,
- 1524 Terminator:Decider, Superior:Inferior etc.

1525           b) in association with Application Messages that are communicated between  
1526           Application Elements.

1527       In the first case, interoperable communication requires a specification of how the abstract BTP  
1528       messages are represented and encoded, and how they are transmitted. This specification is a  
1529       **carrier protocol binding** (or just “binding”, if the context is clear). BTP allows bindings to a  
1530       multiplicity of Carrier Protocols. The only requirement that BTP makes is that the transmission of  
1531       a message either delivers an uncorrupted message or fails. BTP does not require that the carrier  
1532       report failure to deliver a message, to either side, nor that messages are delivered in the order they  
1533       are sent (though implementations can take advantage of information from a richer carrier, which  
1534       can improve performance in various ways). BTP messages communicated in this way have  
1535       semantics that are defined in this specification – a PREPARE message (for example), refers back  
1536       to the ENROL via the “inferior-identifier” parameter and is an instruction to the Inferior to  
1537       become and report that it is prepared.

1538       In the second case, the full semantics cannot be defined in this specification. Interoperation with  
1539       BTP requires that the parties have a common understanding of what is being confirmed or  
1540       cancelled, but this mutual understanding is defined by the Contract of the application, not by  
1541       BTP. (The Contract may be explicit or implicit, declared by one side as take-it-or-leave-it, or may  
1542       be negotiated in some way.) Part of this Contract will include how the combination of the  
1543       Application Protocol (i.e. the Application Messages and their sequencing) and BTP operate such  
1544       that the two sides are agreed as to which Application Operations are part of which Business  
1545       Transaction. This will often be achieved by sending Application Messages and BTP messages in  
1546       “association” in some way – thus an Application Message sent in association with a CONTEXT  
1547       can be specified (by the application Contract) to mean that if work is done as result of the receipt  
1548       of the message, one or more Inferiors should be enrolled to apply the Confirm/Cancel decision to  
1549       that work. Similarly, an Application Message may be sent associated with an ENROL with the  
1550       contractual understanding that the message refers to some application work that has been made  
1551       the responsibility of the Inferior being enrolled.

1552       The concrete representation of this “association” is also a matter for the Application Protocol  
1553       specification. There are several ways this can be done, including:

- 1554           • the BTP message is contained within the Application Message, or both are contained  
1555           within a larger construct;
- 1556           • the Application Message contains a field that is the superior-identifier or inferior-  
1557           identifier that is also present on the CONTEXT or the ENROL
- 1558           • the BTP message contains a qualifier that references (a field of) the Application  
1559           Message in some way (e.g. if the Application Message is an invoice, the qualifier might  
1560           contain the invoice number)
- 1561           • the encoding of the BTP and Application Messages reference each other (e.g. using  
1562           XML id and refid attributes)

1563 In all cases, the application specification<sup>5</sup> will need to define the mechanism so that both parties  
1564 have common understanding. Many applications will use the same mechanism and their  
1565 specifications can therefore take advantage of standard patterns, and their implementations of  
1566 standard tools.

1567 The association of an Application Message with a BTP message is analogous to the concept of  
1568 “related” BTP messages. “Related” BTP messages are sent as a group, with a declared and  
1569 defined semantic for the group. Associated application and BTP messages can be considered as  
1570 “related”, with the proviso that the semantic is defined by the application, not by BTP.

1571 There is no necessary relationship between how the Application Messages and any associated  
1572 BTP messages are transmitted by Carrier Protocols, and the carrier binding for the BTP messages.  
1573 BTP messages are invariably sent to a BTP Actor whose address has been passed to the sender by  
1574 some means – thus a CONTEXT contains the address of the Superior to which ENROLs will be  
1575 sent, and the ENROL contains the address of the Inferior. Similarly, BEGUN contains the address  
1576 (as Decider) of the new Composer or Coordinator. These addresses are all sets of addresses  
1577 (possibly of cardinality one), and each individual address identifies which binding is to be used.  
1578 Thus, for example, when a CONTEXT is sent associated with an Application Message, the  
1579 ENROL will travel on a carrier binding identified by the particular address from the CONTEXT  
1580 that the Enroller chooses to use – which may have no relationship to how the Application  
1581 Message arrived.

1582 Despite this, it will be common that the application binding and the BTP binding will use the  
1583 same carrier. This is the case in the bindings specified in this edition of the specification, which  
1584 define a binding of BTP to SOAP 1.1 over HTTP. Included in this SOAP/HTTP binding  
1585 specification, are rules that allow an application to associate (relate) a single CONTEXT or a  
1586 single ENROL (carried in the SOAP header) with the Application Message(s) carried in the  
1587 SOAP body.

## 1588 **5.6 Other elements**

### 1589 **5.6.1 Identifiers**

1590 An Identifier is a globally unambiguous identification of the state corresponding to one of  
1591 Decider, Superior or Inferior. Where a single entity has more than one of these roles (at the same  
1592 BTP Node in the same transaction, as with a Sub-coordinator that is both Superior and Inferior),  
1593 the Identifiers may be the same or different, at implementation option - they are distinguished by  
1594 which messages the Identifier is used on. (A Superior has only one Superior-identifier, although it  
1595 may be in multiple Superior:Inferior relationships, each with a separate state in terms of the state  
1596 table).

1597 The state identified by an Identifier can be accessed by BTP messages sent to any of the addresses  
1598 supplied with the Identifier in the appropriate message (CONTEXT, BEGUN, ENROL), or as  
1599 updated by REDIRECT. An Identifier itself has no location implications. (Identifiers are

---

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

1600 specified, in the XML representation, as syntactically URIs - by their use as names of BTP  
1601 entities, they are URNs. If an Identifier happens to specify an network location (i.e. it is a URL),  
1602 it is treated as an opaque value by BTP)

1603 Identifiers are specified as being globally unambiguous - the same Identifier only ever identifies  
1604 one Decider, Superior or Inferior over all systems and all time. In practice, an Identifier could be  
1605 re-used if there is no possibility of the colliding values being confused. However implementations  
1606 are recommended to use truly unambiguous Identifiers (that is to use them as URNs).

## 1607 5.6.2 Addresses

1608 In most cases, BTP Actors that need to communicate are informed of each others addresses from  
1609 received BTP messages. When an Inferior is to be enrolled, a CONTEXT message which  
1610 contains the address of the Superior will have been received or otherwise passed to the Enroller  
1611 and the Inferior. The ENROL message received by the Superior contains the address of the  
1612 Inferior. The BEGUN returned from a Factory to the Initiator contains the address of the Decider,  
1613 and this can be passed to the Terminator or any **Status Requestor**.

1614 The addresses carried in these messages (which are effectively “call-back” addresses, to be used  
1615 as the destination of future messages) are sets of tripartite addresses. Each contains an identifier  
1616 (binding name) for the binding to an underlying transport, or Carrier Protocol, a “binding  
1617 address”, in a format specific to the carrier which is the information necessary to connect using  
1618 that carrier, and an optional additional information field. This additional information is opaque to  
1619 all but the future destination (which also created this address for itself) and is used however the  
1620 implementation there wishes (e.g. it can be used to distinguish a particular program object, or to  
1621 relay on, perhaps over a different protocol). The multiple members of the set allow support of  
1622 multiple carrier bindings (including both different versions of standard bindings and proprietary  
1623 bindings) and for relocation of the BTP Actor.

1624 When a message is actually to be sent, the sender, possessing the set of addresses for the  
1625 destination, chooses one - restricting its choice to bindings that it supports obviously, but not  
1626 otherwise constrained by the specification. The binding address will be used by the senders  
1627 carrier implementation (depending on the protocol, the address may or may not be transmitted –  
1628 with http, for example, it is), The additional information, if present, will be included in the BTP  
1629 message. The chosen address is considered the “target-address” when considering the abstract  
1630 message, but only the additional information will normally appear within the encoded BTP-  
1631 message (the encoding used is part of the binding specification, which could require that all of the  
1632 address is (redundantly) transmitted, if the specifier so chose).

1633 Where a BTP message invokes a reply – as with the Initiator:Factory, Terminator:Decider and  
1634 Status Requestor:various roles – the receiver (Factory, Decider, etc) of the message will not know  
1635 *a priori* the address of the sender. Accordingly, in these cases the abstract messages are specified  
1636 as containing a single “reply-address”. Depending on the binding, and the particular use of the  
1637 binding, the “reply-address” may be directly represented in the encoding of the BTP message, or  
1638 may be implicit in the Carrier Protocol. Similar considerations apply in the Superior:Inferior  
1639 relationship, where although the addresses are normally known by the other side, there are cases  
1640 when a message is received, and must be responded to, but the Peer is unknown. Accordingly, the  
1641 Superior:Inferior messages contain (in abstract) a single “senders-address”. As with the “reply-  
1642 address”es, it may be implicit in the Carrier Protocol.

1643 The CONTEXT message does not contain a “target-address”, even as an abstract message, as it is  
1644 never transmitted between BTP Actors on its own – it is always either related to a BTP BEGIN or  
1645 BEGUN message, or is passed between Application Elements with some (application-detailed)  
1646 association with Application Messages.

### 1647 **5.6.3 Qualifiers**

1648 Qualifiers are elements of the BTP messages used to exchange additional information between  
1649 the Actors. Qualifiers can be specified in the BTP specification (“standard qualifiers”), by  
1650 industry groups, by BTP implmentors or for the purposes of particular applications. Of the  
1651 standard qualifiers in this version of the specification some are constraints on the BTP Contract,  
1652 such as time limits, and some are further identifiers used to distinguish specific parties in the BTP  
1653 interchange. Non-standard qualifiers could extend the protocol or carry application-specific  
1654 information.



## 1655 Part 2. Normative Specification of BTP

### 1656 **6 Actors, Roles and Relationships**

1657 Actors are software agents which process computations. BTP Actors are addressable for the  
1658 purposes of receiving application and BTP protocol messages transmitted over some underlying  
1659 communications or carrier protocol. (See section “Addressing” for more detail.)

1660 BTP Actors play roles in the sending, receiving and processing of messages. These roles are  
1661 associated with responsibilities or obligations under the terms of software contracts defined by  
1662 this specification. (These contracts are stated formally in the sections entitled “Abstract Messages  
1663 and Associated Contracts” and “State Tables”.) A BTP Actor’s computations put the contracts  
1664 into effect.

1665 A Role is defined and described in terms of a single Business Transaction. An implementation  
1666 supporting a Role may, as an addressable entity, play the same Role in multiple Business  
1667 Transactions, simultaneously or consecutively, or a separate addressable entity may be created for  
1668 each transaction. This is a choice for the implementer, and the addressing mechanisms allow  
1669 interoperation between implementations that make different choices.

1670 Within a single transaction, one Actor may play several roles, or each Role may be assigned to a  
1671 distinct Actor. This is again a choice for the implementer. An Actor playing a Role is termed an  
1672 “actor-in-role”.

1673 Actors may interoperate, in the sense that the roles played by Actors may be implemented using  
1674 software created by different vendors for each actor-in-role. The section “Conformance”, gives  
1675 guidelines on the groups of roles that may be implemented in a partial, interoperable  
1676 implementation of BTP.

1677 The descriptions of the roles concentrate on the normal progression of a Business Transaction,  
1678 and some of the more important divergences from this. They do not cover all exception cases –  
1679 the message set definition and the state tables provide a more comprehensive specification.

1680 *Note – A BTP Role is approximately equivalent to an interface in some distributed*  
1681 *computing mechanisms, or a port-type in WSDL. The definition of a Role includes*  
1682 *behaviour.*

#### 1683 **6.1 Relationships**

1684 There are two primary relationships in BTP.

- 1685 • Between an Application Element that determines that a Business Transaction should be  
1686 completed (the Role of Terminator) and the BTP Actor at the top of the Transaction Tree  
1687 (the Role of Decider);
- 1688 • Between BTP Actors within the tree, where one (the Superior) will inform the other (the  
1689 Inferior) what the outcome decision is.

1690 These primary relationships are involved in arriving at a decision on the outcome of a Business  
1691 Transaction, and propagating that decision to all parties to the transaction. Taking the path that is  
1692 followed when a Business Transaction is confirmed:

- 1693 1. The Terminator determines that the Business Transaction should Confirm, if it can;  
1694 or (for a Cohesion), which parts should Confirm
- 1695 2. The Terminator asks the Decider to apply the desired outcome to the tree, if it can  
1696 guarantee the consistency of the Confirm decision
- 1697 3. The Decider, which is Superior to one or more Inferiors, asks its Inferiors if they can  
1698 agree to a Confirm decision (for a Cohesion, this may not be all the Inferiors)
- 1699 4. If any of those Inferiors are also Superiors, they ask their Inferiors and so on down  
1700 the tree
- 1701 5. Inferiors that are not Superiors report if they can agree to a Confirm to their Superior
- 1702 6. Inferiors that are also Superiors report their agreement only if they received such  
1703 agreement from their Inferiors, and can agree themselves
- 1704 7. Eventually agreement (or not) is reported to the Decider. If all have agreed, the  
1705 Decider makes and persists the Confirm decision (hence the term “Decider” – it  
1706 decides, everything else just asked); if any have disagreed, or if the Confirm decision  
1707 cannot be persisted, a Cancel decision is made
- 1708 8. The Decider, as Superior tells its Inferiors of the outcome
- 1709 9. Inferiors that are also Superiors tell their Inferiors, recursively down the tree
- 1710 10. The Decider replies to the Terminator’s request to Confirm, reporting the outcome  
1711 decision

1712 There are other relationships that are secondary to Terminator:Decider, Superior:Inferior, mostly  
1713 involved in the establishment of the primary relationships. The various particular relationships  
1714 can be grouped as the “control” relationships – primarily Terminator:Decider, but also  
1715 Initiator:Factory; and the “outcome” relationships – primarily Superior:Inferior, but also  
1716 Enroller:Superior.

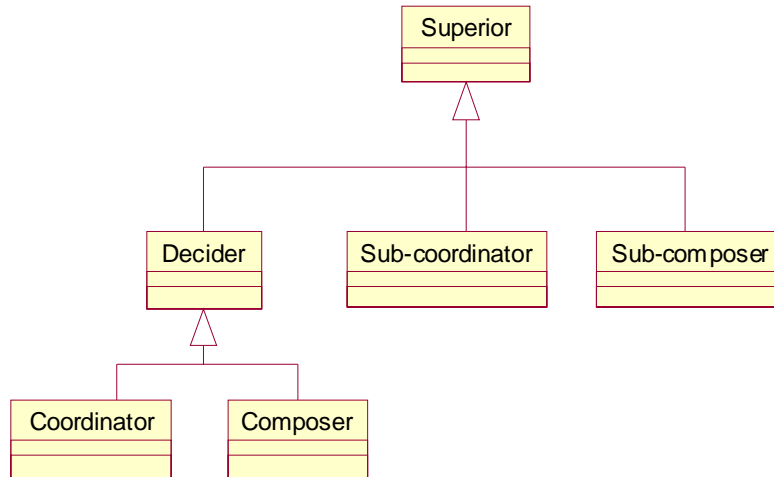
1717 The two groups of relationships are linked in that a Decider is a Superior to one or more Inferiors.  
1718 There are also similarities in the semantics of some of the exchanges (messages) within the  
1719 relationships. However they differ in that

- 1720 1. All exchanges between Terminator and Decider are initiated by the Terminator (it is  
1721 essentially a request/response relationship); either of Superior or Inferior may initiate  
1722 messages to the other
- 1723 2. The Superior:Inferior relationship is recoverable – depending on the progress of the  
1724 relationship, the two sides will re-establish their shared state after failure; the  
1725 Terminator:Decider relationship is not recoverable

1726 3. The nature of the Superior:Inferior relationship requires that the two parties know of  
 1727 each other's addresses from when the relationship is established; the Decider does not  
 1728 need to know the address of the Terminator (provided it has some way of returning  
 1729 the response to a received message).

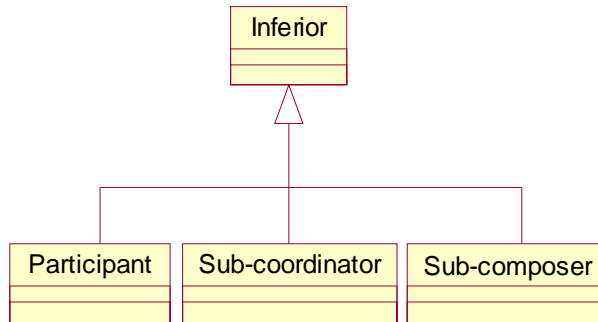
1730 **6.2 Roles**

1731 Figure 6-1 and Figure 6-2 show the BTP roles that are specialisations of the central Superior and  
 1732 Inferior roles.



1733  
 1734

**Figure 6-1 Superior and derived roles**



1735  
 1736

**Figure 6-2 Inferior and derived roles**

1737 In the following sections, the responsibility of each Role is defined, and the messages that are  
 1738 sent or received by that Role are listed. Note that some roles exist only to have a name for an  
 1739 Actor that issues a message and receives a reply to that message. Some of these roles may be  
 1740 played by several Actors in the course of a single Business Transaction.

1741 For each Role, a table shows which messages are received and sent. Where the messages appear  
 1742 on the same line, the second is a reply to the first. (Consequently the columns are sometimes sent  
 1743 first, received second, sometimes vice versa.)

1744 **6.2.1 Roles involved in the Outcome Relationships**

1745 **6.2.2 Superior**

1746 Accepts enrolments of Inferiors from Enrollers, establishing a Superior:Inferior relationship with  
 1747 each. In cooperation with other Actors and constrained by the messages exchanged with the  
 1748 Inferior, the Superior determines the **Outcome** applicable to the Inferior and informs the Inferior  
 1749 by sending CONFIRM or CANCEL. This outcome can be Confirm only if a PREPARED  
 1750 message is received from the Inferior, and if a record, identifying the Inferior can be persisted.  
 1751 (Whether this record is also a record of a Confirm decision depends on the Superior’s position in  
 1752 the Business Transaction as a whole.). The Superior must retain this persistent record until it  
 1753 receives a CONFIRMED (or, in exceptional cases, CANCELLED or HAZARD) from the  
 1754 Inferior.

1755 A Superior may delegate the taking of the Confirm or Cancel decision to an Inferior, if there is  
 1756 only one Inferior, by sending CONFIRM\_ONE\_PHASE.

1757 A Superior may be *Atomic* or *Cohesive*; an Atomic Superior will apply the same decision to all of  
 1758 its Inferiors; a Cohesive Superior may apply Confirm to some Inferiors and Cancel to others, or  
 1759 may Confirm some after others have reported cancellation. The set of Inferiors that the Superior  
 1760 confirms (or attempts to Confirm) is called the “Confirm-set”.

1761 If RESIGN is received from an Inferior, the Superior:Inferior relationship is ended; the Inferior  
 1762 has no further effect on the behaviour of the Superior as a whole.

Superior receives	Superior sends
ENROL	ENROLLED
	PREPARE
	CONFIRM
	CANCEL
	RESIGNED
	CONFIRM_ONE_PHASE
	CONTRADICTION
	SUPERIOR_STATE
PREPARED	
CONFIRMED	
CANCELLED	
HAZARD	
RESIGN	
INFERIOR_STATE	
REQUEST_STATUS	STATUS
REQUEST_INFERIORS_STATUS	INFERIOR_STATUSES

1763

1764 Receipt of ENROL establishes a new Superior:Inferior relationship (unless the ENROL is a  
 1765 duplicate). ENROLLED is sent only if a reply is asked for on the ENROL.

1766 **6.2.3 Inferior**

1767 Responsible for applying the Outcome to some set of associated operations – the application  
 1768 determines which operations are the responsibility of a particular Inferior.

1769 An Inferior is **Enrolled** with a single Superior (hereafter referred to as “its Superior”),  
 1770 establishing a Superior:Inferior relationship. If the Inferior is able to ensure that either a Confirm  
 1771 or Cancel decision can be applied to the associated operations, and can persist information to  
 1772 retain that condition, it sends a PREPARED message to the Superior. When the Outcome is  
 1773 received from the Superior, the Inferior applies it, deletes the persistent information, and replies  
 1774 with CANCELLED or CONFIRMED as appropriate.

1775 If an Inferior is unable to come to a prepared state, it cancels the associated operations and  
 1776 informs the Superior with a CANCELLED message. If it is unable to either come to a prepared  
 1777 state, or to Cancel the associated operations, it informs the Superior with a HAZARD message.

1778 An Inferior that has Become Prepared may, exceptionally, make an autonomous decision to be  
 1779 applied to the associated operations, without waiting for the Outcome from the Superior. It is  
 1780 required to persist this autonomous decision and report it to the Superior with CONFIRMED or  
 1781 CANCELLED as appropriate. If, when CONFIRM or CANCEL is received, the autonomous  
 1782 decision and the decision received from the Superior are contradictory, the Inferior must retain  
 1783 the record of the autonomous decision until receiving a CONTRADICTION message.

Inferior receives	Inferior sends
PREPARE	
CONFIRM	
CANCEL	
RESIGNED	
CONFIRM_ONE_PHASE	
CONTRADICTION	
SUPERIOR_STATE	
	PREPARED
	CONFIRMED
	CANCELLED
	HAZARD
	RESIGN
	INFERIOR_STATE
REQUEST_STATUS	STATUS
REQUEST_INFERIORS_STATUS	INFERIOR_STATUSES

1784

1785 **6.2.4 Enroller**

1786 Causes the enrolment of an Inferior with a Superior. This Role is distinguished because in some  
 1787 implementations the enrolment request will be performed by the application, in some the  
 1788 application will ask the Actor that will play the Role of Inferior to enrol itself, and a Factory may  
 1789 enrol a new Inferior (which will also be Superior) as a result of receiving BEGIN&CONTEXT.

Enroller sends	Enroller receives
ENROL	ENROLLER

1790

1791 ENROLLED is received only if the Enroller asked for a response when the ENROL was sent.

1792 An ENROL message sent from an Enroller that did not require an ENROLLED response may be  
 1793 modified *en route* to the Superior by an intermediate Actor to ask for an ENROLLED response to  
 1794 be sent to the intermediate. (This may occur in the “one-shot” scenario, where an ENROL/no-rsp-  
 1795 req is received in relation to a CONTEXT\_REPLY/related; the receiver of the  
 1796 CONTEXT\_REPLY will need to ensure the enrolment is successful).

### 1797 6.2.5 Participant

1798 An Inferior which is specialized for the purposes of an application. Some Application Operations  
 1799 are associated directly with the Participant, which is responsible for determining whether a  
 1800 prepared condition is possible for them, and for applying the outcome. (“associated directly” as  
 1801 opposed to involving another BTP Superior:Inferior relationship, in which this Actor is the  
 1802 Superior).

1803 The associated operations may be performed by the Actor that has the Role of Participant, or they  
 1804 may be performed by another Actor, and only the Confirm/Cancel application is performed by the  
 1805 Participant.

1806 In either case, the Participant, as part of becoming prepared (i.e. before it can send PREPARED  
 1807 to the Superior), will persist information allowing it apply a Confirm decision to the operations  
 1808 and to apply a Cancel decision. The nature of this information depends on the operations.

1809 *Note – Possible approaches are:*

- 1810 • *The operations may be performed completely and the Participant persists*  
 1811 *information to perform Counter-effect operations (compensating operations) to*  
 1812 *apply cancellation;*
- 1813 • *The operations may be just checked and not performed at all; the Participant*  
 1814 *persists information to perform them to apply confirmation;*
- 1815 • *The Participants persists the prior state of data affected by the operations and the*  
 1816 *operations are performed; the Participant restores the prior state to apply*  
 1817 *cancellation;*
- 1818 • *As the previous, but other access to the affected data is forbidden until the decision*  
 1819 *is known*

1820 Since a Participant is an Inferior, it sends and receives the messages for an Inferior.

### 1821 6.2.6 Sub-coordinator

1822 An Inferior which is also an Atomic Superior.

1823 A sub-coordinator is the Inferior in one Superior:Inferior relationship and the Superior in one or  
 1824 more Superior:Inferior relationships.

1825 From the perspective of its Superior (the one the sub-coordinator is Inferior to), there is no  
1826 difference between a sub-coordinator and any other Inferior. From this perspective, the  
1827 “associated operations” of the sub-coordinator as an Inferior include the relationships with its  
1828 Inferiors.

1829 A sub-coordinator does not Become Prepared (and send PREPARED to its Superior) until and  
1830 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is propagated  
1831 to all Inferiors.

1832 Since a Sub-coordinator is both an Inferior and a Superior, it sends and receives the messages for  
1833 both.

### 1834 **6.2.7 Sub-composer**

1835 An Inferior which is also a Cohesive Superior.

1836 Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from the  
1837 perspective of its Superior.

1838 A sub-composer is similar to a sub-coordinator, except that the constraints linking the different  
1839 Inferiors concern only those Inferiors in the Confirm-set. How the Confirm-set is controlled, and  
1840 when, is not defined in this specification.

1841 If the sub-composer is instructed to Cancel, by receiving a CANCEL message from its Superior,  
1842 the cancellation is propagated to all its Inferiors.

1843 Since a Sub-composer is both an Inferior and a Superior, it sends and receives the messages for  
1844 both.

## 1845 **6.3 Roles involved in the Control Relationships**

### 1846 **6.3.1 Decider**

1847 A Superior that is not also the Inferior on a Superior:Inferior relationship. It is the top BTP node  
1848 in the Transaction Tree and receives requests from a Terminator as to the desired outcome for the  
1849 Business Transaction. If the Terminator asks the Decider to Confirm the Business Transaction, it  
1850 is the responsibility of the Decider to finally take the Confirm decision. The taking of the decision  
1851 is synonymous with the persisting of information identifying the Inferiors that are to be  
1852 confirmed. An Inferior cannot be confirmed unless PREPARED has been received from it.

1853 A Decider is instructed to Cancel by receiving CANCEL\_TRANSACTION.

1854 A Decider that is an Atomic Superior (all Inferiors will have the same outcome) is a Coordinator.  
1855 A Decider that is a Cohesive Superior (some Inferiors may Cancel, some Confirm) is a  
1856 Composer.

Decider receives	Decider sends
CONFIRM_TRANSACTION	TRANSACTION_CONFIRMED TRANSACTION_CANCELLED

Decider receives	Decider sends
	INFERIOR_STATUSES
CANCEL_TRANSACTION	TRANSACTION_CANCELLED INFERIOR_STATUSES
REQUEST_INFERIOR_STATUSES	INFERIOR_STATUSES

1857

1858 A Decider is also a Superior and thus sends and receives the messages for a Superior.

### 1859 6.3.2 Coordinator

1860 A Decider that is an Atomic Superior. The same outcome decision will be applied to all Inferiors  
1861 (excluding any from which RESIGN is received).

1862 PREPARED must be received from all remaining Inferiors for a Confirm decision to be taken.

1863 A Coordinator must make a Cancel decision if

- 1864 • it is instructed to Cancel by the Terminator
- 1865 • if CANCELLED is received from any Inferior
- 1866 • if it is unable to persist a Confirm decision

1867 Since a Coordinator is a Decider, it receives the messages appropriate for a Decider and a  
1868 Superior.

### 1869 6.3.3 Composer

1870 A Decider that is a Cohesive Superior. If the Terminator requests confirmation of the Cohesion,  
1871 that request will determine the Confirm-set of the Cohesion.

1872 PREPARED must be received from all Inferiors in the Confirm-set (excluding any from which  
1873 RESIGN is received) for a Confirm decision to be taken.

1874 A Composer must make a Cancel decision (applying to all Inferiors) if

- 1875 • it is instructed to Cancel by the Terminator
- 1876 • if CANCELLED is received from any Inferior in the Confirm-set
- 1877 • if it is unable to persist a Confirm decision

1878 A Composer may be asked to prepare some or all of its Inferiors by receiving  
1879 PREPARE\_INFERIORS. It issues PREPARE to any of those Inferiors from which none of  
1880 PREPARED, CANCELLED or RESIGN have been received, and replies to the  
1881 PREPARE\_INFERIORS with INFERIOR\_STATUSES.

1882 A Composer may be asked to Cancel some of its Inferiors, but not itself, by receiving  
 1883 CANCEL\_INFERIORS.

Composer receives	Composer sends
PREPARE_INFERIORS	INFERIOR_STATUSES
CANCEL_INFERIORS	INFERIOR_STATUSES

1884 **6.3.4 Terminator**

1885 Asks a Decider to Confirm the Business Transaction, or instructs it to Cancel all or (for a  
 1886 Cohesion) part of the Business Transaction.

1887 All communications between Terminator and Decider are initiated by the Terminator. A  
 1888 Terminator is usually an Application Element.

1889 A request to Confirm is made by sending CONFIRM\_TRANSACTION to the target Decider. If  
 1890 the Decider is a Cohesion Composer, the Terminator may select which of the Composer's  
 1891 Inferiors are to be included in the Confirm-set. If the Decider is an Atom Coordinator, all  
 1892 Inferiors are included. After applying the decision, the Decider replies with  
 1893 TRANSACTION\_CONFIRMED, TRANSACTION\_CANCELLED or (in the case of problems)  
 1894 INFERIOR\_STATUSES.

1895 A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its Inferiors  
 1896 with PREPARE\_INFERIORS. The Composer replies with INFERIOR\_STATUSES.

1897 A Terminator may send CANCEL\_TRANSACTION to instruct the Decider to Cancel the whole  
 1898 Business Transaction.. The Decider replies with CANCEL\_COMPLETE if all Inferiors Cancel  
 1899 successfully, and with INFERIOR\_STATUSES in the case of problems.. If the Decider is a  
 1900 Cohesion Composer, the Terminator may send CANCEL\_INFERIORS to Cancel some of the  
 1901 Inferiors; the Decider always replies with INFERIOR\_STATUSES.

1902 A Terminator may check the status of the Inferiors of the Decider by sending  
 1903 REQUEST\_INFERIOR\_STATUSES. The Decider replies with INFERIOR\_STATUSES.

Terminator sends	Terminator receives
CONFIRM_TRANSACTION	TRANSACTION_CONFIRMED TRANSACTION_CANCELLED INFERIOR_STATUSES
CANCEL_TRANSACTION	TRANSACTION_CANCELLED INFERIOR_STATUSES
PREPARE_INFERIORS	INFERIOR_STATUSES
CANCEL_INFERIORS	INFERIOR_STATUSES
REQUEST_INFERIOR_STATUSES	INFERIOR_STATUSES

1904 **6.3.5 Initiator**

1905 Requests a **Factory** to create a Superior – this will either be a Decider (representing a new top-  
1906 level Business Transaction) or a sub-coordinator or sub-composer to be the Inferior of an existing  
1907 Business Transaction.

Initiator sends	Initiator receives
BEGIN	BEGUN & CONTEXT
BEGIN & CONTEXT	BEGUN & CONTEXT

1908

1909 The received CONTEXT is that for the new Superior.

1910 **6.3.6 Factory**

1911 Creates Superiors and returns the CONTEXT for the new Superior. The following types of  
1912 Superior are created :

1913 Decider, which is either  
1914 Composer or  
1915 Coordinator  
1916 Sub-composer  
1917 Sub-coordinator  
1918

Factory receives	Factory sends
BEGIN	BEGUN & CONTEXT
BEGIN & CONTEXT	BEGUN & CONTEXT

1919

1920 If the BEGIN has no related CONTEXT, the Factory creates a Decider, either a Cohesion  
1921 Composer or an Atom Coordinator, as determined by the “superior type” parameter on the  
1922 BEGIN.

1923 If the BEGIN has a related CONTEXT, the new Superior is also enrolled as an Inferior of the  
1924 Superior identified by the CONTEXT. The new Superior is thus a sub-composer or sub-  
1925 coordinator, as determined by the “superior type” parameter on the BEGIN.

1926 **6.4 Other roles**

1927 **6.4.1 Redirector**

1928 Sends a REDIRECT message to inform a Superior or Inferior that an address previously supplied  
1929 for the Peer (i.e. an Inferior or Superior, respectively) is no longer appropriate, and to supply a  
1930 new address or set of addresses to replace the old one.

1931 A Redirector may send a REDIRECT message in response to receiving a message using the old  
1932 address, or may send REDIRECT at its own initiative.

1933 If a Superior moves from the superior-address in its CONTEXT, or an Inferior moves from the  
 1934 inferior-address in the ENROL message, the implementation **must** ensure that a Redirector  
 1935 catches any inbound messages using the old address and replies with a REDIRECT message  
 1936 giving the new address. (Note that the inbound message may itself be a REDIRECT message, in  
 1937 which case the Redirector shall use the new address in the received message as the target for the  
 1938 REDIRECT that it sends.)

1939 After receiving a REDIRECT message, the BTP Actor **must** use the new address not the old one,  
 1940 unless failure prevents it updating its information.

Redirector receives	Redirector sends
Any message for Superior or Inferior	REDIRECT

#### 1941 6.4.2 Status Requestor

1942 Requests and receives the current status of a Transaction Tree Node – any of an Inferior, Superior  
 1943 or Decider, or the current status of the nodes relationships with its Inferiors, if any. The Role of  
 1944 Status Requestor has no responsibilities – it is just a name for where the REQUEST\_STATUS  
 1945 and REQUEST\_INFERIOR\_STATUSES comes from (REQUEST\_INFERIOR\_STATUSES is  
 1946 also issued by a Terminator to a Decider).

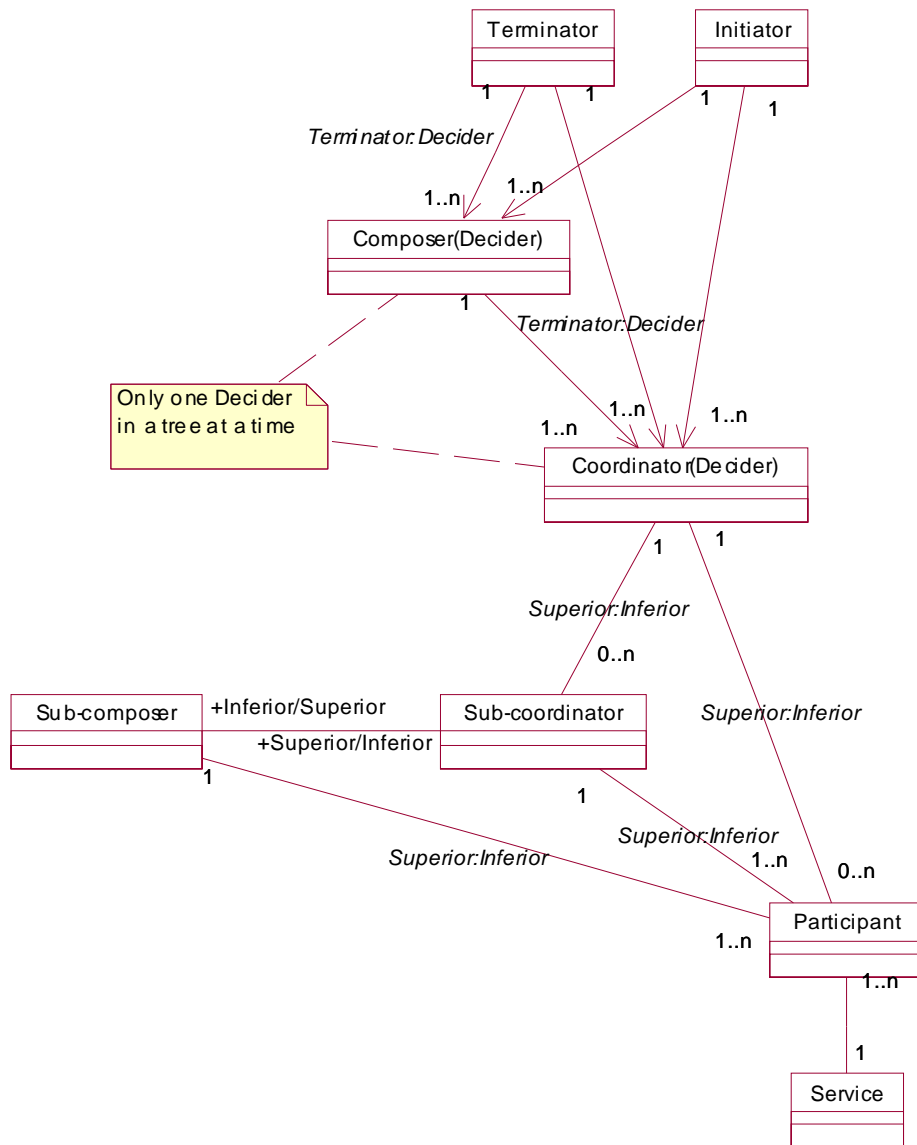
Status Requestor sends	Status Requestor receives
REQUEST_STATUS	STATUS
REQUEST_INFERIOR_STATUS	INFERIOR_STATUSES

1947

1948 The receiver of the request can refuse to provide the status information by replying with  
 1949 FAULT(StatusRefused). The information returned in STATUS will always relate to the  
 1950 Transaction Tree Node as a whole (e.g. as an Inferior, even if it is also a Superior).

1951 **6.5 Summary of relationships**

1952 Figure 6-3 summarises the relationships between the BTP roles. BTP can be implemented using  
 1953 proprietary equivalents of the Terminator and Decider roles.



1954  
 1955

**Figure 6-3 Summary of relationships between roles**

## 1956 **7 Abstract Messages and Associated Contracts**

1957 BT Protocol Messages are defined in this section in terms of the abstract information that has to  
1958 be communicated. These abstract messages will be mapped to concrete messages communicated  
1959 by a particular Carrier Protocol (there can be several such mappings defined).

1960 The abstract message set and the associated state table assume the Carrier Protocol will

- 1961 • deliver messages completely and correctly, or not at all (corrupted messages will not be  
1962 delivered);
- 1963 • report some communication failures, but will not necessarily report all (i.e. not all  
1964 message deliveries are positively acknowledged within the carrier);
- 1965 • sometimes deliver successive messages in a different order than they were sent; and
- 1966 • does not have built-in mechanisms to link a request and a response

1967 Note that these assumptions would be met by a mapping to SMTP and more than met by  
1968 mappings to SOAP/HTTP.

1969 However, when the abstract message set is mapped to a Carrier Protocol that provides a richer  
1970 service (e.g. reports all delivery failures, guarantees ordered delivery or offers a request/response  
1971 mechanism), the mapping can take advantage of these features. Typically in such cases, some of  
1972 the parameters of an abstract message will be implicit in the carrier mechanisms, while the values  
1973 of other parameters will be directly represented in transmitted elements.

1974 The abstract messages include **Delivery Parameters** that are concerned with the transmission and  
1975 delivery of the messages as well as **Payload Parameters** directly concerned with the progression  
1976 of the BTP relationships. When bound to a particular Carrier Protocol and for particular  
1977 implementation configurations, parts or all of the Delivery Parameters may be implicit in the  
1978 Carrier Protocol and will not appear in the "on-the-wire" representation of the BTP messages as  
1979 such. Delivery Parameters are defined as being only those parameters that are concerned with the  
1980 transmission of this message, or of an immediate reply (thus address parameters to be used in  
1981 repeated later messages and the identifiers of both sender and receiver are Payload Parameters).  
1982 In the tables in this section, Delivery Parameters are shown in shaded cells.

### 1983 **7.1 Addresses**

1984 All of the messages except CONTEXT have a "target address" parameter and many also have  
1985 other address parameters. These latter identify the desired target of other messages in the set. In  
1986 all cases, the exact value will have been originally determined by the implementation that is the  
1987 target or intended target.

1988 The detailed format of the address will depend on the particular Carrier Protocol, but at this  
1989 abstract level is considered to have three parts. The first part, the "binding name", identifies the  
1990 binding to a particular Carrier Protocol – some bindings are specified in this document, others can  
1991 be specified elsewhere. The second part of the address, the "binding address", is meaningful to  
1992 the Carrier Protocol itself, which will use it for the communication (i.e. it will permit a message

1993 to be delivered to a receiver). The third part, “additional information”, is not used or understood  
1994 by the Carrier Protocol. The “additional information” may be a structured value.

1995 When a message is actually transmitted, the “binding name” of the target address will identify  
1996 which Carrier Protocol is in use and the “binding address” will identify the destination, as known  
1997 to the Carrier Protocol. The entire binding address is considered to be “consumed” by the Carrier  
1998 Protocol implementation. All of it may be used by the sending implementation, or some of it may  
1999 be transmitted in headers, or as part of a URL in the Carrier Protocol, but then used or consumed  
2000 by the receiving implementation of the Carrier Protocol to direct the BTP message to a BTP-  
2001 aware entity (BTP-aware in that it is capable of interpreting the BTP messages). The “additional  
2002 information” of the target address will be part of the BTP message itself and used in some way by  
2003 the receiving BTP-aware entity (it could be used to route the message on to some other BTP  
2004 entity). Thus, for the target address, only the “additional information” field is transmitted in the  
2005 BTP message and the “additional information” is opaque to parties other than the recipient.

2006 For other addresses in BTP messages, all three components will be within the message.

2007 All messages that concern a particular Superior:Inferior relationship have an identifier parameter  
2008 for the target side as well as the target address. This allows full flexibility for implementation  
2009 choices – an implementation can:

2010 a) Use the same binding address and additional information for multiple Business  
2011 Transactions, using the identifier parameter to locate the relevant state  
2012 information;

2013 b) Use the same binding address for multiple Business Transactions and use the  
2014 additional information to locate the information; or

2015 c) Use a different binding address for each Business Transaction.

2016 Which of these choices is used is opaque to the entity sending the message – both parts of the  
2017 address and the identifier originated at the recipient of this message (and were transmitted as  
2018 parameters of earlier messages in the opposite direction).

2019 BTP recovery requires that the state information for a Superior or Inferior is accessible after  
2020 failure and that the Peer can distinguish between temporary inaccessibility and the permanent  
2021 non-existence of the state information. As is explained in “5.4.4 Redirection” in the conceptual  
2022 model, BTP provides mechanisms – having a set of **BTP Addresses** for some parameters, and the  
2023 REDIRECT message – that make this possible, even if the recovered state information is on a  
2024 different address to the original one (as may be the case if case c) above is used).

## 2025 **7.2 Request/response pairs**

2026 Many of the messages combine in pairs as a request and its response. However, in some cases the  
2027 response message is sent without a triggering request, or as a possible response to more than one  
2028 type of request. To allow for this, the abstract message set treats each message as standalone; but  
2029 where a request does expect a reply, a “reply-address” parameter will be present. For any  
2030 message with a reply address parameter, in the case of certain errors, a FAULT message will be  
2031 sent to the reply address instead of the expected reply.

2032 Between Superior and Inferior the address of the Peer is normally known (from the “superior-  
2033 address” on an earlier CONTEXT or the “inferior-address” on a received ENROL). However, in  
2034 some cases a message will be received for a Superior or Inferior that is not known – the state  
2035 information no longer exists. This is not an exceptional condition but occurs when one side has  
2036 either not created or has removed its persistent state in accordance with the procedures, but a  
2037 message has got lost in a failure, and the Peer still has state information. The response to a  
2038 message for an unknown (and logically non-existent) Superior is SUPERIOR\_STATE/unknown,  
2039 for an unknown Inferior it is INFERIOR\_STATE/unknown. However, since the intended target is  
2040 unknown, there is no information to locate the Peer, which sent the undeliverable message. To  
2041 enable the receiver to reply with the appropriate \*\_STATE/unknown, all the messages between  
2042 Superior and Inferior have a “senders-address” parameter. If a FAULT message is to be sent in  
2043 response to message which (as an abstract message) has a “senders-address” parameter, the  
2044 FAULT message is sent to that address.

2045 *Note – Both reply-address and senders-address may be absent when the Carrier Protocol*  
2046 *itself has a request/response pattern. In these cases, the reply or sender address is*  
2047 *implicitly that of the sender of the request (and thus the destination of a response)*

### 2048 **7.3 Compounding messages**

2049 BTP messages may be sent in combination with each other, or with other (application) messages.  
2050 There are two cases:

- 2051 a) Sending the messages together where the combination has semantic  
2052 significance. One message is said to be “related to” the other – the combination  
2053 is termed a “group”.
- 2054 b) Sending of the messages where the combination has no semantic significance,  
2055 but is merely a convenience or optimisation. This is termed “bundling” – the  
2056 combination is termed a “bundle”.

2057 The form A&B is used to refer to a combination (group) where message B is sent in relation to A  
2058 (“relation” is asymmetric). The form A+B is used to refer to A and B bundled together- the  
2059 transmission of the bundle "A+B" is semantically identical to the transmission of A followed by  
2060 the transmission of B.

2061 Only certain combinations of messages are possible in a group, and the meaning of the relation is  
2062 specifically defined for each such combination in the next section. A particular group is treated as  
2063 a unit for transmission – it has a single target address. This is usually that of one of the messages  
2064 in the group – the specification for the group defines which.

2065 A “bundle” of messages may contain both unrelated messages and groups of related messages.  
2066 The only constraint on which messages and groups can be bundled is that all have the same  
2067 binding address, but may have different “additional information” values. (Messages within a  
2068 related group may have different addresses, where the rules of their relatedness permit this).  
2069 Unless constrained by the binding, any messages or groups that are to be sent to the same binding  
2070 address may be bundled – the fact that the binding addresses are the same is a necessary and  
2071 sufficient condition for the sender to determine that the messages can be bundled.

2072 A particular and important case of related messages is where a BTP CONTEXT message is sent  
2073 related to an Application Message. In this case, the target of the Application Message defines the  
2074 destination of the CONTEXT message. The receiving implementation may in fact remove the  
2075 CONTEXT before delivering the Application Message to the application (Service) proper, but  
2076 from the perspective of the sender, the two are sent to the same place.

2077 The compounding mechanisms, and the multi-part address structures, support the “one-wire” and  
2078 “one-shot” communication patterns.

2079 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,  
2080 including the associated Application Messages, pass via the same “endpoints”. These “endpoints”  
2081 may in fact be relays, routing messages on to particular Actors within their domain. The onward  
2082 routing will require some further addressing, but this has to be opaque to the sender. This can be  
2083 achieved if the relaying endpoint ensures that all addresses for Actors in its domain have the  
2084 relay’s address as their binding address, and any routing information it will need in its own  
2085 domain is placed in the additional information. (This may involve the relay changing addresses in  
2086 messages as they pass through it on the way out). On receiving a message, it determines the  
2087 within-domain destination from the received additional information (which is thus rewritten) and  
2088 forwards the message appropriately. The sender is unaware of this, and merely sees addresses  
2089 with the same binding address, which it is permitted to bundle. The content of the “additional  
2090 information” is a matter only for the relay – it could put an entire BTP Address in there, or other  
2091 implementation-defined information. Note that a quite different one-wire implementation can be  
2092 constructed where there is no relaying, but the receiving entity effectively performs all roles,  
2093 using the received identifiers to locate the appropriate state.

2094 “One-shot” communication makes it possible to send an Application Message, receive the  
2095 application reply, enrol an Inferior to be responsible for the Confirm/Cancel of the operations of  
2096 those message and inform the Superior that the Inferior is prepared, all in one two-way exchange  
2097 across the network (e.g. one request/reply of a Carrier Protocol).. The application request is sent  
2098 with a related CONTEXT message. The application response is sent with a relation group of  
2099 CONTEXT\_REPLY/related, ENROL/no-rsp-req message and a PREPARED message. This is  
2100 possible even if the Superior address is different from the address of the Application Element that  
2101 sends the original message (if the application exchange is request/reply, there may not even be an  
2102 identifiable address for the Application Element). The target addresses of the ENROL and  
2103 PREPARED (the Superior address) are not transmitted; the Actor that was originally responsible  
2104 for adding the CONTEXT to the outbound Application Message remembers the Superior address  
2105 and forwards the ENROL and PREPARED appropriately.

2106 With “one-shot”, if there are multiple Inferiors created as a result of a single Application  
2107 Message, there is an ENROL and PREPARED message for each sent related to the  
2108 CONTEXT\_REPLY. If an operation fails, a CANCELLED message is sent instead of a  
2109 PREPARED.

2110 If the CONTEXT has “superior-type” of “atom”, then subsequent messages to the same Service,  
2111 with the same related CONTEXT/atom, can have their associated operations put under the control  
2112 of the same Inferior, and only a CONTEXT\_REPLY/completed is sent back with the response (if  
2113 the new operations fail, it will be necessary to send back CONTEXT\_REPLY/repudiated, or send  
2114 CANCELLED). If the “superior type” on the CONTEXT is “cohesive”, each operation will  
2115 require separate enrolment.

2116 Whether the “one-shot” mechanism is used is determined by the implementation on the  
2117 responding (Inferior) side. This may be subject to configuration and may also be constrained by  
2118 the application or by the binding in use.

## 2119 **7.4 Extensibility**

2120 To simplify interoperation between implementations of this edition of BTP with implementations  
2121 of future editions, the “must-be-understood” sub-parameter as specified for Qualifiers may be  
2122 defined for use with any parameter added to an existing message in a future revision of this  
2123 specification. The default for “must-be-understood” shall be “true”, so an implementation  
2124 receiving an unrecognised parameter without a “false” value for “must-be-understood” shall not  
2125 accept it (the FAULT value “UnrecognisedParameter” is available, but other errors, including  
2126 lower-layer parsing/unmarshalling errors may be reported instead). If “must-be-understood” with  
2127 the value “false” is present as a sub-parameter of a parameter in any message, a receiving  
2128 implementation **should** ignore the parameter.

2129 How the sub-parameter is associated with the new parameter is determined by the particular  
2130 binding.

2131 No special mechanism is provided to allow for the introduction of completely new messages.

## 2132 **7.5 Messages**

### 2133 **7.5.1 Qualifiers**

2134 All messages have a Qualifiers parameter which contains zero or more Qualifier values. A  
2135 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

2136

2137 **Qualifier group** ensures the Qualifier name is unambiguous. Qualifiers in the same group  
2138 need not have any functional relationship. The qualifier group will typically be used to  
2139 identify the specification that defines the qualifier’s meaning and use. Qualifiers may  
2140 be defined in this or other standard specifications, in specifications of a particular  
2141 community of users or of implementations or by bilateral agreement.

2142 **Qualifier name** this identifies the meaning and use of the Qualifier, using a name that is  
2143 unambiguous within the scope of the Qualifier group.

2144 **Must-be-understood** if this has the value “true” and the receiving entity does not  
2145 recognise the Qualifier type (or does not implement the necessary functionality), a  
2146 FAULT “UnsupportedQualifier” shall be returned and the message shall not be  
2147 processed. Default is “true”.

2148 **To-be-propagated** if this has the value “true” and the receiving entity passes the BTP  
2149 message (which may be a CONTEXT, but can be other messages) onwards to other  
2150 entities, the same Qualifier value shall be included. If the value is “false”, the Qualifier  
2151 shall not be automatically included if the BTP message is passed onwards. (If the  
2152 receiving entity does support the qualifier type, it is possible a propagated message  
2153 may contain another instance of the same type, even with the same Content – this is  
2154 not considered propagation of the original qualifier.). Default is “false”.

2155 **Content** the type (which may be structured) and meaning of the content is defined by the  
2156 specification of the Qualifier.

## 2157 **7.6 Messages not restricted to outcome or Control Relationships.**

2158 The messages in this section are used between various roles. CONTEXT message is used in the  
2159 Initiator:Factory relationship (when it is related to BEGIN or to BEGUN), and related to an  
2160 application ‘message’ to propagate the Business Transaction between parts of the  
2161 application. CONTEXT\_REPLY is used as the reply to a CONTEXT.REQUEST\_STATUS can  
2162 be issued to, and STATUS returned by any of Decider, Superior or Inferior. FAULT can be used  
2163 on any relationship to indicate an error condition back to the sender of a message.

### 2164 **7.6.1 CONTEXT**

2165 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more Application  
2166 Messages. (The means by which this relationship is represented is determined by the binding and  
2167 the binding mechanisms of the Application Protocol.) The “superior-type” parameter identifies  
2168 whether the Superior will apply the same decision to all Inferiors enrolled using the same superior  
2169 identifier (“superior-type” is “atom”) or whether it may apply different decisions (“superior-type”  
2170 is “cohesion”).

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

2171

2172 **superior-address** the address to which ENROL and other messages from an enrolled  
2173 Inferior are to be sent. This can be a set of alternative addresses.

2174 **superior-identifier** identifies the Superior. This shall be globally unambiguous.

2175 **superior-type** identifies whether the CONTEXT refers to a Cohesion or an Atom. Default  
2176 is atom.

2177 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction timelimit”  
2178 is carried by CONTEXT.

2179 **reply-address** the address to which a replying CONTEXT\_REPLY is to be sent. This may  
2180 be different each time the CONTEXT is transmitted – it refers to the destination of a  
2181 replying CONTEXT\_REPLY for this particular transmission of the CONTEXT.

2182 There is no “target-address” parameter for CONTEXT as it is only transmitted in relation to the  
2183 Application Messages, BEGIN and BEGUN.

2184 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the  
2185 “superior-type” with the appropriate value.

## 2186 7.6.2 CONTEXT\_REPLY

2187 CONTEXT\_REPLY is sent after receipt of CONTEXT (related to Application Message(s)) to  
2188 indicate whether all necessary enrolments have already completed (ENROLLED has been  
2189 received) or will be completed by ENROL messages sent in relation to the CONTEXT\_REPLY  
2190 or if an enrolment attempt has failed. CONTEXT\_REPLY may be sent related to an Application  
2191 Message (typically the response to the Application Message related to the CONTEXT). In some  
2192 bindings the CONTEXT\_REPLY may be implicit in the Application Message.  
2193 CONTEXT\_REPLY is used in some of the related groups to allow BTP messages to be sent to a  
2194 Superior with an Application Message.

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

2195

2196 **superior-identifier** the “superior-identifier” from the CONTEXT

2197 **completion-status:** reports whether all enrol operations made necessary by the receipt of  
2198 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

Value	meaning
	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 <b>not</b> been honoured.

2199

2200       **qualifiers** standardised or other qualifiers.

2201       **target-address** the address to which the CONTEXT\_REPLY is sent. This shall be the  
2202       “reply-address” from the CONTEXT.

2203       The form CONTEXT\_REPLY/completed, CONTEXT\_REPLY/related and  
2204       CONTEXT\_REPLY/repudiated refer to CONTEXT\_REPLY messages with status having the  
2205       appropriate value. The form CONTEXT\_REPLY/ok refers to either of  
2206       CONTEXT\_REPLY/completed or CONTEXT\_REPLY/related.

2207       If there are no necessary enrolments (e.g. the Application Messages related to the received  
2208       CONTEXT did not require the enrolment of any Inferiors), then CONTEXT\_REPLY/completed  
2209       is used.

2210       If a CONTEXT\_REPLY/repudiated is received, the receiving implementation **must** ensure that  
2211       the Business Transaction will not be confirmed.

### 2212       7.6.3   REQUEST\_STATUS

2213       Sent to an Inferior, Superior or to a Decider to ask it to reply with STATUS. The receiver may  
2214       reject the request with a FAULT(StatusRefused).

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

2215

2216       **target identifier** The identifier for the Business Transaction, or part of Business  
2217       Transaction whose status is sought. If the target-address is a “decider-address”, this  
2218       parameter shall be the “transaction-identifier” on the BEGUN message. If the “target-  
2219       address” is an “inferior-address”, this parameter shall be the “inferior-identifier” on the  
2220       ENROL message. If the “target-address” is a “superior-address”, this parameter shall  
2221       be the “superior-identifier” on the CONTEXT.

- 2222       **qualifiers** standardised or other qualifiers.
- 2223       **target-address** the address to which the REQUEST\_STATUS message is sent. This can  
2224           be any of “decider-address”, “inferior-address” or “superior-address”.
- 2225       **reply-address** the address to which the replying STATUS should be sent.
- 2226       Types of FAULT possible (sent to “reply-address”)
- 2227        ***General***
- 2228        ***Redirect** – if the intended target now has a different address*
- 2229        ***StatusRefused** – if the receiver is not prepared to report its status to the sender of this*  
2230           *message*
- 2231        ***UnknownTransaction** – if the target-identifier is unknown*

2232       **7.6.4 STATUS**

2233       Sent by a Inferior, Superior or Decider in reply to a REQUEST\_STATUS, reporting the overall  
2234       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

- 2235
- 2236       **responders-identifier** the identifier of the state, identical to the “target-identifier” on the  
2237       REQUEST\_STATUS.
- 2238       **status** states the current status of the Transaction Tree Node represented by the sender.  
2239       Some of the values are only issued if the sender is an Inferior. If the Transaction Tree  
2240       Node is both Superior and Inferior (i.e. is a sub-coordinator or sub-composer), and two  
2241       status values would be valid for the current state, it is the sender’s option which one is  
2242       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

<b>status value</b>	<b>Meaning from Superior</b>	<b>Meaning from Inferior</b>
<i>Active</i>	New enrolment of inferiors is possible	The Inferior is enrolled
<i>Resigning</i>	Not applicable	RESIGN has been sent; RESIGNED is awaited
<i>Resigned</i>	Not applicable	RESIGNED has been received
<i>Preparing</i>	Not applicable	PREPARE has been received; PREPARED has not been sent
<i>Prepared</i>	Not applicable	PREPARED has been sent; no outcome has been received or autonomous decision made
<i>Confirming</i>	Confirm decision has been made or CONFIRM has been received as Inferior but responses from inferiors are pending	CONFIRM has been received; CONFIRMED/response has not been sent
<i>Confirmed</i>	CONFIRMED/responses have been received from all Inferiors	CONFIRMED/response has been sent
<i>Cancelling</i>	Cancel decision has been made but responses from inferiors are pending	CANCEL has been received or auto-cancel has been decided
<i>Cancelled</i>	CANCELLED has been received from all Inferiors	CANCELLED has been sent
<i>Cancel-contradiction</i>	Not applicable	Autonomous Cancel decision was made, CONFIRM received; CONTRADICTION has not been received
<i>Confirm-contradiction</i>	Not applicable	Autonomous confirm decision was made, CANCEL received; CONTRADICTION has not been received
<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

2244           **qualifiers** standardised or other qualifiers.

2245           **target-address** the address to which the STATUS is sent. This will be the “reply-address”  
 2246           on the REQUEST\_STATUS message

2247   Types of FAULT possible

2248            *General*

2249   **7.6.5 FAULT**

2250   Sent in reply to various messages to report an error condition . The FAULT message is used on  
 2251   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

2252

2253           **superior-identifier** the “superior-identifier” as on the CONTEXT message and as used on  
 2254           the ENROL message (present only if the FAULT is sent to the superior).

2255           **inferior-identifier** the “inferior-identifier” as on the ENROL message (present only if the  
 2256           FAULT is sent to the inferior)

2257           **fault-type** identifies the nature of the error, as specified for each of the main messages.

2258           **fault-data** information relevant to the particular error. Each “fault-type” defines the  
 2259           content of the “fault-data”:

<b>fault-type</b>	<b>meaning</b>	<b>fault-data</b>
<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	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
<i>Redirect</i>	The target of the BTP message now has a different address	Set of BTP Addresses, to be used instead of the address the BTP message was received on

2260

2261 **fault-text** Free text describing the fault or providing more information. Whether this  
2262 parameter is present, and exactly what it contains are an implementation option.

2263 **qualifiers** standardised or other qualifiers.

2264 **target-address** the address to which the FAULT is sent. This may be the “reply-address”  
2265 from a received message or the address of the opposite side (superior/inferior) as given  
2266 in a CONTEXT or ENROL message

2267 *Note – If the carrier mechanism used for the transmission of BTP messages is capable of*  
2268 *delivering messages in a different order than they were sent in, the “WrongState”*  
2269 *FAULT is not sent and should be ignored if received.*

## 2270 7.6.6 REQUEST\_INFERIOR\_STATUSES, INFERIOR\_STATUSES

2271 REQUEST\_INFERIOR\_STATUSES may be sent to and INFERIOR\_STATUSES sent from any  
2272 Decider, Superior or Inferior, asking it to report on the status of its relationships with Inferiors (if  
2273 any). Since Deciders are required to respond to REQUEST\_INFERIOR\_STATUSES with  
2274 INFERIOR\_STATUSES but non-Deciders may just issue FAULT(StatusRefused), and  
2275 INFERIOR\_STATUSES is also used as a reply to other messages from Terminator to Decider,  
2276 these messages are described below under the messages used in the Control Relationships.

## 2277 7.7 Messages used in the Outcome Relationships

### 2278 7.7.1 ENROL

2279 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a  
2280 CONTEXT message in relation to an application request.

2281 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

2282

2283 **superior-identifier**. The “superior-identifier” as on the CONTEXT message

2284 **response- requested** true if an ENROLLED response is required, false otherwise. Default  
2285 is false.

2286 **inferior-address** the address to which PREPARE, CONFIRM, CANCEL and  
2287 SUPERIOR\_STATE messages for this Inferior are to be sent.

2288 **inferior-identifier** an identifier that identifies this Inferior. This shall be globally  
2289 unambiguous..

2290 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior name” may be  
2291 present.

2292 **target-address** the address to which the ENROL is sent. This will be the “superior-  
2293 address” from the CONTEXT message.

2294 **reply-address** the address to which a replying ENROLLED is to be sent, if “response-  
2295 requested” is true. If this field is absent and “response-requested” is true, the  
2296 ENROLLED should be sent to the “inferior-address” (or one of them, at sender’s  
2297 option)

2298 Types of FAULT possible (sent to “reply-address”)

2299 *General*

2300 *InvalidSuperior* – if “superior-identifier” is unknown

2301 *Redirect* – if the Superior now has a different superior-address

2302 *DuplicateInferior* – if inferior with at least one of the set “inferior-address” the same and  
2303 the same “inferior-identifier” is already enrolled

2304 *WrongState* – if it is too late to enrol new Inferiors (generally if the Superior has already  
2305 sent a PREPARED message to its superior or terminator, or if it has already issued  
2306 CONFIRM to other Inferiors).

2307 The form ENROL/rsp-req refers to an ENROL message with “response-requested” having the  
2308 value “true”; ENROL/no-rsp-req refers to an ENROL message with “response-requested” having  
2309 the value “false”

2310 ENROL/no-rsp-req is typically sent in relation to CONTEXT\_REPLY/related. ENROL/rsp-req is  
2311 typically when CONTEXT\_REPLY/completed will be used (after the ENROLLED message has  
2312 been received.)

## 2313 7.7.2 ENROLLED

2314 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been  
2315 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

2316

2317 **inferior-identifier** The “inferior-identifier” as on the ENROL message

2318           **qualifiers** standardised or other qualifiers.

2319           **target-address** the address to which the ENROLLED is sent. This will be the “reply-  
 2320           address” from the ENROL message (or one of the “inferior-address”s if the “reply-  
 2321           address” was empty)

2322           **sender-address** the address from which the ENROLLED is sent. This is an address of the  
 2323           Superior.

2324   No FAULT messages are issued on receiving ENROLLED.

2325   **7.7.3 RESIGN**

2326   Sent from an enrolled Inferior to the Superior to remove the Inferior from the enrolment. This can  
 2327   only be sent if the operations of the Business Transaction have had no effect as perceived by the  
 2328   Inferior.

2329   RESIGN may be sent at any time prior to the sending of a PREPARED or CANCELLED  
 2330   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

2331

2332           **superior-identifier** The “superior-identifier” as on the ENROL message

2333           **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message

2334           **response-requested** is set to “true” if a RESIGNED response is required. Default is  
 2335           “false”.

2336           **qualifiers** standardised or other qualifiers.

2337           **target-address** the address to which the RESIGN is sent. This will be the superior address  
 2338           as used on the ENROL message.

2339           **sender-address** the address from which the RESIGN is sent. This is an address of the  
 2340           Inferior.

2341           *Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued*  
 2342           *early.*

2343 Types of FAULT possible (sent to “sender-address”)

2344 **General**

2345 **InvalidSuperior** – if “superior-identifier” is unknown

2346 **InvalidInferior** – if no ENROL had been received for this “inferior-identifier”inferior-

2347 **WrongState** – if a PREPARED or CANCELLED has already been received by the  
2348 Superior from this Inferior

2349 The form RESIGN/rsp-req refers to an RESIGN message with “response-requested” having the  
2350 value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “response-requested”  
2351 having the value “false”

#### 2352 7.7.4 RESIGNED

2353 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

2354

2355 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this  
2356 Inferior.

2357 **qualifiers** standardised or other qualifiers.

2358 **target-address** the address to which the RESIGNED is sent. This will be the “inferior-  
2359 address” from the ENROL message.

2360 **sender-address** the address from which the RESIGNED is sent. This is an address of the  
2361 Superior.

2362 After receiving this message the Inferior will not receive any more messages with this “inferior-  
2363 identifier”.

2364 Types of FAULT possible (sent to “sender-address”)

2365 **General**

2366 **WrongState** - if RESIGN has not been sent

2367 **7.7.5 PREPARE**

2368 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor RESIGN  
2369 have been received, requesting a PREPARED message. PREPARE can be sent after receiving a  
2370 PREPARED message.

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

2371

2372 **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.

2373 **qualifiers** standardised or other qualifiers. The standard qualifier “Minimum inferior  
2374 timeout” is carried by PREPARE.

2375 **target-address** the address to which the PREPARE message is sent. This will be the  
2376 “inferior-address” from the ENROL message.

2377 **sender-address** the address from which the PREPARE is sent. This is an address of the  
2378 Superior.

2379 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or  
2380 RESIGN.

2381 Types of FAULT possible (sent to “sender-address”)

2382 **General**

2383 **InvalidInferior** – if “inferior-identifier” is unknown

2384 **WrongState** – if a CONFIRM or CANCEL has already been received by this Inferior.

2385 **7.7.6 PREPARED**

2386 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when the  
2387 Inferior has determined the operations associated with the Inferior can be confirmed and can be  
2388 cancelled, as may be instructed by the Superior. The level of isolation is a local matter (i.e. it is  
2389 the Inferiors choice, as constrained by the shared understanding of the application exchanges) –  
2390 other access may be blocked, may see applied results of operations or may see the original state.

Parameter	Type
superior-identifier	Identifier

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

2391

2392

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

2393

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

2394

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

2395

2396

2397

2398

2399

2400

2401

2402

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

2403

2404

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

2405

2406

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

2407

2408

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

2409

2410

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

2411

2412

2413

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

2414

**General**

2415

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

2416

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

2417

2418 The form PREPARED/cancel refers to a PREPARED message with “default-is cancel” = “true”.  
2419 The unqualified form PREPARED refers to a PREPARED message with “default-is cancel” =  
2420 “false”.

### 2421 7.7.7 CONFIRM

2422 Sent by the Superior to an Inferior from whom PREPARED has been received.

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

2423

2424 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this  
2425 Inferior.

2426 **qualifiers** standardised or other qualifiers.

2427 **target-address** the address to which the CONFIRM message is sent. This will be the  
2428 “inferior-address” from the ENROL message.

2429 **sender-address** the address from which the CONFIRM is sent. This is an address of the  
2430 Superior.

2431 On receiving CONFIRM, the Inferior is released from its promise to be able to undo the  
2432 operations of associated with the Inferior. The effects of the operations can be made available to  
2433 everyone (if they weren’t already).

2434 Types of FAULT possible (sent to “sender-address”)

2435 **General**

2436 **InvalidInferior** – if “inferior-identifier” is unknown

2437 **WrongState** – if no PREPARED has been sent by, or if CANCEL has been received by  
2438 this Inferior.

### 2439 7.7.8 CONFIRMED

2440 Sent after the Inferior has applied the confirmation, both in reply to CONFIRM or when the  
2441 Inferior has made an autonomous Confirm decision, and in reply to a CONFIRM\_ONE\_PHASE  
2442 if the Inferior decides to Confirm its associated operations.

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

2443

2444 **superior-identifier** the “superior-identifier” as on the CONTEXT message.

2445 **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.

2446 **confirm-received** “true” if CONFIRMED is sent after receiving a CONFIRM message;  
 2447 “false” if an autonomous Confirm decision has been made and either if no CONFIRM  
 2448 message has been received or the implementation cannot determine if CONFIRM has  
 2449 been received (due to loss of state information in a failure).

2450 **qualifiers** standardised or other qualifiers.

2451 **target-address** the address to which the CONFIRMED is sent. This will be the Superior  
 2452 address as on the CONTEXT message.

2453 **sender-address** the address from which the CONFIRMED is sent. This is an address of  
 2454 the Inferior.

2455 Types of FAULT possible (sent to “sender-address”)

2456 *General*

2457 *InvalidSuperior* – if “superior-identifier” is unknown

2458 *InvalidInferior* – if no ENROL has been received for this “inferior-identifier”, or if  
 2459 RESIGN has been received from this Inferior.

2460 *Note – A CONFIRMED message arriving before a CONFIRM message is sent, or after a*  
 2461 *CANCEL has been sent will occur when the Inferior has taken an autonomous*  
 2462 *decision and is not regarded as occurring in the wrong state. (The latter will cause a*  
 2463 *CONTRADICTION message to be sent.)*

2464 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm-received” =  
 2465 “false”; CONFIRMED/response refers to a CONFIRMED message with “confirm-received” =  
 2466 “true”.

### 2467 7.7.9 CANCEL

2468 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.

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

2469

2470       **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.

2471       **qualifiers** standardised or other qualifiers.

2472       **target-address** the address to which the CANCEL message is sent. This will be the  
2473       “inferior-address” from the ENROL message.

2474       **sender-address** the address from which the CANCEL is sent. This is an address of the  
2475       Superior.

2476       When received by an Inferior, the effects of any operations associated with the Inferior should be  
2477       undone. If the Inferior had sent PREPARED, the Inferior is released from its promise to be able to  
2478       Confirm the operations.

2479       Types of FAULT possible (sent to “sender-address”)

2480        *General*

2481        *InvalidInferior* – if “inferior-identifier” is unknown

2482        *WrongState* – if a CONFIRM has been received by this Inferior.

### 2483       7.7.10 CANCELLED

2484       Sent when the Inferior has applied (or is applying) cancellation of the operations associated with  
2485       the Inferior. CANCELLED is sent from Inferior to Superior in the following cases:

- 2486               1. before (and instead of) sending PREPARED, to indicate the Inferior is unable to  
2487               apply the operations in full and is cancelling all of them;
- 2488               2. in reply to CANCEL, regardless of whether PREPARED has been sent;
- 2489               3. after sending PREPARED and then making and applying an autonomous  
2490               decision to Cancel.
- 2491               4. in reply to CONFIRM\_ONE\_PHASE if the Inferior decides to Cancel the  
2492               associated operations

2493       As is specified in the state tables, cases 1, 2 and 3 are not always distinct in some circumstances  
2494       of recovery and resending of messages.

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

2495

2496 **superior-identifier** the “superior-identifier” as on the CONTEXT message.

2497 **inferior-identifier** the inferior identifier as on the earlier ENROL message.

2498 **qualifiers** standardised or other qualifiers.

2499 **target-address** the address to which the CANCELLED is sent. This will be the Superior  
2500 address as on the CONTEXT message.

2501 **sender-address** the address from which the CANCELLED is sent. This is an address of  
2502 the Inferior.

2503 Types of FAULT possible (sent to “sender-address”)

2504 **General**

2505 **InvalidSuperior** – if “superior-identifier” is unknown

2506 **InvalidInferior** – if no ENROL has been received for this ”inferior-identifier”, or if  
2507 RESIGN has been received from this Inferior

2508 **WrongState** – if CONFIRM has been sent

2509 *Note – A CANCELLED message arriving before a CANCEL message is sent, or after a*  
2510 *CONFIRM has been sent will occur when the Inferior has taken an autonomous*  
2511 *decision and is not regarded as occurring in the wrong state. (The latter will cause a*  
2512 *CONTRADICTION message to be sent.)*

### 2513 7.7.11 CONFIRM\_ONE\_PHASE

2514 Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In this  
2515 case the two-phase exchange is not performed between the Superior and Inferior and the outcome  
2516 decision for the operations associated with the Inferior is determined by the Inferior.

Parameter	Type
inferior-identifier	Identifier
report-hazard	boolean

Parameter	Type
qualifiers	List of qualifiers
target-address	BTP Address
sender-address	BTP Address

2517

2518 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this  
 2519 Inferior.

2520 **report hazard** Defines whether the superior wishes to be informed if a mixed condition  
 2521 occurs for the operations associated with the Inferior. If “report-hazard” is “true”, the  
 2522 Inferior will reply with HAZARD if a mixed condition occurs, or if the Inferior cannot  
 2523 determine that a mixed condition has not occurred. If “report-hazard” is false, the  
 2524 Inferior will report only its own decision, regardless of whether that decision was  
 2525 correctly and consistently applied. Default is false.

2526 **qualifiers** standardised or other qualifiers.

2527 **target-address** the address to which the CONFIRM\_ONE\_PHASE message is sent This  
 2528 will be the “inferior-address” on the ENROL message.

2529 **sender-address** the address from which the CONFIRM\_ONE\_PHASE is sent. This is an  
 2530 address of the Superior.

2531 CONFIRM\_ONE\_PHASE can be issued by a Superior to an Inferior from whom PREPARED  
 2532 has been received (subject to the requirement that there is only one enrolled Inferior).

2533 Types of FAULT possible (sent to “sender-address”)

2534 *General*

2535 *InvalidInferior* – if “inferior-identifier” is unknown

2536 *WrongState* – if a PREPARE has already been sent to this Inferior

### 2537 7.7.12 HAZARD

2538 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly and  
 2539 consistently Cancel or Confirm the operations in accord with the decision , or when the Inferior is  
 2540 unable to determine that a “mixed” condition has not occurred.

2541 HAZARD is also used to reply to a CONFIRM\_ONE\_PHASE if the Inferior determines there is a  
 2542 mixed condition within its associated operations or is unable to determine that there is not a  
 2543 mixed condition.

2544 *Note - If the Inferior makes its own autonomous decision then it signals that decision with*  
 2545 *CONFIRMED or CANCELLED and waits to receive a confirmatory CONFIRM or*

2546 *CANCEL, or a CONTRADICTION if the autonomous decision by the Inferior was*  
 2547 *the opposite of that made by the Superior.*  
 2548

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

2549

2550 **superior-identifier** The “superior-identifier” as on the ENROL message

2551 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message

2552 **level** indicates, with value “mixed” that a mixed condition has definitely occurred; or, with  
 2553 value “possible” that it is unable to determine whether a mixed condition has occurred  
 2554 or not.

2555 **qualifiers** standardised or other qualifiers.

2556 **target-address** the address to which the HAZARD is sent. This will be the superior  
 2557 address from the ENROL message.

2558 **sender-address** the address from which the HAZARD is sent. This is an address of the  
 2559 Inferior.

2560 Types of FAULT possible (sent to “sender-address”)

2561 ***General***

2562 ***InvalidSuperior*** – if “superior-identifier” is unknown

2563 ***InvalidInferior*** – if no ENROL has been received for this “inferior-identifier”, or if  
 2564 RESIGN has been received from this Inferior

2565 The form HAZARD/mixed refers to a HAZARD message with “level” = “mixed”, the form  
 2566 HAZARD/possible refers to a HAZARD message with “level” = “possible”.

2567 **7.7.13 CONTRADICTION**

2568 Sent by the Superior to an Inferior that has taken an autonomous decision contrary to the decision  
2569 for the Atom. This is detected by the Superior when the ‘wrong’ one of CONFIRMED or  
2570 CANCELLED is received. CONTRADICTION is also sent in response to a HAZARD message.

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

2571

2572 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this  
2573 Inferior.

2574 **qualifiers** standardised or other qualifiers.

2575 **target-address** the address to which the CONTRADICTION message is sent. This will be  
2576 the “inferior-address” from the ENROL message.

2577 **sender-address** the address from which the CONTRADICTION is sent. This is an  
2578 address of the Superior.

2579 Types of FAULT possible (sent to “sender-address”)

2580 **General**

2581 **InvalidInferior** – if “inferior-identifier” is unknown

2582 **WrongState** – if neither CONFIRMED or CANCELLED has been sent by this Inferior

2583 **7.7.14 SUPERIOR\_STATE**

2584 Sent by a Superior as a query to an Inferior when

2585 1. in the active state

2586 2. there is uncertainty what state the Inferior has reached (due to recovery from  
2587 previous failure or other reason).

2588 Also sent by the Superior to the Inferior in response to a received INFERIOR\_STATE, in  
2589 particular states.

Parameter	Type
inferior-identifier	Identifier

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

2590

2591 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this  
 2592 Inferior.

2593 **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

2594

2595 **response-requested** true, if SUPERIOR\_STATE is sent as a query at the Superior’s  
 2596 initiative; false, if SUPERIOR\_STATE is sent in reply to a received  
 2597 INFERIOR\_STATE or other message. Can only be true if status is active or prepared-  
 2598 received. Default is “false”

2599 **qualifiers** standardised or other qualifiers.

2600 **target-address** the address to which the SUPERIOR\_STATE message is sent. This will  
 2601 be the “inferior-address” from the ENROL message.

2602           **sender-address** the address from which the SUPERIOR\_STATE is sent. This is an  
2603           address of the Superior.

2604       The Inferior, on receiving SUPERIOR\_STATE with “response-requested = true, should reply in a  
2605       timely manner by (depending on its state) repeating the previous message it sent or by sending  
2606       INFERIOR\_STATE with the appropriate status value.

2607       A status of unknown shall only be sent if it has been determined for certain that the Superior has  
2608       no knowledge of the Inferior, or (equivalently) it can be determined that the relationship with the  
2609       Inferior was cancelled. If there could be persistent information corresponding to the Superior, but  
2610       it is not accessible from the entity receiving an INFERIOR\_STATE/\*y (or other) message  
2611       targeted to the Superior or that entity cannot determine whether any such persistent information  
2612       exists or not, the response shall be Inaccessible.

2613       SUPERIOR\_STATE/unknown is also used as a response to messages, other than  
2614       INFERIOR\_STATE/\*y that are received when the Inferior is not known (and it is known there is  
2615       no state information for it).

2616       The form SUPERIOR\_STATE/abcd refers to a SUPERIOR\_STATE message status having a  
2617       value equivalent to “abcd” (for active, prepared-received, unknown and inaccessible) and with  
2618       “response-requested” = “false”. SUPERIOR\_STATE/abcd/y refers to a similar message, but with  
2619       “response-requested” = “true”. The form SUPERIOR\_STATE/\*y refers to a  
2620       SUPERIOR\_STATE message with “response-requested” = “true” and any value for status.

#### 2621       7.7.15 INFERIOR\_STATE

2622       Sent by an Inferior as a query when in the active state to a Superior, when (due recovery from  
2623       previous failure or other reason) there is uncertainty what state the Superior has reached.

2624       Also sent by the Inferior to the Superior in response to a received SUPERIOR\_STATE, in  
2625       particular states.

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

2626

2627           **superior-identifier** The “superior-identifier” as used on the ENROL message

2628           **inferior-identifier** The “inferior-identifier” as on the ENROL message

2629 **status** states the current state of the Inferior, which corresponds to the last message sent to  
2630 the Superior by (or in the case of ENROL for) the Inferior

<b>status value</b>	<b>meaning/previous message sent</b>
<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

2631

2632 **response-requested** “true” if INFERIOR\_STATE is sent as a query at the Superior’s  
2633 initiative; “false” if INFERIOR\_STATE is sent in reply to a received  
2634 SUPERIOR\_STATE or other message. Can only be “true” if “status” is “active” or  
2635 “prepared-received”. Default is “false”

2636 **qualifiers** standardised or other qualifiers.

2637 **target-address** the address to which the INFERIOR\_STATE is sent. This will be the  
2638 “target-address” as used the original ENROL message.

2639 **sender-address** the address from which the INFERIOR\_STATE is sent. This is an  
2640 address of the Inferior.

2641 The Superior, on receiving INFERIOR\_STATE with “response-requested” = “true”, should reply  
2642 in a timely manner by (depending on its state) repeating the previous message it sent or by  
2643 sending SUPERIOR\_STATE with the appropriate status value.

2644 A status of “unknown” shall only be sent if it has been determined for certain that the Inferior has  
2645 no knowledge of a relationship with the Superior. If there could be persistent information  
2646 corresponding to the Superior, but it is not accessible from the entity receiving an  
2647 SUPERIOR\_STATE/\*y (or other) message targetted on the Inferior or the entity cannot  
2648 determine whether any such persistent information exists, the response shall be “inaccessible”.

2649 INFERIOR\_STATE/unknown is also used as a response to messages, other than  
2650 SUPERIOR\_STATE/\*y that are received when the Inferior is not known (and it is known there  
2651 is no state information for it).

2652 A SUPERIOR\_STATE/INFERIOR\_STATE exchange that determines that one or both sides are  
2653 in the active state does not require that the Inferior be cancelled (unlike some other two-phase  
2654 commit protocols). The relationship between Superior and Inferior, and related Application

2655 Elements may be continued, with new Application Messages carrying the same CONTEXT.  
2656 Similarly, if the Inferior is prepared but the Superior is active, there is no required impact on the  
2657 progression of the relationship between them.

2658 The form INFERIOR\_STATE/abcd refers to a INFERIOR\_STATE message status having a  
2659 value equivalent to “abcd” (for active, unknown and inaccessible) and with “response-requested”  
2660 = “false”. INFERIOR\_STATE/abcd/y refers to a similar message, but with “response-requested”  
2661 = “true”. The form INFERIOR\_STATE/\*y refers to a INFERIOR\_STATE message with  
2662 “response-requested” = “true” and any value for status.

#### 2663 7.7.16 REDIRECT

2664 Sent when the address previously given for a Superior or Inferior is no longer valid and the  
2665 relevant state information is now accessible with a different address (but the same superior or  
2666 “inferior-identifier”).

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

2667

2668 **superior-identifier** The “superior-identifier” as on the CONTEXT message and used on an  
2669 ENROL message. (present only if the REDIRECT is sent from the Inferior).

2670 **inferior-identifier** The “inferior-identifier” as on the ENROL message

2671 **old-address** The previous address of the sender of REDIRECT. A match is considered to  
2672 apply if any of the “old-address” values match one that is already known.

2673 **new-address** The (set of alternatives) “new-address” values to be used for messages sent  
2674 to this entity.

2675 **qualifiers** standardised or other qualifiers.

2676 **target-address** the address to which the REDIRECT is sent. This is the address of the  
2677 opposite side (superior/inferior) as given in a CONTEXT or ENROL message

2678 If the Actor whose address is changed is an Inferior, the “new-address” value replaces the  
2679 “inferior-address” as present in the ENROL.

2680 If the Actor whose address is changed is a Superior, the “new-address” value replaces the  
2681 Superior address as present in the CONTEXT message (or as present in any other mechanism  
2682 used to establish the Superior:Inferior relationship).

## 2683 7.8 Messages used in Control Relationships

### 2684 7.8.1 BEGIN

2685 A request to a Factory to create a new Business Transaction. This may either be a new top-level  
2686 transaction, in which case the Composer or Coordinator will be the Decider, or the new Business  
2687 Transaction may be immediately made the Inferior within an existing Business Transaction (thus  
2688 creating a sub-Composer or sub-Coordinator).

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

2689

2690 **transaction-type** identifies whether a new Cohesion or new Atom is to be created; this  
2691 value will be the “superior-type” in the new CONTEXT

2692 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction timelimit”  
2693 may be present on BEGIN, to set the timelimit for the new Business Transaction and  
2694 will be copied to the new CONTEXT. The standard qualifier “Inferior name” may be  
2695 present if there is a CONTEXT related to the BEGIN.

2696 **target-address** the address of the entity to which the BEGIN is sent. How this address is  
2697 acquired and the nature of the entity are outside the scope of this specification.

2698 **reply-address** the address to which the replying BEGUN and related CONTEXT message  
2699 should be sent.

2700 A new top-level Business Transaction is created if there is no CONTEXT related to the BEGIN.  
2701 A Business Transaction that is to be Inferior in an existing Business Transaction is created if the  
2702 CONTEXT message for the existing Business Transaction is related to the BEGIN. In this case,  
2703 the Factory is responsible for enrolling the new Composer or Coordinator as an Inferior of the  
2704 Superior identified in that CONTEXT.

2705 *Note – This specification does not provide a standardised means to determine which of*  
2706 *the Inferiors of a sub-Composer are in its Confirm set. This is considered part of the*  
2707 *application:inferior relationship.*

2708 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction-type” having the  
2709 corresponding value.

2710 Types of FAULT possible (sent to “reply-address”)

2711

**General**

2712

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

2713

**WrongState** - only issued if there is a related CONTEXT, and the Superior identified by

2714

the CONTEXT is in the wrong state to enrol new Inferiors

2715

**7.8.2 BEGUN**

2716

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

2717

the new Business Transaction.

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

2718

2719

**decider-address** for a top-most transaction (no CONTEXT related to the BEGIN), this is

2720

the address to which PREPARE\_INFERIORS, CONFIRM\_TRANSACTION,

2721

CANCEL\_TRANSACTION, CANCEL\_INFERIORS and

2722

REQUEST\_INFERIOR\_STATUSES messages are to be sent; if a CONTEXT was

2723

related to the BEGIN this parameter is absent

2724

**inferior-address** for a non-top-most transaction (a CONTEXT was related to the BEGIN),

2725

this is the “inferior-address” used in the enrolment with the Superior identified by the

2726

CONTEXT related to the BEGIN. The parameter is optional (implementor’s choice) if

2727

this is not a top-most transaction; it shall be absent if this is a top-most transaction.

2728

**transaction-identifier** if this is a top-most transaction, this is an globally-unambiguous

2729

identifier for the new Decider (Composer or Coordinator). If this is not a top-most

2730

transaction, the transaction-identifier shall be the inferior-identifier used in the

2731

enrolment with the Superior identified by the CONTEXT related to the BEGIN.

2732

*Note – The “transaction-identifier” may be identical to the “superior-identifier” in*

2733

*the CONTEXT that is related to the BEGUN*

2734

**qualifiers** standardised or other qualifiers.

2735

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

2736

from the BEGIN.

2737

At implementation option, the “decider-address” and/or “inferior-address” and the “superior-

2738

address” in the related CONTEXT may be the same or may be different. There is no general

2739

requirement that they even use the same bindings. Any may also be the same as the “target-

2740 address” of the BEGIN message (the identifier on messages will ensure they are applied to the  
2741 appropriate Composer or Coordinator).

2742 No FAULT messages are issued on receiving BEGUN.

### 2743 7.8.3 PREPARE\_INFERIORS

2744 Sent from a Terminator to a Decider, but only if it is a Cohesion Composer, to tell it to prepare all  
2745 or some of its inferiors, by sending PREPARE to any that have not already sent PREPARED,  
2746 RESIGN or CANCELLED to the Decider (Composer) on its relationships as Superior. If the  
2747 inferiors-list parameter is absent, the request applies to all the inferiors; if the parameter is  
2748 present, it applies only to the identified inferiors of the Decider (Composer).

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

2749

2750 **transaction identifier** identifies the Decider and will be the transaction-identifier from the  
2751 BEGUN message.

2752 **inferiors-list** defines which of the Inferiors of this Decider preparation is requested for,  
2753 using the “inferior-identifiers” as on the ENROL received by the Decider (in its Role  
2754 as Superior). If this parameter is absent, the PREPARE applies to all Inferiors.

2755 **qualifiers** standardised or other qualifiers.

2756 **target-address** the address to which the PREPARE\_INFERIORS message is sent. This  
2757 will be the decider-address from the BEGUN message.

2758 **reply-address** the address of the Terminator sending the PREPARE\_INFERIORS  
2759 message.

2760 For all Inferiors identified in the inferiors-list parameter (all Inferiors if the parameter is absent),  
2761 from which none of PREPARED, CANCELLED or RESIGNED has been received, the Decider  
2762 shall issue PREPARE. It will reply to the Terminator, using the “reply-address” on the  
2763 PREPARE\_INFERIORS message, sending an INFERIOR\_STATUSES message giving the status  
2764 of the Inferiors identified on the inferiors-list parameter (all of them if the parameter was absent).

2765 If one or more of the “inferior-identifier”s in the "inferior-list" is unknown (does not correspond  
2766 to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. It is an implementation  
2767 option whether CANCEL is sent to any of the Inferiors that are validly identified in the "inferiors-  
2768 list".

2769 Types of FAULT possible (sent to Superior address)

2770 **General**

2771 **InvalidDecider** – if Decider address is unknown

2772 **Redirect** – if the Decider now has a different “decider-address”

2773 **UnknownTransaction** – if the transaction-identifier is unknown

2774 **InvalidInferior** – if one or more inferior-identifiers on the inferiors-list is unknown

2775 **WrongState** – if a CONFIRM\_TRANSACTION or CANCEL\_TRANSACTION has  
2776 already been received by this Composer.

2777 The form PREPARE\_INFERIORS/all refers to a PREPARE\_INFERIORS message where the  
2778 “inferiors-list” parameter is absent. The form PREPARE\_INFERIORS/specific refers to a  
2779 PREPARE\_INFERIORS message where the “inferiors-list” parameter is present.

2780 **7.8.4 CONFIRM\_TRANSACTION**

2781 Sent from a Terminator to a Decider to request confirmation of the Business Transaction. If the  
2782 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

2783

2784 **transaction-identifier** identifies the Decider. This will be the transaction-identifier from  
2785 the BEGUN message.

2786 **inferiors-list** defines which Inferiors enrolled with the Decider, if it is a Cohesion  
2787 Composer, are to be confirmed, using the “inferior-identifiers” as on the ENROL  
2788 received by the Decider (in its Role as Superior). Shall be absent if the Decider is an  
2789 Atom Coordinator.

2790 **report-hazard** Defines whether the Terminator wishes to be informed of hazard events and  
2791 contradictory decisions within the Business Transaction. If “report-hazard” is “true”,  
2792 the receiver will wait until responses (CONFIRMED, CANCELLED or HAZARD)  
2793 have been received from all of its inferiors, ensuring that any hazard events are  
2794 reported. If “report-hazard” is “false”, the Decider will reply with

2795           TRANSACTION\_CONFIRMED or TRANSACTION\_CANCELLED as soon as the  
2796           decision for the transaction is known.

2797           **qualifiers** standardised or other qualifiers.

2798           **target-address** the address to which the CONFIRM\_TRANSACTION message is sent.  
2799           This will be the “decider-address” on the BEGUN message.

2800           **reply-address** the address of the Terminator sending the CONFIRM\_TRANSACTION  
2801           message.

2802           If the “inferiors-list” parameter is present, the Inferiors identified shall be the “Confirm-set” of  
2803           the Cohesion. If the parameter is absent and the Business Transaction is a Cohesion, the  
2804           “Confirm-set” shall be all remaining Inferiors. If the Business Transaction is an Atom, the  
2805           “Confirm-set” is automatically all the Inferiors.

2806           Any Inferiors from which RESIGN is received are not counted in the Confirm-set.

2807           If, for each of the Inferiors in the Confirm-set, PREPARE has not been sent and PREPARED has  
2808           not been received, PREPARE shall be issued to that Inferior.

2809           *NOTE -- If PREPARE has been sent but PREPARED not yet received from an Inferior in*  
2810           *the Confirm-set, it is an implementation option whether and when to re-send*  
2811           *PREPARE. The Superior implementation may choose to re-send PREPARE if there*  
2812           *are indications that the earlier PREPARE was not delivered.*

2813           A Confirm decision may be made only if PREPARED has been received from all Inferiors in the  
2814           “Confirm-set”. The making of the decision shall be persistent (and if it is not possible to persist  
2815           the decision, it is not made). If there is only one remaining Inferior in the “Confirm set” and  
2816           PREPARE has not been sent to it, CONFIRM\_ONE\_PHASE may be sent to it.

2817           All remaining Inferiors that are not in the Confirm set shall be cancelled.

2818           If a Confirm decision is made and “report-hazard” was “false”, a  
2819           TRANSACTION\_CONFIRMED message shall be sent to the “reply-address”.

2820           If a Cancel decision is made and “report-hazard” was “false”, a TRANSACTION\_CANCELLED  
2821           message shall be sent to the “reply-address”.

2822           If "report-hazard" was "true", TRANSACTION\_CONFIRMED shall be sent to the "reply-  
2823           address" after CONFIRMED has been received from each Inferior in the Confirm-set and  
2824           CANCELLED or RESIGN from each and any Inferior not in the Confirm-set.

2825           If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.  
2826           CANCELLED from an Inferior in the Confirm-set or CONFIRMED from an Inferior not in the  
2827           Confirm-set), an INFERIOR\_STATUSES reporting the status for all Inferiors shall be sent to the  
2828           “reply-address”.

2829           If one or more of the "inferior-identifier"s in the "inferior-list" is unknown (does not correspond  
2830           to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. The Decider shall not make a  
2831           Confirm decision and shall not send CONFIRM to any Inferior.

2832 Types of FAULT possible (sent to “reply-address”)

2833        **General**

2834        **InvalidDecider** – if Decider address is unknown

2835        **Redirect** – if the Decider now has a different “decider-address”

2836        **UnknownTransaction** – if the transaction-identifier is unknown

2837        **InvalidInferior** – if one or more “inferior -identifiers” in the inferiors-list is unknown

2838        **WrongState** – if a CANCEL\_TRANSACTION has already been received .

2839 The form CONFIRM\_TRANSACTION/all refers to a CONFIRM\_TRANSACTION message  
 2840 where the “inferiors-list” parameter is absent. The form CONFIRM\_TRANSACTION/specific  
 2841 refers to a CONFIRM\_TRANSACTION message where the “inferiors-list” parameter is present.

2842 **7.8.5 TRANSACTION\_CONFIRMED**

2843 A Decider sends TRANSACTION\_CONFIRMED to a Terminator in reply to  
 2844 CONFIRM\_TRANSACTION if all of the Confirm-set confirms (and, for a Cohesion, all other  
 2845 Inferiors Cancel) without reporting hazards, or if the Decider made a Confirm decision and the  
 2846 CONFIRM\_TRANSACTION had a “report-hazards” value of “false”.

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

2847

2848        **transaction-identifier** the “transaction-identifier” as on the BEGUN message (i.e. the  
 2849        identifier of the Decider as a whole).

2850        **qualifiers** standardised or other qualifiers.

2851        **target-address** the address to which the TRANSACTION\_CONFIRMED is sent., this  
 2852        will be the “reply-address” from the CONFIRM\_TRANSACTION message

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

2854        **General**

2855        **InvalidTerminator** – if Terminator address is unknown

2856        **UnknownTransaction** – if the transaction-identifier is unknown

2857 **7.8.6 CANCEL\_TRANSACTION**

2858 Sent by a Terminator to a Decider at any time before CONFIRM\_TRANSACTION has been sent.

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

2859

2860 **transaction-identifier** identifies the Decider and will be the transaction-identifier from the  
2861 BEGUN message.

2862 **report-hazard** Defines whether the Terminator wishes to be informed of hazard events and  
2863 contradictory decisions within the Business Transaction. If "report-hazard" is "true",  
2864 the receiver will wait until responses (CONFIRMED, CANCELLED or HAZARD)  
2865 have been received from all of its inferiors, ensuring that any hazard events are  
2866 reported. If "report-hazard" is "false", the Decider will reply with  
2867 TRANSACTION\_CANCELLED immediately.

2868 **qualifiers** standardised or other qualifiers.

2869 **target-address** the address to which the CANCEL\_TRANSACTION message is sent.  
2870 This will be the decider-address from the BEGUN message.

2871 **reply-address** the address of the Terminator sending the CANCEL\_TRANSACTION  
2872 message.

2873 The Business Transaction is cancelled – this is propagated to any remaining Inferiors by issuing  
2874 CANCEL to them. No more Inferiors will be permitted to enrol.

2875 If "report-hazard" was "false", a TRANSACTION\_CANCELLED message shall be sent to the  
2876 "reply-address".

2877 If "report-hazard" was "true" and any HAZARD or CONFIRMED message was received, an  
2878 INFERIOR\_STATUSES reporting the status for all Inferiors shall be sent to the "reply-address".

2879 If "report-hazard" was "true", TRANSACTION\_CANCELLED shall be sent to the "reply-  
2880 address" after CANCELLED or RESIGN has been received from each Inferior.

2881 Types of FAULT possible (sent to Superior address)

2882 **General**

- 2883 *InvalidDecider* – if Decider address is unknown
- 2884 *Redirect* – if the Decider now has a different “decider-address”
- 2885 *UnknownTransaction* – if the transaction-identifier is unknown
- 2886 *WrongState* – if a CONFIRM\_TRANSACTION has been received by this Composer.

2887 **7.8.7 CANCEL\_INFERIORS**

2888 Sent by a Terminator to a Decider, but only if is a Cohesion Composer, at any time before  
 2889 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

- 2890
- 2891 **transaction-identifier** identifies the Decider and will be the transaction-identifier from the  
 2892 BEGUN message.
- 2893 **inferiors-list** defines which of the Inferiors of this Decider are to be cancelled, using the  
 2894 “inferior-identifiers” as on the ENROL received by the Decider (in its Role as  
 2895 Superior).
- 2896 **qualifiers** standardised or other qualifiers.
- 2897 **target-address** the address to which the CANCEL\_TRANSACTION message is sent.  
 2898 This will be the decider-address from the BEGUN message.
- 2899 **reply-address** the address of the Terminator sending the CANCEL\_TRANSACTION  
 2900 message.

2901 Only the Inferiors identified in the inferiors-list are to be cancelled. Any other inferiors are  
 2902 unaffected by a CANCEL\_INFERIORS. Further Inferiors may be enrolled.

2903 *Note – A CANCEL\_INFERIORS for all of the currently enrolled Inferiors will leave the*  
 2904 *Cohesion ‘empty’, but permitted to continue with new Inferiors, if any enrol.*

2905 If one or more of the "inferior-identifier"s in the "inferior-list" is unknown (does not correspond  
 2906 to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. It is an implementation  
 2907 option whether CANCEL is sent to any of the Inferiors that are validly identified in the "inferiors-  
 2908 list".

- 2909 Types of FAULT possible (sent to Superior address)
- 2910 **General**
- 2911 **InvalidDecider** – if Decider address is unknown
- 2912 **Redirect** – if the Decider now has a different “decider-address”
- 2913 **UnknownTransaction** – if the transaction-identifier is unknown
- 2914 **InvalidInferior** – if one or more inferior-identifiers on the inferiors-list is unknown
- 2915 **WrongState** – if a CONFIRM\_TRANSACTION or CANCEL\_TRANSACTION has been  
2916 received by this Composer.

### 2917 7.8.8 TRANSACTION\_CANCELLED

2918 A Decider sends TRANSACTION\_CANCELLED to a Terminator in reply to  
2919 CANCEL\_TRANSACTION or in reply to CONFIRM\_TRANSACTION if the Decider decided  
2920 to Cancel. In both cases, TRANSACTION\_CANCELLED is used only if all Inferiors cancelled  
2921 without reporting hazards or the CANCEL\_TRANSACTION or CONFIRM\_TRANSACTION  
2922 had a “report-hazard” value of “false.

#### Parameter

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

- 2923
- 2924 **transaction-identifier** the “transaction-identifier” as on the BEGUN message (i.e. the  
2925 identifier of the Decider as a whole).
- 2926 **qualifiers** standardised or other qualifiers.
- 2927 **target-address** the address to which the TRANSACTION\_CANCELLED is sent. This  
2928 will be the “reply-address” from the CANCEL\_TRANSACTION or  
2929 CONFIRM\_TRANSACTION message.

- 2930 Types of FAULT possible (sent to “decider-address”)
- 2931 **General**
- 2932 **InvalidTerminator** – if Terminator address is unknown
- 2933 **UnknownTransaction** – if the transaction-identifier is unknown

2934 **7.8.9 REQUEST\_INFERIOR\_STATUSES**

2935 Sent to a Decider to ask it to report the status of its Inferiors with an INFERIOR\_STATUSES  
2936 message. It can also be sent to any Actor with a “superior-address” or “inferior-address”, asking it  
2937 about the status of that Transaction Tree Nodes Inferiors, if there are any. In this latter case, the  
2938 receiver may reject the request with a FAULT(StatusRefused). If it is prepared to reply, but has  
2939 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

2940

2941 **target-identifier** identifies the transaction (or Transaction Tree Node). When the message  
2942 is used to a Decider, this will be the transaction-identifier from the BEGUN message.  
2943 Otherwise it will be the superior-identifier from a CONTEXT or an inferior-identifier  
2944 from an ENROL message.

2945 **inferiors-list** defines which inferiors enrolled with the target are to be included in the  
2946 INFERIOR\_STATUSES, using the “inferior-identifiers” as on the ENROL received  
2947 by the Decider (in its Role as Superior). If the list is absent, the status of all enrolled  
2948 Inferiors will be reported.

2949 **qualifiers** standardised or other qualifiers.

2950 **target-address** the address to which the REQUEST\_STATUS message is sent. When  
2951 used to a Decider, this will be the “decider-address” from the BEGUN message.  
2952 Otherwise it may be a “superior-address” from a CONTEXT or “inferior-address”  
2953 from an ENROL message.

2954 **reply-address** the address to which the replying INFERIOR\_STATUSES is to be sent

2955 Types of FAULT possible (sent to reply-address)

2956 **General**

2957 **Redirect** – if the intended target now has a different address

2958 **StatusRefused** – if the receiver is not prepared to report its status to the sender of this  
2959 message. This “fault-type” shall not be issued when a Decider receives  
2960 REQUEST\_STATUSES from the Terminator.

2961 **UnknownTransaction** – if the transaction-identifier is unknown

2962 The form REQUEST\_INFERIOR\_STATUSES/all refers to a REQUEST\_STATUS with the  
 2963 inferiors-list absent. The form REQUEST\_INFERIOR\_STATUS/specific refers to a  
 2964 REQUEST\_INFERIOR\_STATUS with the inferiors-list present.

2965 **7.8.10 INFERIOR\_STATUSES**

2966 Sent by a Decider to report the status of all or some of its inferiors in response to a  
 2967 REQUEST\_INFERIOR\_STATUSES, PREPARE\_INFERIORS, CANCEL\_INFERIORS,  
 2968 CANCEL\_TRANSACTION with “report-hazard” value of “true” and  
 2969 CONFIRM\_TRANSACTION with “report-hazard” value of “true”. It is also used by any Actor in  
 2970 response to a received REQUEST\_INFERIOR\_STATUSES to report the status of inferiors, if  
 2971 there are any.

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

2972

2973 **responders-identifier** the target-identifier used on the  
 2974 REQUEST\_INFERIOR\_STATUSES.

2975 **status-list** contains a number of Status-items, each reporting the status of one of the  
 2976 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).

2977

2978 The status value reports the current status of the particular inferior, as known to the Decider  
 2979 (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

status value	Meaning
<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)

2980

2981 **general-qualifiers** standardised or other qualifiers applying to the  
 2982 INFERIOR\_STATUSES as a whole. Each Status-item contains a “qualifiers” field  
 2983 containing qualifiers applying to (and received from) the particular Inferior.

2984 **target-address** the address to which the INFERIOR\_STATUSES is sent. This will be the  
 2985 “reply-address” on the received message

2986 If the inferiors-list parameter was present on the received message, only the inferiors identified by  
 2987 that parameter shall have their status reported in status-list of this message. If the inferiors-list  
 2988 parameter was absent, the status of all enrolled inferiors shall be reported, except that an inferior  
 2989 that had been reported as *cancelled* or *resigned* on a previous INFERIOR\_STATUSES message  
 2990 **may** be omitted (sender’s option).

2991 Types of FAULT possible (sent to “decider-address”)

2992

## *General*

2993

*InvalidTerminator* – if Terminator address is unknown

2994

*UnknownTransaction* – if the transaction-identifier is unknown

2995

## **7.9 Groups – combinations of related messages**

2996

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

2997

2998

2999

3000

3001

3002

### **7.9.1 CONTEXT & Application Message**

3003

3004

3005

3006

3007

3008

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

3009

3010

3011

3012

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

3013

3014

3015

3016

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

3017

3018

3019

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

3020

3021

3022

If the CONTEXT is a CONTEXT/atom, the Actor receiving the CONTEXT shall ensure that a CONTEXT\_REPLY message is sent back to the “reply-address” of the CONTEXT with the appropriate completion status.

3023

3024

3025

3026

3027

*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.*

3028    **7.9.2   CONTEXT\_REPLY & ENROL**

3029        **Meaning:** the enrolment of the Inferior identified in the ENROL is to be performed with  
3030        the Superior identified in the CONTEXT message this CONTEXT\_REPLY is replying  
3031        to. If the “completion-status” of CONTEXT\_REPLY is “related”, failure of this  
3032        enrolment shall prevent the confirmation of the Business Transaction.

3033        **target-address:** the “target-address” is that of the CONTEXT\_REPLY. This will be the  
3034        “reply-address” of the CONTEXT message (in many cases, including request/reply  
3035        application exchanges, this address will usually be implicit).

3036        The “target-address” of the ENROL message is omitted.

3037        The Actor receiving the related group will use the retained Superior address from the  
3038        CONTEXT sent earlier to forward the ENROL. When doing so, it changes the ENROL to  
3039        ask for a response (if it was an ENROL/no-rsp-req) and supplies its own address as the  
3040        “reply-address”, remembering the original “reply-address” if there was one.

3041        If ENROLLED is received and the original received ENROL was ENROL/rsp-req, the  
3042        ENROLLED is forwarded back to the original “reply-address”.

3043        If this attempt fails (i.e. ENROLLED is not received), and the “completion-status” of the  
3044        CONTEXT\_REPLY was “related”, the Actor is required to ensure that the Superior does  
3045        not proceed to confirmation. How this is achieved is an implementation option, but must  
3046        take account of the possibility that direct communication with the Superior may fail. (One  
3047        method is to prevent CONFIRM\_TRANSACTION being sent to the Superior (in its Role  
3048        as Decider); another is to enrol as another Inferior before sending the original CONTEXT  
3049        out with an Application Message). If the Superior is a sub-coordinator or sub-composer,  
3050        an enrolment failure must ensure the sub-coordinator does not send PREPARED to its  
3051        own Superior.

3052        If the Actor receiving the related group is also the Superior (i.e. it has the same binding  
3053        address), the explicit forwarding of the ENROL is not required, but the resultant effect –  
3054        that if enrolment fails the Superior does not Confirm or issue PREPARED – shall be the  
3055        same.

3056        A CONTEXT\_REPLY & ENROL group may contain multiple ENROL messages, for  
3057        several Inferiors. Each ENROL shall be forwarded and an ENROLLED reply received  
3058        before the Superior is allowed to Confirm if the “completion-status” in the  
3059        CONTEXT\_REPLY was “related”.

3060        When the group is constructed, if the CONTEXT had “superior-type” value of “atom”,  
3061        the “completion-status” of the CONTEXT\_REPLY shall be “related”. If the “superior-  
3062        type” was “cohesive”, the “completion-status” shall be “incomplete” or “related” (as  
3063        required by the application). If the value is “incomplete”, the Actor receiving the group  
3064        shall forward the ENROLs, but is not required to prevent confirmation (though it may do  
3065        so).

3066 **7.9.3 CONTEXT\_REPLY (& ENROL) & PREPARED / & CANCELLED**

3067 This combination is characterised by a related CONTEXT\_REPLY and either or both of  
3068 PREPARED and CANCELLED, with or without ENROL.

3069 **Meaning:** If ENROL is present, the meaning and required processing is the same as for  
3070 CONTEXT\_REPLY & ENROL. The PREPARED or CANCELLED message(s) are  
3071 forwarded to the Superior identified in the CONTEXT message this CONTEXT\_REPLY  
3072 is replying to.

3073 *Note – the combination of CONTEXT\_REPLY & ENROL & CANCELLED may be used*  
3074 *to force cancellation of an atom*

3075 **target-address:** the “target-address” is that of the CONTEXT\_REPLY. This will be the  
3076 “reply-address” of the CONTEXT message (in many cases, including request/reply  
3077 application exchanges, this address will usually be implicit).

3078 The “target-address” of the PREPARED and CANCELLED message is omitted – they  
3079 will be sent to the Superior identified in the earlier CONTEXT message.

3080 The Actor receiving the group forwards the PREPARED or CANCELLED message to the  
3081 Superior in as for an ENROL, using the retained Superior address from the CONTEXT  
3082 sent earlier, except there is no reply required from the Superior.

3083 If (as is usual) an ENROL and PREPARED or CANCELLED message are for the same  
3084 Inferior, the ENROL shall be sent first, but the Actor need not wait for the ENROLLED  
3085 to come back before sending the PREPARED or CANCELLED (so an  
3086 ENROL+PREPARED bundle from this Actor to the Superior could be used).

3087 The group can contain multiple ENROL, PREPARED and CANCELLED messages.  
3088 Each PREPARED and CANCELLED message will be for a different Inferior.. There is  
3089 no constraint on the order of their forwarding, except that ENROL and PREPARED or  
3090 CANCELLED for the same Inferior shall be delivered to the Superior in the order  
3091 ENROL first, followed by the other message for that Inferior.

3092 **7.9.4 CONTEXT\_REPLY & ENROL & Application Message (& PREPARED)**

3093 This combination is characterised by a related CONTEXT\_REPLY, ENROL and an Application  
3094 Message. PREPARED may or may not be present in the related group.

3095 **Meaning:** the relation between the BTP messages is as for the preceding groups, The  
3096 transmission of the Application Message (and application effects implied by its  
3097 transmission) has been associated with the Inferior identified by the ENROL and will be  
3098 subject to the outcome delivered to that Inferior.

3099 **target-address:** the “target-address” of the group is the “target-address” of the  
3100 CONTEXT\_REPLY which shall also be the “target-address” of the Application Message.  
3101 The ENROL and PREPARED messages do not contain their “target-address” parameters.

3102 The processing of ENROL and PREPARED messages is the same as for the previous  
3103 groups.

3104 This group can be used when participation in Business Transaction (normally a  
3105 cohesion), is initiated by the service (Inferior) side, which fetches or acquires the  
3106 CONTEXT, with some associated application semantic, performs some work for the  
3107 transaction and sends an Application Message with a related ENROL. The  
3108 CONTEXT\_REPLY allows the addressing of the application (and the  
3109 CONTEXT\_REPLY) to be distinct from that of the Superior.

3110 The Actor receiving the group may associate the “inferior-identifier” received on the  
3111 ENROL with the Application Message in a manner that is visible to the application  
3112 receiving the message (e.g. for subsequent use in Terminator:Decider exchanges).

### 3113 7.9.5 BEGUN & CONTEXT

3114 **Meaning:** the CONTEXT is that for the new Business Transaction, containing the  
3115 Superior address.

3116 **target-address:** the “target-address” is that of the BEGUN message – this will be the  
3117 “reply-address” of the earlier BEGIN message.

### 3118 7.9.6 BEGIN & CONTEXT

3119 **Meaning:** the new Business Transaction is to be an Inferior (sub-coordinator or sub-  
3120 composer) of the Superior identified by the CONTEXT. The Factory (receiver of the  
3121 BEGIN) will perform the enrolment.

3122 **target-address:** the “target-address” is that of the BEGIN – this will be the address of the  
3123 Factory.

## 3124 7.10 Standard qualifiers

3125 The following qualifiers are expected to be of general use to many applications and environments.  
3126 The URI “urn:oasis:names:tc:BTP:1.0:qualifiers” is used in the Qualifier group  
3127 value for the qualifiers defined here.

### 3128 7.10.1 Transaction timelimit

3129 The transaction timelimit allows the Superior (or an Application Element initiating the Business  
3130 Transaction) to indicate the expected length of the active phase, and thus give an indication to the  
3131 Inferior of when it would be appropriate to initiate cancellation if the active phase appears to  
3132 continue too long. The time limit ends (the clock stops) when the Inferior decides to be prepared  
3133 and issues PREPARED to the Superior.

3134 It should be noted that the expiry of the time limit does not change the permissible actions of the  
3135 Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is **permitted** to  
3136 initiate cancellation for internal reasons. The timelimit gives an indication to the entity of when it  
3137 will be useful to exercise this right.

3138 The qualifier is propagated on a CONTEXT message.

3139 The “Qualifier name” shall be “transaction-timelimit”.

3140 The “Content” shall contain the following field:

Content field	Type
Timelimit	Integer

3141

3142 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the  
3143 time of transmission of the containing CONTEXT, of the active phase of the Business  
3144 Transaction.

### 3145 7.10.2 Inferior timeout

3146 This qualifier allows an Inferior to limit the duration of its “promise”, when sending PREPARED,  
3147 that it will maintain the ability to Confirm or Cancel the effects of all associated operations.  
3148 Without this qualifier, an Inferior is expected to retain the ability to Confirm or Cancel  
3149 indefinitely. If the timeout does expire, the Inferior is released from its promise and can apply the  
3150 decision indicated in the qualifier.

3151 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply a  
3152 Confirm or Cancel decision before the CONFIRM or CANCEL is received and before this  
3153 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision, and  
3154 (as with other transaction mechanisms), is considered to be an exceptional event. As with  
3155 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the expiry  
3156 of this timeout, is liable to cause contradictory decisions across the Business Transaction. BTP  
3157 ensures that at least the occurrence of such a contradiction will be (eventually) reported to the  
3158 Superior of the Business Transaction. BTP treats “true” heuristic decisions and autonomous  
3159 decisions after timeout the same way – in fact, the expiry in this timeout does not cause a  
3160 qualitative (state table) change in what can happen, but rather a step change in the probability that  
3161 it will.

3162 The expiry of the timeout does not strictly require that the Inferior immediately invokes the  
3163 intended decision, only that is at liberty to do so. An implementation may choose to only apply  
3164 the decision if there is contention for the underlying resource, for example. Nevertheless,  
3165 Superiors are recommended to avoid relying on this and ensure decisions for the Business  
3166 Transaction are made before these timeouts expire (and allow a margin of error for network  
3167 latency etc.).

3168 The qualifier may be present on a PREPARED message. If the PREPARED message has the  
3169 “default-is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall have  
3170 the value “cancel”.

3171 The “Qualifier name” shall be “inferior-timeout”.

3172 The “Content” shall contain the following fields:

Content field	Type
Timeout	Integer
IntendedDecision	"confirm" or "cancel"

3173

3174 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the  
 3175 carrying message, the Inferior intends to maintain its ability to either Confirm or Cancel the  
 3176 effects of the associated operations, as ordered by the receiving Superior.

3177 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an  
 3178 autonomous decision is made.

### 3179 7.10.3 Minimum inferior timeout

3180 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the  
 3181 Inferior. If a Superior knows that the decision for the Business Transaction will not be determined  
 3182 for some period, it can require that Inferiors do not send PREPARED messages with Inferior  
 3183 timeouts that would expire before then. An Inferior that is unable or unwilling to send a  
 3184 PREPARED message with a longer (or no) timeout **should** Cancel, and reply with  
 3185 CANCELLED.

3186 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If present on  
 3187 more than one, and with different values of the MinimumTimeout field, the value on  
 3188 ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall prevail over  
 3189 either of the others.

3190 The "Qualifier name" shall be "minimum-inferior-timeout".

3191 The "Content" shall contain the following field:

Content field	Type
MinimumTimeout	Integer

3192

3193 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be  
 3194 acceptable in the Inferior timeout qualifier on an answering PREPARED message.

### 3195 7.10.4 Inferior name

3196 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on  
 3197 INFERIOR\_STATUSES and thus allow the Terminator to determine which Inferior (of the  
 3198 Composer or Coordinator) is related to which application work. This is in addition to the  
 3199 "inferior-identifier" field. The name can be human-readable and can also be used in fault tracing,  
 3200 debugging and auditing.

3201 The name is never used by the BTP Actors themselves to identify each other or to direct  
3202 messages. (The BTP Actors use the addresses and the identifiers in the message parameters for  
3203 those purposes.)

3204 This specification makes no requirement that the names are unambiguous within any scope  
3205 (unlike the globally unambiguous “inferior-identifier” on ENROLLED and BEGUN). Other  
3206 specifications, including those defining use of BTP with a particular application may place  
3207 requirements on the use and form of the names. (This may include reference to information  
3208 passed in Application Messages or in other, non-standardised, qualifiers.)

3209 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item in  
3210 INFERIOR\_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if present,  
3211 the same qualifier value **should** be included in the consequent ENROL. If  
3212 INFERIOR\_STATUSES includes a Status-item for an Inferior whose ENROL had an inferior-  
3213 name qualifier, the same qualifier value **should** be included in the Status-item.

3214 The “Qualifier -name” shall be “inferior-name”

3215 The “Content” shall contain the following fields:

Content field	Type
inferior-name	String

3216

3217 **Inferior name** the name assigned to the enrolling Inferior.

## 3218 **8 State Tables**

3219 The state tables deal with the state transitions of the Superior and Inferior roles and which  
3220 message can be sent and received in each state. The state tables directly cover only a single, bi-  
3221 lateral Superior:Inferior relationship. The interactions between, for example, multiple Inferiors of  
3222 a single Superior that will apply the same decision to all or some (of them), are dealt with in the  
3223 definitions of the “decision” events which also specify when changes are made to persistent state  
3224 information (see below).

3225 There are two state tables, one for Superior, one for Inferior. States are identified by a letter-digit  
3226 pair, with upper-case letters for the superior, lower-case for the inferior. The same letter is used to  
3227 group states which have the same, or similar, persistent state, with the digit indicating volatile  
3228 state changes or minor variations. Corresponding upper and lower-case letters are used to identify  
3229 (approximately) corresponding Superior and Inferior states.

3230 The Inferior table includes events occurring both at the Inferior as such and at the associated  
3231 Enroller, as the Enroller’s actions are constrained by and constrain the Inferior Role itself.

3232 In the state tables, each side is either waiting to make a decision or can send a message. For some  
3233 states, the message to be sent is a repetition of a regular message; for other states, the  
3234 INFERIOR\_STATE or SUPERIOR\_STATE message can be sent, requesting a response.  
3235 Normally, on entry to a state that allows the sending of any message other than one of the  
3236 \*\_STATE messages, the implementation will send that message – failure to do so will cause the

3237 relationship to lock up. The message can be resent if the implementation determines that the  
3238 original message (or the next message sent in reply) may have been lost.

## 3239 **8.1 Status queries**

3240 In BTP the messages SUPERIOR\_STATE and INFERIOR\_STATE are available to prompt the  
3241 Peer to report its current state by repeating the previous message (when this is allowed) or by  
3242 sending the other \*\_STATE message. The “reply\_requested” parameter of these messages  
3243 distinguishes between their use as a prompt and as a reply. An implementation receiving a  
3244 \*\_STATE message with “reply\_requested” as “true” is not required to reply immediately – it may  
3245 choose to delay any reply until a decision event occurs and then send the appropriate new  
3246 message (e.g. on receiving INFERIOR\_STATE/prepared/y while in state E1, a superior is  
3247 permitted to delay until it has performed “decide to confirm” or “decide to cancel”). However,  
3248 this may cause the other side to repeatedly send interrogatory \*\_STATE messages.

3249 Note that a Superior (or some entity standing in for a now-extinct Superior) uses  
3250 SUPERIOR\_STATE/unknown to reply to messages received from an Inferior where the  
3251 Superior:Inferior relationship is in an unknown (using state “Y1”). The \*\_STATE messages with  
3252 a “state” value “inaccessible” can be used as a reply when **any** message is received and the  
3253 implementation is temporarily unable to determine whether the relationship is known or what the  
3254 state is. Receipt of the \*\_STATE/inaccessible messages is not shown in the tables and has no  
3255 effect on the state at the receiving side (though it may cause the implementation to resend its own  
3256 message after some interval of its own choosing).

## 3257 **8.2 Decision events**

3258 The persistent state changes (equivalent to logging in a regular transaction system) and some  
3259 other events are modelled as “decision events” (e.g. “decide to confirm”, “decide to be  
3260 prepared”). The exact nature of the real events and changes in an implementation that are  
3261 modelled by these events depends on the position of the Superior or Inferior within the Business  
3262 Transaction and on features of the implementation (e.g. making of a persistent record of the  
3263 decision means that the information will survive at least some failures that otherwise lose state  
3264 information, but the level of survival depends on the purpose of the implementation). Table 3 and  
3265 Table 4 define the decision events.

3266 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of  
3267 PREPARE indicates that the application exchange (to associate operations with the Inferior) is  
3268 complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier state  
3269 corresponding to an incomplete application exchange. However, implementations are not required  
3270 to make the sending of PREPARE persistent in terms of recovery – a Superior that experiences  
3271 failure after sending PREPARE may, on recovery, have no information about the transaction, in  
3272 which case it is considered to be in the completed state (Z), which will imply the cancellation of  
3273 the Inferior and its associated operations.

3274 Where a Superior is an Intermediate (i.e. is itself an Inferior to another Superior entity), in a  
3275 Transaction Tree, its “decide to confirm” and “decide to cancel” decisions will in fact be the  
3276 receipt of a CONFIRM or CANCEL instruction from its own Superior, without necessary change  
3277 of local persistent information (which would combine both superior and inferior information,  
3278 pointing both up and down the tree).

3279 **8.3 Disruptions – failure events**

3280 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or may)  
3281 cause a change of state. The disruption events in the state tables model different extents of loss of  
3282 state information. An implementation is **not** required to exhibit all the possible disruption events,  
3283 but it is not allowed to exhibit state transitions that do not correspond to a possible disruption.  
3284 The different levels of disruption describe legitimate states for the endpoint to be in after it has  
3285 been restored to normal functioning. The absence of a destination state for the disruption events  
3286 means that such a transition is not legitimate – thus, for example, an Inferior that has decided to  
3287 be prepared will always recover to the same state, by virtue of the information persisted in the  
3288 “decide to be prepared” event.

3289 In addition to the disruption events in the tables, there is an implicit “disruption 0” event, which  
3290 involves possible interruption of service and loss of messages in transit, but no change of state  
3291 (either because no state information was lost, or because recovery from persistent information  
3292 restores the implementation to the same state). The “disruption 0” event would typically be an  
3293 appropriate abstraction for a communication failure.

3294 **8.4 Invalid cells and assumptions of the communication mechanism**

3295 The empty cells in state table represent events that cannot happen. For events corresponding to  
3296 sending a message or any of the decision events, this prohibition is absolute – e.g. a conformant  
3297 implementation in the Superior active state “B1” will not send CONFIRM. For events  
3298 corresponding to receiving a message, the interpretation depends on the properties of the  
3299 underlying communications mechanism.

3300 For all communication mechanisms, it is assumed that

3301 a) the two directions of the Superior:Inferior communication are not synchronised –  
3302 that is messages travelling in opposite directions can cross each other to any  
3303 degree; any number of messages may be in transit in either direction; and

3304 b) messages may be lost arbitrarily

3305 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered at  
3306 all, are delivered to the receiver in the order they were sent) , then receipt of a message in a state  
3307 where the corresponding cell is empty indicates that the far-side has sent a message out of order –  
3308 a FAULT message with the “fault-type” “WrongState” can be returned.

3309 If the communication mechanisms cannot guarantee ordered delivery, then messages received  
3310 where the corresponding cell is empty should be ignored. Assuming the far-side is conformant,  
3311 these messages can assumed to be “stale” and have been overtaken by messages sent later but  
3312 already delivered. (If the far-side is non-conformant, there is a problem anyway).

3313 **8.5 Meaning of state table events**

3314 The tables in this section define the events (rows) in the state tables. Table 2 defines the events  
3315 corresponding to sending or receiving BTP messages and the disruption events. Table 3 describes  
3316 the decision events for an Inferior, Table 4 those for a Superior.

3317 The decision events for a Superior, defined in Table 4 cannot be specified without reference to  
 3318 other Inferiors to which it is Superior and to its relation with the application or other entity that  
 3319 (acting ultimately on behalf of the application) drives it.

3320 The term “remaining Inferiors” refers to any Actors to which this endpoint is Superior and which  
 3321 are to be treated as an atomic decision unit with (and thus including) the Inferior on this  
 3322 relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior-type” of  
 3323 “atom”, this will be all Inferiors established with same Superior address and “superior-identifier”  
 3324 except those from which RESIGN has been received. If the CONTEXT had “superior-type” of  
 3325 “cohesion”, the “remaining Inferiors” excludes any that it has been determined will be cancelled,  
 3326 as well as any that have resigned – in other words it includes only those for which a Confirm  
 3327 decision is still possible or has been made. The determination of exactly which Inferiors are  
 3328 “remaining Inferiors” in a Cohesion is determined, in some way, by the application. The term  
 3329 “Other remaining Inferiors” excludes this Inferior on this relationship. A Superior with a single  
 3330 Inferior will have no “other remaining Inferiors”.

3331 In order to ensure that the confirmation decision is delivered to all remaining Inferiors, despite  
 3332 failures, the Superior must persistently record which these Inferiors are (i.e. their addresses and  
 3333 identifiers). It must also either record that the decision is Confirm, or ensure that the Confirm  
 3334 decision (if there is one) is persistently recorded somewhere else, and that it will be told about it.  
 3335 This latter would apply if the Superior were also BTP Inferior to another entity which persisted a  
 3336 Confirm decision (or recursively deferred it still higher). However, since there is no requirement  
 3337 that the Superior be also a BTP Inferior to any other entity, the behaviour of asking another entity  
 3338 to make (and persist) the Confirm decision is termed "offering confirmation" - the Superior offers  
 3339 the possible confirmation of itself, and its remaining Inferiors to some other entity. If that entity  
 3340 (or something higher up) then does make and persist a Confirm decision, the Superior is  
 3341 "instructed to confirm" (which is equivalent BTP CONFIRM).

3342 The application, or an entity acting indirectly on behalf of the application, may request a Superior  
 3343 to prepare an Inferior (or all Inferiors). This typically implies that there will be no more  
 3344 operations associated with the Inferior. Following a request to prepare all remaining Inferiors, the  
 3345 Superior may offer confirmation to the entity that requested the prepare. (If the Superior is also a  
 3346 BTP Inferior, its superior can be considered an entity acting on behalf of the application.)

3347 The application, or an entity acting indirectly on behalf of the application, may also request  
 3348 confirmation. This means the Superior is to attempt to make and persist a Confirm decision itself,  
 3349 rather than offer confirmation.

3350 **Table 2 : send, receive and disruption events**

Event name	Meaning
send/receive ENROL/rsp-req	send/receive ENROL with response-requested = true
send/receive ENROL/no-rsp-req	send/receive ENROL with response-requested = false
send/receive RESIGN/rsp-req	send/receive RESIGN with response-requested = true
send/receive RESIGN/no-rsp-req	send/receive RESIGN with response-requested = false
send/receive PREPARED	send/receive PREPARED, with default-cancel = false

Event name	Meaning
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

3351

3352

**Table 3 : Decision events for Inferior**

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

Event name	Meaning
decide to cancel autonomously	<ul style="list-style-type: none"> <li>Decision to Cancel autonomously has been made persistent</li> <li>the effects of associated operations will be cancelled regardless of failures</li> </ul>
apply ordered confirmation	<ul style="list-style-type: none"> <li>Effects of all associated operations have been confirmed;</li> <li>Persistent information is effectively removed</li> </ul>
remove persistent information	<ul style="list-style-type: none"> <li>Persistent information is effectively removed;</li> </ul>
detect problem	<ul style="list-style-type: none"> <li>For at least some of the associated operations, EITHER <ul style="list-style-type: none"> <li>they cannot be consistently cancelled or consistently confirmed; OR</li> <li>it cannot be determined whether they will be cancelled or confirmed</li> </ul> </li> <li>AND, information about this is not persistent</li> </ul>
detect and record problem	<ul style="list-style-type: none"> <li>As for the first condition of "detect problem"</li> <li>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)</li> </ul>

3353

3354

**Table 4: Decision events for a Superior**

Event name	Meaning
decide to confirm one-phase	<ul style="list-style-type: none"> <li>All associated Application Messages to be sent to the service have been sent;</li> <li>There are no other remaining Inferiors</li> <li>If an Atom, all enrolments that would create other Inferiors have completed (no outstanding CONTEXT_REPLYS)</li> <li>The Superior has been requested to confirm</li> </ul>
decide to prepare	<ul style="list-style-type: none"> <li>All associated Application Messages to be sent to the service have been sent;</li> <li>The Superior has been requested to prepare this Inferior</li> </ul>
decide to confirm	<ul style="list-style-type: none"> <li>Either <ul style="list-style-type: none"> <li>PREPARED or PREPARED/cancel has been received from all other remaining Inferiors; AND</li> </ul> </li> </ul>

Event name	Meaning
	<ul style="list-style-type: none"> <li>o Superior has been requested to confirm; AND</li> <li>o persistent information records the confirm decision and identifies all remaining Inferiors;</li> <li>• Or</li> <li>o persistent information records an offer of confirmation and has been instructed to confirm</li> </ul>
decide to cancel	<ul style="list-style-type: none"> <li>• Superior has not offered confirmation; OR</li> <li>• Superior has offered confirmation and has been instructed to Cancel; OR</li> <li>• Superior has offered confirmation but has made an autonomous cancellation decision</li> </ul>
remove confirm information	<ul style="list-style-type: none"> <li>• Persistent information has been effectively removed;</li> </ul>
record contradiction	<ul style="list-style-type: none"> <li>• Information recording the contradiction has been persisted (to the degree considered appropriate)</li> </ul>

3355

## 3356 8.6 Persistent information

3357 Persisted information (especially prepared information at an Inferior, confirm information at a  
3358 Superior) may include qualifications of the state carried in Qualifiers of the corresponding  
3359 message (e.g. inferior timeouts in prepared information). It may also include application-specific  
3360 information (especially in Inferiors) to allow the future confirmation or cancellation of the  
3361 associated operations. In some cases it will also include information allowing an Application  
3362 Message sent with a BTP message (e.g. PREPARED) to be repeated.

3363 The “effective” removal of persistent information allows for the possibility that the information is  
3364 retained (perhaps for audit and tracing purposes) but some change to the persistent information  
3365 (as a whole) means that if there is a failure after such change, on recovery, the persistent  
3366 information does not cause the endpoint to return the state it would have recovered to before the  
3367 change.

3368 In all cases, the degree to which information described as “persistent” will survive failure is a  
3369 configuration and implementation option. An implementation **should** describe the level of failure  
3370 that it is capable of surviving. For applications manipulating information that is itself volatile (e.g.  
3371 network configurations), there is no requirement to make the BTP state information more  
3372 persistent than the application information.

3373 The degree of persistence of the recording of a hazard (problem) at an Inferior and recording of a  
3374 detected contradiction at a Superior may be different from that applying to the persistent prepared  
3375 and confirm information. Implementations and configuration may choose to pass hazard and  
3376 contradiction information via management mechanisms rather than through BTP. Such passing of  
3377 information to a management mechanism could be treated as “record problem” or “record  
3378 contradiction”.

**Table 5 : Superior states**

State	summary
I1	CONTEXT created
A1	ENROLing
B1	ENROLLED (active)
B2	ENROLLED – repeat ENROL received
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 6 : Inferior states**

State	summary
i1	aware of CONTEXT
a1	enrolling
b1	enrolled
c1	resigning
d1	preparing
e1	prepared
e2	prepared,default to cancel
f1	confirming
f2	confirming after default cancel
g1	CANCEL received in prepared state
g2	CANCEL received in prepared/cancel state
h1	Autonomously confirmed
h2	autonomously confirmed, superior confirmed
j1	autonomously cancelled
j2	autonomously cancelled, superior cancelled
k1	autonomously cancelled, contradicted
k2	autonomously cancelled, CONTRADICTION received
l1	autonomously confirmed, contradicted
l2	autonomously confirmed, CONTRADICTION received
m1	confirmation applied
n1	cancelling
p1	hazard detected, not recorded
p2	hazard detected in prepared state, not recorded
q1	hazard recorded
s1	CONFIRM_ONE_PHASE received after prepared state
s2	CONFIRM_ONE_PHASE received
s3	CONFIRM_ONE_PHASE received, confirming
s4	CONFIRM_ONE_PHASE received, cancelling
s5	CONFIRM_ONE_PHASE received, hazard detected
s6	CONFIRM_ONE_PHASE received, hazard recorded
x1	completed, presuming abort
x2	completed, presuming abort after prepared/cancel
y1	completed, queried

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

3382

3383 **8.7 Superior state table**

3384

**Table 7: Superior state table – normal forward progression**

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

3385

3386

**Table 8: 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		

3387

3388

**Table 9: 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	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 10: 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		

3391

3392

3392 **8.8 Inferior state table**

3393 **Table 11: Inferior state table – normal forward progression**

	i 1	a1	b1	c1	d1	e1	e2	f1	f2
send ENROL/rsp-req	a1	a1							
send ENROL/no-rsp-req	b1		b1						
send RESI GN/rsp-req				c1					
send RESI GN/no-rsp-req				z					
send PREPARED						e1			
send PREPARED/cancel							e2		
send CONFIR MED/auto									
send CONFIR MED/response									
send CANCELLED			z		z				
send HAZARD									
send INF_STATE/active/y		a1	b1		d1				
send INF_STATE/active			b1		d1				
send INF_STATE/unknown									
receive ENROLLED		b1	b1	c1		e1	e2		
receive RESI GNED				z					
receive PREPARE		d1	d1	c1	d1	e1	e2		
receive CONFIR M_ONE_PHASE		s2	s2	z		s1	s1		
receive CONFIR M						f1	f2	f1	f2
receive CANCEL		n1	n1	z	n1	g1	g2		
receive CONTRADI CTI ON									
receive SUP_STATE/active/y		b1	b1	c1		e1	e2		
receive SUP_STATE/active		b1	b1	c1		e1	e2		
receive SUP_STATE/prepared-rcvd/y						e1	e2		
receive SUP_STATE/prepared-rcvd						e1	e2		
receive SUP_STATE/unknown		z	z	z	z	x1	x2		
decide to resi gn			c1		c1				
decide to be prepared			e1		e1				
decide to be prepared/cancel			e2		e2				
decide to confi rm autonomously						h1			
decide to cancel autonomously						j 1	z1		
apply ordered confi rmation								m1	m1
remove persi stent i nformati on									
detect probl em		p1	p1		p1	p2	p2	p2	p2
detect and record probl em									
di srupti on I		z	z	z	z			e1	e2
di srupti on II					b1				
di srupti on III									

3394

3395

**Table 12: Inferior state table – cancellation and contradiction**

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

3396

3397

**Table 13: 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 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 p2 p2 p1	p2 p2 p2 p2 p2	q1 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		

3398

3399

**Table 14: Inferior state table – request confirm states**

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

3400

3401

3401

**Table 15: 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   x1	y2 y2 y2 y2 x2	y1 z y2 y2 z	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				

3402

3403

## 3403 **9 Persistent information**

3404 The BTP recovery mechanisms require that information is persisted by the BTP Actors that  
3405 perform the Superior and Inferior roles. To ensure consistent application of the outcome, despite  
3406 failures, the Inferior must persist some state information at the point of becoming prepared, and  
3407 the Superior at the point of making a Confirm decision. If the Superior is a Sub-coordinator or  
3408 Sub-composer, it must persist information when, as an Inferior it becomes prepared. The  
3409 minimum information to be persisted is the identifiers and addresses of the Peer Inferiors and  
3410 Superior – the fact of the persistence being itself an indication of the preparedness or Confirm  
3411 decision. However, BTP allows recovery of a Superior:Inferior relationship to occur in other  
3412 cases – during the active phase, and before a Confirm decision has been made. Thus, in general,  
3413 the BTP Actors will need to persist the current state of the relationships.

3414 Since BTP messages may carry application-specified qualifiers, which may need to be re-sent in  
3415 the case of failure (because the first attempt got lost). BTP Actors should be prepared to persist  
3416 such qualifiers as well.

3417 A Participant will normally also need to persist some information concerning the application  
3418 work whose final or counter effect it is responsible for. The nature of this information is not  
3419 considered further in this specification.

3420 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to re-  
3421 establish communication with the Superior, to apply a Confirm decision and to apply a Cancel  
3422 decision. It will thus need to include

3423 “superior-address”(as on CONTEXT as updated by REDIRECT)

3424 “superior-identifier” (as on CONTEXT)

3425 “default-is-cancel” value (as on PREPARED)

3426 A Superior must record corresponding information to allow it to re-establish communication with  
3427 the Inferior. Thus, for each Inferior

3428 “inferior-address” (as on ENROL, as updated by REDIRECT)

3429 “inferior-identifier” (as on ENROL)

3430 In order to recover their own function, both Superior and Inferior will need to persist their own  
3431 Identifier (“superior-identifier” and “inferior-identifier”) and, depending on the implementation,  
3432 may need to persist their original “superior-address” or “inferior-address”.

## 3433 **10 XML representation of Message Set**

3434 This section describes the syntax for BTP messages in XML. These XML messages represent a  
3435 midpoint between the abstract messages and what actually gets sent on the wire.

3436 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC 3121](#)

3437 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:1.0:core

3438 In addition to an XML schema, this specification uses an informal syntax to describe the structure  
3439 of the BTP messages. The syntax appears as an XML instance, but the values contain data types  
3440 instead of values. The following symbols are appended to some of the XML constructs: ? (zero  
3441 or one), \* (zero or more), + (one or more.) The absence of one of these symbols corresponds to  
3442 "one and only one."

3443 The Delivery Parameters are shown in the XML with a darker background.

## 3444 **10.1 Field types**

### 3445 **10.1.1 Addresses**

3446 As described in the “Abstract Message and Associated Contracts – Addresses” section, a BTP  
3447 Address comprises three parts, and for a “target-address” only the “additional information” field  
3448 is inside the BTP messages. For all BTP messages whose abstract form includes a “target-  
3449 address” parameter, the corresponding XML representation includes a “target-additional-  
3450 information” element. This element may be omitted if it would be empty.

3451 For other addresses, all three fields are represent, as in:

```
3452 <btq:some-address>  
3453   <btq:binding-name>...carrier binding name...</btq:binding-name>  
3454   <btq:binding-address>...carrier specific  
3455   address...</btq:binding-address>  
3456   <btq:additional-information>...optional additional addressing  
3457   information...</btq:additional-information> ?  
3458 </btq:some-address>  
3459
```

3460 A "published" address can be a set of <some-address>, which are alternatives which can be  
3461 chosen by the Peer (sender.) Multiple addresses are used in two cases: different bindings to same  
3462 endpoint, or backup endpoints. In the former, the receiver of the message has the choice of which  
3463 address to use (depending on which binding is preferable.) In the case where multiple addresses  
3464 are used for redundancy, a priority attribute can be specified to help the receiver choose among  
3465 the addresses- the address with the highest priority should be used, other things being equal. The  
3466 priority is used as a hint and does not enforce any behaviour in the receiver of the message.  
3467 Default priority is a value of 1.

### 3468 **10.1.2 Qualifiers**

3469 The “Qualifier name” is used as the element name, within the namespace of the “Qualifier  
3470 group”.

3471 **Examples:**

```
3472 <btqp:inferior-timeout  
3473   xmlns:btqp="urn:oasis:names:tc:BTP:1.0:qualifiers"  
3474   xmlns:btq="urn:oasis:names:tc:BTP:1.0:core"  
3475   btq:must-be-understood="false"  
3476   btq:to-be-propagated="false">1800</btqp:inferior-timeout>  
3477 <auth:username
```

```
3478     xmlns:auth="http://www.example.com/ns/auth"
3479     xmlns:btp="urn:oasis:names:tc:BTP:1.0:core"
3480     btp:must-be-understood="true"
3481     btp:to-be-propagated="true">jtauber</auth:username>
3482
```

3483 Attributes **must-be-understood** has default value **“true”** and **to-be-propagated** has default value  
3484 **“false”**.

### 3485 10.1.3 Identifiers

3486 Identifiers shall be URIs "

3487 *Note – Identifiers need to be globally unambiguous. Apart from their generation, .the*  
3488 *only operation the BTP implementations have to perform on identifiers is to match*  
3489 *them.*

### 3490 10.1.4 Message References

3491 Each BTP message has an optional id attribute to give it a unique identifier. An application can  
3492 make use of those identifiers, but no processing is enforced.

## 3493 10.2 Messages

### 3494 10.2.1 CONTEXT

```
3495 <btp:context id?>
3496   <btp:superior-address> +
3497     ...address...
3498   </btp:superior-address>
3499   <btp:superior-identifier>...URI...</btp:superior-identifier>
3500   <btp:superior-type>cohesion|atom</btp:superior-type>
3501   <btp:qualifiers> ?
3502     ...qualifiers...
3503   </btp:qualifiers>
3504   <btp:reply-address> ?
3505     ...address...
3506   </btp:reply-address>
3507 </btp:context>
```

### 3508 10.2.2 CONTEXT\_REPLY

```
3509 <btp:context-reply id?>
3510   <btp:superior-identifier>...URI...</btp:superior-identifier>
3511   <btp:completion-
3512 status>completed|incomplete|related|repudiated</btp:completion-
3513 status>
3514   <btp:qualifiers> ?
3515     ...qualifiers...
3516   </btp:qualifiers>
3517   <btp:target-additional-information> ?
3518     ...additional address information...
3519   </btp:target-additional-information>
3520 </btp:context-reply>
```

### 3521 10.2.3 REQUEST\_STATUS

```
3522 <btpr:request-status id?>
3523   <btpr:target-identifier>...URI...</btpr:target-identifier>
3524   <btpr:qualifiers> ?
3525     ...qualifiers...
3526 </btpr:qualifiers>
3527 <btpr:target-additional-information> ?
3528   ...additional address information...
3529 </btpr:target-additional-information>
3530 <btpr:reply-address> ?
3531   ...address...
3532 </btpr:reply-address>
3533 </btpr:request-status>
```

### 3534 10.2.4 STATUS

```
3535 <btpr:status id?>
3536   <btpr:responders-identifier>...URI...</btpr:responders-identifier>
3537   <btpr:status-value>created|enrolling|active|resigning|
3538     resigned|preparing|prepared|
3539     confirming|confirmed|cancelling|cancelled|
3540     cancel-contradiction|confirm-contradiction|
3541     hazard|contradicted|unknown|inaccessible</btpr:status-
3542 value>
3543   <btpr:qualifiers> ?
3544     ...qualifiers...
3545 </btpr:qualifiers>
3546 <btpr:target-additional-information> ?
3547   ...additional address information...
3548 </btpr:target-additional-information>
3549 </btpr:status>
```

### 3550 10.2.5 FAULT

```
3551 <btpr:fault id?>
3552   <btpr:superior-identifier>...URI...</btpr:superior-identifier> ?
3553   <btpr:inferior-identifier>...URI...</btpr:inferior-identifier> ?
3554   <btpr:fault-type>...fault type name...</btpr:fault-type>
3555   <btpr:fault-data>...fault data...</btpr:fault-data> ?
3556   <btpr:fault-text>...string data ...</btpr:fault-data> ?
3557   <btpr:qualifiers> ?
3558     ...qualifiers...
3559 </btpr:qualifiers>
3560 <btpr:target-additional-information> ?
3561   ...additional address information...
3562 </btpr:target-additional-information>
3563 </btpr:fault>
3564
```

3565 The following fault type names are represented by simple strings, corresponding to the entries  
3566 defined in the abstract message set:

- 3567           • communication-failure
- 3568           • duplicate-inferior
- 3569           • general
- 3570           • invalid-decider
- 3571           • invalid-inferior
- 3572           • invalid-superior
- 3573           • status-refused
- 3574           • invalid-terminator
- 3575           • unknown-parameter
- 3576           • unknown-transaction
- 3577           • unsupported-qualifier
- 3578           • wrong-state
- 3579           • redirect
- 3580

3581 Revisions of this specification may add other fault type names, which shall be simple strings of  
 3582 letters, numbers and hyphens. If other specifications define fault type names to be used with BTP,  
 3583 the names shall be URIs.

3584 Fault data can take on various forms:

3585 Identifier:

```
3586           <btp: fault-data>...URI...</btp: fault-data>
```

3588 Inferior Identity:

```
3589           <btp: fault-data>
3590            <btp: inferior-address> +
3591            ...address...
3592           </btp: inferior-address>
3593           <btp: inferior-identifier>...URI...</btp: inferior-identifier>
3594           </btp: fault-data>
```

### 3596 10.2.6 ENROL

```
3597           <btp: enrol id?>
3598            <btp: superior-identifier>...URI...</btp: superior-identifier>
3599            <btp: response-requested>true| false</btp: response-requested>
3600            <btp: inferior-address> +
3601            ...address...
3602            </btp: inferior-address>
3603            <btp: inferior-identifier>...URI...</btp: inferior-identifier>
3604            <btp: qualifiers> ?
3605            ...qualifiers...
```

```
3606 </btp:qualifiers>
3607 <btp:target-additional-information> ?
3608 ...additional address information...
3609 </btp:target-additional-information>
3610 <btp:reply-address> ?
3611 ...address...
3612 </btp:reply-address>
3613 </btp:enrol>
```

## 3614 10.2.7 ENROLLED

```
3615 <btp:enrolled id?>
3616 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3617 <btp:qualifiers> ?
3618 ...qualifiers...
3619 </btp:qualifiers>
3620 <btp:target-additional-information> ?
3621 ...additional address information...
3622 </btp:target-additional-information>
3623 <btp:sender-address> ?
3624 ...address...
3625 </btp:sender-address>
3626 </btp:enrolled>
```

## 3627 10.2.8 RESIGN

```
3628 <btp:resign id?>
3629 <btp:superior-identifier>...URI...</btp:superior-identifier>
3630 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3631 <btp:response-requested>true|false</btp:response-requested>
3632 <btp:qualifiers> ?
3633 ...qualifiers...
3634 </btp:qualifiers>
3635 <btp:target-additional-information> ?
3636 ...additional address information...
3637 </btp:target-additional-information>
3638 <btp:sender-address> ?
3639 ...address...
3640 </btp:sender-address>
3641 </btp:resign>
```

## 3642 10.2.9 RESIGNED

```
3643 <btp:resigned id?>
3644 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3645 <btp:qualifiers> ?
3646 ...qualifiers...
3647 </btp:qualifiers>
3648 <btp:target-additional-information> ?
3649 ...additional address information...
3650 </btp:target-additional-information>
3651 <btp:sender-address> ?
3652 ...address...
3653 </btp:sender-address>
```

3654 </btp:resigned>

### 3655 10.2.10 PREPARE

```
3656 <btp:prepare id?>
3657 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3658 <btp:qualifiers> ?
3659 ...qualifiers...
3660 </btp:qualifiers>
3661 <btp:target-additional-information> ?
3662 ...additional address information...
3663 </btp:target-additional-information>
3664 <btp:sender-address> ?
3665 ...address...
3666 </btp:sender-address>
3667 </btp:prepare>
```

### 3668 10.2.11 PREPARED

```
3669 <btp:prepared id?>
3670 <btp:superior-identifier>...URI...</btp:superior-identifier>
3671 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3672 <btp:default-is-cancel>true|false</btp:default-is-cancel>
3673 <btp:qualifiers> ?
3674 ...qualifiers...
3675 </btp:qualifiers>
3676 <btp:target-additional-information> ?
3677 ...additional address information...
3678 </btp:target-additional-information>
3679 <btp:sender-address> ?
3680 ...address...
3681 </btp:sender-address>
3682 </btp:prepared>
```

### 3683 10.2.12 CONFIRM

```
3684 <btp:confirm id?>
3685 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3686 <btp:qualifiers> ?
3687 ...qualifiers...
3688 </btp:qualifiers>
3689 <btp:target-additional-information> ?
3690 ...additional address information...
3691 </btp:target-additional-information>
3692 <btp:sender-address> ?
3693 ...address...
3694 </btp:sender-address>
3695 </btp:confirm>
```

### 3696 10.2.13 CONFIRMED

```
3697 <btp:confirmed id?>
3698 <btp:superior-identifier>...URI...</btp:superior-identifier>
3699 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
```

```
3700 <btpp:confirmed-received>true|false</btpp:confirmed-received>
3701 <btpp:qualifiers> ?
3702   ...qualifiers...
3703 </btpp:qualifiers>
3704 <btpp:target-additional-information> ?
3705   ...additional address information...
3706 </btpp:target-additional-information>
3707 <btpp:sender-address> ?
3708   ...address...
3709 </btpp:sender-address>
3710 </btpp:confirmed>
```

#### 3711 10.2.14 CANCEL

```
3712 <btpp:cancel id?>
3713 <btpp:inferior-identifier>...URI...</btpp:inferior-identifier>
3714 <btpp:qualifiers> ?
3715   ...qualifiers...
3716 </btpp:qualifiers>
3717 <btpp:target-additional-information> ?
3718   ...additional address information...
3719 </btpp:target-additional-information>
3720 <btpp:sender-address> ?
3721   ...address...
3722 </btpp:sender-address>
3723 </btpp:cancel>
```

#### 3724 10.2.15 CANCELLED

```
3725 <btpp:cancelled id?>
3726 <btpp:superior-identifier>...URI...</btpp:superior-identifier>
3727 <btpp:inferior-identifier>...URI...</btpp:inferior-identifier> ?
3728 <btpp:qualifiers> ?
3729   ...qualifiers...
3730 </btpp:qualifiers>
3731 <btpp:target-additional-information> ?
3732   ...additional address information...
3733 </btpp:target-additional-information>
3734 <btpp:sender-address> ?
3735   ...address...
3736 </btpp:sender-address>
3737 </btpp:cancelled>
```

#### 3738 10.2.16 CONFIRM\_ONE\_PHASE

```
3739 <btpp:confirm-one-phase id?>
3740 <btpp:inferior-identifier>...URI...</btpp:inferior-identifier>
3741 <btpp:report-hazard>true|false</btpp:report-hazard>
3742 <btpp:qualifiers> ?
3743   ...qualifiers...
3744 </btpp:qualifiers>
3745 <btpp:target-additional-information> ?
3746   ...additional address information...
3747 </btpp:target-additional-information>
```

```
3748 <btpt:sender-address> ?
3749 ...address...
3750 </btpt:sender-address>
3751 </btpt:confirm-one-phase>
```

## 3752 10.2.17 HAZARD

```
3753 <btpt:hazard id?>
3754 <btpt:superior-identifier>...URI...</btpt:superior-identifier>
3755 <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>
3756 <btpt:level>mixed|possible</btpt:level>
3757 <btpt:qualifiers> ?
3758 ...qualifiers...
3759 </btpt:qualifiers>
3760 <btpt:target-additional-information> ?
3761 ...additional address information...
3762 </btpt:target-additional-information>
3763 <btpt:sender-address> ?
3764 ...address...
3765 </btpt:sender-address>
3766 </btpt:hazard>
```

## 3767 10.2.18 CONTRADICTION

```
3768 <btpt:contradiction id?>
3769 <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>
3770 <btpt:qualifiers> ?
3771 ...qualifiers...
3772 </btpt:qualifiers>
3773 <btpt:target-additional-information> ?
3774 ...additional address information...
3775 </btpt:target-additional-information>
3776 <btpt:sender-address> ?
3777 ...address...
3778 </btpt:sender-address>
3779 </btpt:contradiction>
```

## 3780 10.2.19 SUPERIOR\_STATE

```
3781 <btpt:superior-state id?>
3782 <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>
3783 <btpt:status>active|prepared-
3784 received|inaccessible|unknown</btpt:status>
3785 <btpt:response-requested>true|false</btpt:response-requested>
3786 <btpt:qualifiers> ?
3787 ...qualifiers...
3788 </btpt:qualifiers>
3789 <btpt:target-additional-information> ?
3790 ...additional address information...
3791 </btpt:target-additional-information>
3792 <btpt:sender-address> ?
3793 ...address...
3794 </btpt:sender-address>
3795 </btpt:superior-state>
```

## 3796 10.2.20 INFERIOR\_STATE

```
3797 <btpr:inferior-state id?>
3798   <btpr:superior-identifier>...URI...</btpr:superior-identifier>
3799   <btpr:inferior-identifier>...URI...</btpr:inferior-identifier>
3800   <btpr:status>active|inaccessible|unknown</btpr:status>
3801   <btpr:response-requested>true|false</btpr:response-requested>
3802   <btpr:qualifiers> ?
3803     ...qualifiers...
3804 </btpr:qualifiers>
3805 <btpr:target-additional-information> ?
3806   ...additional address information...
3807 </btpr:target-additional-information>
3808 <btpr:sender-address> ?
3809   ...address...
3810 </btpr:sender-address>
3811 </btpr:inferior-state>
```

## 3812 10.2.21 REDIRECT

```
3813 <btpr:redirect id?>
3814   <btpr:superior-identifier>...URI...</btpr:superior-identifier> ?
3815   <btpr:inferior-identifier>...URI...</btpr:inferior-identifier>
3816   <btpr:old-address> +
3817     ...address...
3818 </btpr:old-address>
3819   <btpr:new-address> +
3820     ...address...
3821 </btpr:new-address>
3822   <btpr:qualifiers> ?
3823     ...qualifiers...
3824 </btpr:qualifiers>
3825 <btpr:target-additional-information> ?
3826   ...additional address information...
3827 </btpr:target-additional-information>
3828 </btpr:redirect>
```

## 3829 10.2.22 BEGIN

```
3830 <btpr:begin id?>
3831   <btpr:transaction-type>cohesion|atom</btpr:transaction-type>
3832   <btpr:qualifiers> ?
3833     ...qualifiers...
3834 </btpr:qualifiers>
3835 <btpr:target-additional-information> ?
3836   ...additional address information...
3837 </btpr:target-additional-information>
3838 <btpr:reply-address> ?
3839   ...address...
3840 </btpr:reply-address>
3841 </btpr:begin>
```

### 3842 10.2.23 BEGUN

```
3843 <btp:begin id?>
3844   <btp:decider-address> *
3845     ...address...
3846   </btp:decider-address>
3847   <btp:inferior-address> *
3848     ...address...
3849   </btp:inferior-address>
3850   <btp:transaction-identifier>...URI...</btp:transaction-
3851 identifier>
3852   <btp:qualifiers> ?
3853     ...qualifiers...
3854   </btp:qualifiers>
3855   <btp:target-additional-information> ?
3856     ...additional address information...
3857   </btp:target-additional-information>
3858 </btp:begin>
```

### 3859 10.2.24 PREPARE\_INFERIORS

```
3860 <btp:prepare-inferiors id?>
3861   <btp:transaction-identifier>...URI...</btp:transaction-
3862 identifier>
3863   <btp:inferiors-list> ?
3864     <btp:inferior-identifier>...URI...</btp:inferior-
3865 identifier> +
3866   </btp:inferiors-list>
3867   <btp:qualifiers> ?
3868     ...qualifiers...
3869   </btp:qualifiers>
3870   <btp:target-additional-information> ?
3871     ..additional address information...
3872   </btp:target-additional-information>
3873   <btp:reply-address> ?
3874     ...address...
3875   </btp:reply-address>
3876 </btp:prepare-inferiors>
```

### 3877 10.2.25 CONFIRM\_TRANSACTION

```
3878 <btp:confirm-transaction id?>
3879   <btp:transaction-identifier>...URI...</btp:transaction-
3880 identifier>
3881   <btp:inferiors-list> ?
3882     <btp:inferior-identifier>...URI...</btp:inferior-
3883 identifier> +
3884   </btp:inferiors-list>
3885   <btp:report-hazard>true|false</btp:report-hazard>
3886   <btp:qualifiers> ?
3887     ...qualifiers...
3888   </btp:qualifiers>
3889   <btp:target-additional-information> ?
3890     ...additional address information...
```

```
3891     </btp:target-additional-information>
3892     <btp:reply-address> ?
3893         ...address...
3894     </btp:reply-address>
3895 </btp:confirm_transaction>
```

#### 3896 10.2.26 TRANSACTION\_CONFIRMED

```
3897 <btp:transaction-confirmed id?>
3898     <btp:transaction-identifier>...URI...</btp:transaction-
3899     identifier>
3900     <btp:qualifiers> ?
3901         ...qualifiers...
3902     </btp:qualifiers>
3903     <btp:target-additional-information> ?
3904         ...additional address information...
3905     </btp:target-additional-information>
3906 </btp:transaction-confirmed>
```

#### 3907 10.2.27 CANCEL\_TRANSACTION

```
3908 <btp:cancel-transaction id?>
3909     <btp:transaction-identifier>...URI...</btp:transaction-
3910     identifier>
3911     <btp:report-hazard>true|false</btp:report-hazard>
3912     <btp:qualifiers> ?
3913         ...qualifiers...
3914     </btp:qualifiers>
3915     <btp:target-additional-information> ?
3916         ...additional address information...
3917     </btp:target-additional-information>
3918     <btp:reply-address> ?
3919         ...address...
3920     </btp:reply-address>
3921 </btp:cancel-transaction>
```

#### 3922 10.2.28 CANCEL\_INFERIORS

```
3923 <btp:cancel-inferiors id?>
3924     <btp:transaction-identifier>...URI...</btp:transaction-
3925     identifier> ?
3926     <btp:inferiors-list>
3927         <btp:inferior-identifier>...URI...</btp:inferior-identifier> +
3928     </btp:inferiors-list>
3929     <btp:qualifiers> ?
3930         ...qualifiers...
3931     </btp:qualifiers>
3932     <btp:target-additional-information> ?
3933         ...additional address information...
3934     </btp:target-additional-information>
3935     <btp:reply-address> ?
3936         ...address...
3937     </btp:reply-address>
3938 </btp:cancel-inferiors>
```

### 3939 10.2.29 TRANSACTION\_CANCELLED

```
3940 <btpt:transaction-cancelled id?>
3941   <btpt:transaction-identifier>...URI...</btpt:transaction-
3942   identifier>
3943   <btpt:qualifiers> ?
3944     ...qualifiers...
3945   </btpt:qualifiers>
3946   <btpt:target-additional-information> ?
3947     ...additional address information...
3948   </btpt:target-additional-information>
3949 </btpt:transaction-cancelled>
```

### 3950 10.2.30 REQUEST\_INFERIOR\_STATUSES

```
3951 <btpt:request-inferior-statuses id?>
3952   <btpt:target-identifier>...URI...</btpt:target-identifier>
3953   <btpt:inferiors-list> ?
3954     <btpt:inferior-identifier>...URI...</btpt:inferior-
3955     identifier> +
3956   </btpt:inferiors-list>
3957   <btpt:qualifiers> ?
3958     ...qualifiers...
3959   </btpt:qualifiers>
3960   <btpt:target-additional-information> ?
3961     ...additional address information...
3962   </btpt:target-additional-information>
3963   <btpt:reply-address> ?
3964     ...address...
3965   </btpt:reply-address>
3966 </btpt:request-inferior-statuses>
```

### 3967 10.2.31 INFERIOR\_STATUSES

```
3968 <btpt:inferior-statuses id?>
3969   <btpt:responders-identifier>...URI...</btpt:responders-identifier>
3970   <btpt:status-list>
3971     <btpt:status-item> +
3972       <btpt:inferior-identifier>...URI...</btpt:inferior-
3973       identifier>
3974       <btpt:status>active|resigned|preparing|prepared|
3975       autonomously-confirmed|autonomously-cancelled|
3976       confirming|confirmed|cancelling|cancelled|
3977       cancel-contradiction|confirm-contradiction|
3978       hazard|invalid</btpt:status>
3979     <btpt:qualifiers> ?
3980       ...qualifiers...
3981     </btpt:qualifiers>
3982   </btpt:status-item>
3983 </btpt:status-list>
3984   <btpt:qualifiers> ?
3985     ...qualifiers...
3986   </btpt:qualifiers>
3987   <btpt:target-additional-information> ?
```

```
3988     ...additional address information...
3989     </btp:target-additional-information>
3990 </btp:inferior-statuses>
```

### 3991 **10.3 Standard qualifiers**

3992 The informal syntax for these messages assumes the namespace prefix “btpq” is associated with  
3993 the URI “urn:oasis:names:tc:BTP:1.0:qualifiers”.

#### 3994 **10.3.1 Transaction timelimit**

```
3995     <btpq:transaction-timelimit>
3996     <btpq:timelimit>
3997     ...time in seconds...
3998     </btpq:timelimit>
3999 </btpq:transaction-timelimit>
```

#### 4000 **10.3.2 Inferior timeout**

```
4001     <btpq:inferior-timeout>
4002     <btpq:timeout>
4003     ...time in seconds...
4004     </btpq:timeout>
4005     <btpq:intended-decision>confirm|cancel</btpq:intended-decision>
4006 </btpq:inferior-timeout>
```

#### 4007 **10.3.3 Minimum inferior timeout**

```
4008     <btpq:minimum-inferior-timeout>
4009     <btpq:minimum-timeout>
4010     ...time in seconds...
4011     </btpq:minimum-timeout>
4012 </btpq:minimum-inferior-timeout>
```

#### 4013 **10.3.4 Inferior name**

```
4014     <btpq:inferior-name>
4015     <btpq:inferior-name>
4016     ...string...
4017     </btpq:inferior-name>
4018 </btpq:inferior-name>
```

### 4019 **10.4 Compounding of Messages**

4020 Relating BTP to one another, in a “group” is represented by containing them within the  
4021 btp:related-group element, with the related messages as child elements. The processing for the  
4022 group is defined in the section “7.9 Groups – combinations of related messages”. For  
4023 example

```
4024     <btp:related-group>
4025     <btp:context-reply>
4026     ...<completion-status>related</completion-status> ...
```

```
4027         </btp:context-reply>
4028         <btp:enrol>...</btp:enrol>
4029         <btp:prepared>...</btp:prepared>
4030     </btp:related-group>
```

4031 If the rules for the group state that the “target-address” of the abstract message is omitted, the  
4032 corresponding target-address-information element shall be absent in the message in the related-  
4033 group. The Carrier Protocol binding specifies how a relation between application and BTP  
4034 messages is represented.

4035 Bundling (semantically insignificant combination) of BTP messages and related groups is  
4036 indicated with the "btp:messages" element, with the bundled messages and related groups as child  
4037 elements. For example (confirming one and cancelling another inferiors of a Cohesion):

```
4038     <btp:messages>
4039         <btp:confirm>...</btp:confirm>
4040         <btp:cancel>...</btp:cancel>
4041     </btp:messages>
4042
```

## 4043 10.5 XML Schemas

### 4044 10.5.1 XML schema for BTP messages

```
4045 <?xml version="1.0"?>
4046 <schema
4047     xmlns="http://www.w3.org/2001/XMLSchema"
4048     targetNamespace="urn:oasis:names:tc:BTP:1.0:core"
4049     xmlns:btp="urn:oasis:names:tc:BTP:1.0:core"
4050     elementFormDefault="qualified">
4051
4052     <!-- Qualifiers -->
4053
4054     <complexType name="qualifier-type">
4055         <complexContent mixed="true">
4056             <restriction base="anyType">
4057                 <sequence>
4058                     <any processContents="lax" minOccurs="0"
4059 maxOccurs="unbounded"/>
4060                 </sequence>
4061                 <attribute name="must-be-understood" type="boolean"
4062 default="true"/>
4063                 <attribute name="to-be-propagated" type="boolean"
4064 default="false"/>
4065             </restriction>
4066         </complexContent>
4067     </complexType>
4068
4069     <element name="qualifier" type="btp:qualifier-type" abstract="true"/>
4070
4071     <element name="qualifiers">
4072         <complexType>
```

```

4075         <choice>
4076             <element ref="btp:qualifier" minOccurs="0"
4077 maxOccurs="unbounded"/>
4078             <any processContents="lax" minOccurs="0"
4079 maxOccurs="unbounded"/>
4080         </choice>
4081     </complexType>
4082 </element>
4083
4084     <!-- example qualifier definition:
4085         <element name="some-qualifer" type="btp:qualifier-type"
4086 substitutionGroup="btp:qualifier"/>
4087     -->
4088
4089
4090
4091     <!-- Message set data types -->
4092
4093     <simpleType name="identifier">
4094         <restriction base="anyURI" />
4095     </simpleType>
4096
4097     <simpleType name="additional-information">
4098         <restriction base="string" />
4099     </simpleType>
4100
4101     <complexType name="address">
4102         <sequence>
4103             <element name="binding-name" type="string"/>
4104             <element name="binding-address" type="string"/>
4105             <element name="additional-information" type="btp:additional-
4106 information" minOccurs="0" />
4107         </sequence>
4108     </complexType>
4109
4110     <simpleType name="superior-type">
4111         <restriction base="string">
4112             <enumeration value="cohesion"/>
4113             <enumeration value="atom"/>
4114         </restriction>
4115     </simpleType>
4116
4117     <simpleType name="transaction-type">
4118         <restriction base="string">
4119             <enumeration value="cohesion"/>
4120             <enumeration value="atom"/>
4121         </restriction>
4122     </simpleType>
4123
4124
4125
4126     <!-- Compounding -->
4127
4128     <element name="messages">
4129         <complexType>

```

```

4130         <sequence>
4131             <element ref="btp:message" minOccurs="0"
4132 maxOccurs="unbounded" />
4133         </sequence>
4134     </complexType>
4135 </element>
4136
4137     <element name="related-group" substitutionGroup="btp:message">
4138         <complexType>
4139             <sequence>
4140                 <element ref="btp:message" minOccurs="0"
4141 maxOccurs="unbounded" />
4142             </sequence>
4143         </complexType>
4144     </element>
4145
4146
4147
4148     <!-- Message set -->
4149
4150     <element name="message" abstract="true" />
4151
4152     <element name="context" substitutionGroup="btp:message">
4153         <complexType>
4154             <sequence>
4155                 <element name="superior-address" type="btp:address"
4156 maxOccurs="unbounded" />
4157                 <element name="superior-identifier" type="btp:identifier" />
4158                 <element name="superior-type" type="btp:superior-type" />
4159                 <element ref="btp:qualifiers" minOccurs="0" />
4160                 <element name="reply-address" type="btp:address"
4161 minOccurs="0" />
4162             </sequence>
4163             <attribute name="id" type="ID" use="optional" />
4164         </complexType>
4165     </element>
4166
4167     <element name="context-reply" substitutionGroup="btp:message">
4168         <complexType>
4169             <sequence>
4170                 <element name="superior-identifier" type="btp:identifier" />
4171                 <element name="completion-status">
4172                     <simpleType>
4173                         <restriction base="string">
4174                             <enumeration value="completed" />
4175                             <enumeration value="incomplete" />
4176                             <enumeration value="related" />
4177                             <enumeration value="repudiated" />
4178                         </restriction>
4179                     </simpleType>
4180                 </element>
4181                 <element ref="btp:qualifiers" minOccurs="0" />
4182                 <element name="target-additional-information"
4183 type="btp:additional-information" minOccurs="0" />
4184             </sequence>

```

```

4185         <attribute name="id" type="ID" use="optional"/>
4186     </complexType>
4187 </element>
4188
4189     <element name="request-status" substitutionGroup="btp:message">
4190         <complexType>
4191             <sequence>
4192                 <element name="target-identifier" type="btp:identifier"/>
4193                 <element ref="btp:qualifiers" minOccurs="0"/>
4194                 <element name="target-additional-information"
4195 type="btp:additional-information" minOccurs="0"/>
4196                 <element name="reply-address" type="btp:address"
4197 minOccurs="0"/>
4198             </sequence>
4199             <attribute name="id" type="ID" use="optional"/>
4200         </complexType>
4201 </element>
4202
4203     <element name="status" substitutionGroup="btp:message">
4204         <complexType>
4205             <sequence>
4206                 <element name="responders-identifier"
4207 type="btp:identifier"/>
4208                 <element name="status-value">
4209                     <simpleType>
4210                         <restriction base="string">
4211                             <enumeration value="created"/>
4212                             <enumeration value="enrolling"/>
4213                             <enumeration value="active"/>
4214                             <enumeration value="resigning"/>
4215                             <enumeration value="resigned"/>
4216                             <enumeration value="preparing"/>
4217                             <enumeration value="prepared"/>
4218                             <enumeration value="confirming"/>
4219                             <enumeration value="confirmed"/>
4220                             <enumeration value="cancelling"/>
4221                             <enumeration value="cancelled"/>
4222                             <enumeration value="cancel-contradiction"/>
4223                             <enumeration value="confirm-contradiction"/>
4224                             <enumeration value="hazard"/>
4225                             <enumeration value="contradicted"/>
4226                             <enumeration value="unknown"/>
4227                             <enumeration value="inaccessible"/>
4228                         </restriction>
4229                     </simpleType>
4230                 </element>
4231                 <element ref="btp:qualifiers" minOccurs="0"/>
4232                 <element name="target-additional-information"
4233 type="btp:additional-information" minOccurs="0"/>
4234             </sequence>
4235             <attribute name="id" type="ID" use="optional"/>
4236         </complexType>
4237 </element>
4238
4239     <element name="fault" substitutionGroup="btp:message">

```

```

4240     <complexType>
4241         <sequence>
4242             <element name="superior-identifier" type="btp:identifier"
4243 minOccurs="0"/>
4244             <element name="inferior-identifier" type="btp:identifier"
4245 minOccurs="0"/>
4246             <element name="fault-type">
4247                 <simpleType>
4248                     <restriction base="string">
4249                         <enumeration value="communication-failure"/>
4250                         <enumeration value="duplicate-inferior"/>
4251                         <enumeration value="general"/>
4252                         <enumeration value="invalid-decider"/>
4253                         <enumeration value="invalid-inferior"/>
4254                         <enumeration value="invalid-superior"/>
4255                         <enumeration value="status-refused"/>
4256                         <enumeration value="invalid-terminator"/>
4257                         <enumeration value="unknown-parameter"/>
4258                         <enumeration value="unknown-transaction"/>
4259                         <enumeration value="unsupported-qualifier"/>
4260                         <enumeration value="wrong-state"/>
4261                         <enumeration value="redirect"/>
4262                     </restriction>
4263                 </simpleType>
4264             </element>
4265             <element name="fault-data" type="anyType" minOccurs="0"/>
4266             <element ref="btp:qualifiers" minOccurs="0"/>
4267             <element name="target-additional-information"
4268 type="btp:additional-information" minOccurs="0"/>
4269         </sequence>
4270         <attribute name="id" type="ID" use="optional"/>
4271     </complexType>
4272 </element>
4273
4274     <element name="enrol" substitutionGroup="btp:message">
4275         <complexType>
4276             <sequence>
4277                 <element name="superior-identifier" type="btp:identifier"/>
4278                 <element name="response-requested" type="boolean"
4279 minOccurs="0" default="false"/>
4280                 <element name="inferior-address" type="btp:address"
4281 minOccurs="1" maxOccurs="unbounded"/>
4282                 <element name="inferior-identifier" type="btp:identifier"/>
4283                 <element ref="btp:qualifiers" minOccurs="0"/>
4284                 <element name="target-additional-information"
4285 type="btp:additional-information" minOccurs="0"/>
4286                 <element name="reply-address" type="btp:address"
4287 minOccurs="0"/>
4288             </sequence>
4289             <attribute name="id" type="ID" use="optional"/>
4290         </complexType>
4291     </element>
4292
4293     <element name="enrolled" substitutionGroup="btp:message">
4294         <complexType>

```

```

4295         <sequence>
4296             <element name="inferior-identifier" type="btp:identifier"/>
4297             <element ref="btp:qualifiers" minOccurs="0"/>
4298             <element name="target-additional-information"
4299 type="btp:additional-information" minOccurs="0"/>
4300             <element name="sender-address" type="btp:address"
4301 minOccurs="0"/>
4302         </sequence>
4303         <attribute name="id" type="ID" use="optional"/>
4304     </complexType>
4305 </element>
4306
4307     <element name="resign" substitutionGroup="btp:message">
4308         <complexType>
4309             <sequence>
4310                 <element name="superior-identifier" type="btp:identifier"/>
4311                 <element name="inferior-identifier" type="btp:identifier"/>
4312                 <element name="response-requested" type="boolean"
4313 minOccurs="0" default="false"/>
4314                 <element ref="btp:qualifiers" minOccurs="0"/>
4315                 <element name="target-additional-information"
4316 type="btp:additional-information" minOccurs="0"/>
4317                 <element name="sender-address" type="btp:address"
4318 minOccurs="0"/>
4319             </sequence>
4320             <attribute name="id" type="ID" use="optional"/>
4321         </complexType>
4322 </element>
4323
4324     <element name="resigned" substitutionGroup="btp:message">
4325         <complexType>
4326             <sequence>
4327                 <element name="inferior-identifier" type="btp:identifier"/>
4328                 <element ref="btp:qualifiers" minOccurs="0"/>
4329                 <element name="target-additional-information"
4330 type="btp:additional-information" minOccurs="0"/>
4331                 <element name="sender-address" type="btp:address"
4332 minOccurs="0"/>
4333             </sequence>
4334             <attribute name="id" type="ID" use="optional"/>
4335         </complexType>
4336 </element>
4337
4338     <element name="prepare" substitutionGroup="btp:message">
4339         <complexType>
4340             <sequence>
4341                 <element name="inferior-identifier" type="btp:identifier"/>
4342                 <element ref="btp:qualifiers" minOccurs="0"/>
4343                 <element name="target-additional-information"
4344 type="btp:additional-information" minOccurs="0"/>
4345                 <element name="sender-address" type="btp:address"
4346 minOccurs="0"/>
4347             </sequence>
4348             <attribute name="id" type="ID" use="optional"/>
4349         </complexType>

```

```

4350     </element>
4351
4352     <element name="prepared" substitutionGroup="btp:message">
4353         <complexType>
4354             <sequence>
4355                 <element name="superior-identifier" type="btp:identifier"/>
4356                 <element name="inferior-identifier" type="btp:identifier"/>
4357                 <element name="default-is-cancel" type="boolean"/>
4358                 <element ref="btp:qualifiers" minOccurs="0"/>
4359                 <element name="target-additional-information"
4360 type="btp:additional-information" minOccurs="0"/>
4361                 <element name="sender-address" type="btp:address"
4362 minOccurs="0"/>
4363             </sequence>
4364             <attribute name="id" type="ID" use="optional"/>
4365         </complexType>
4366     </element>
4367
4368     <element name="confirm" substitutionGroup="btp:message">
4369         <complexType>
4370             <sequence>
4371                 <element name="inferior-identifier" type="btp:identifier"/>
4372                 <element ref="btp:qualifiers" minOccurs="0"/>
4373                 <element name="target-additional-information"
4374 type="btp:additional-information" minOccurs="0"/>
4375                 <element name="sender-address" type="btp:address"
4376 minOccurs="0"/>
4377             </sequence>
4378             <attribute name="id" type="ID" use="optional"/>
4379         </complexType>
4380     </element>
4381
4382     <element name="confirmed" substitutionGroup="btp:message">
4383         <complexType>
4384             <sequence>
4385                 <element name="superior-identifier" type="btp:identifier"/>
4386                 <element name="inferior-identifier" type="btp:identifier"/>
4387                 <element name="confirmed-received" type="boolean"/>
4388                 <element ref="btp:qualifiers" minOccurs="0"/>
4389                 <element name="target-additional-information"
4390 type="btp:additional-information" minOccurs="0"/>
4391                 <element name="sender-address" type="btp:address"
4392 minOccurs="0"/>
4393             </sequence>
4394             <attribute name="id" type="ID" use="optional"/>
4395         </complexType>
4396     </element>
4397
4398     <element name="cancel" substitutionGroup="btp:message">
4399         <complexType>
4400             <sequence>
4401                 <element name="inferior-identifier" type="btp:identifier"/>
4402                 <element ref="btp:qualifiers" minOccurs="0"/>
4403                 <element name="target-additional-information"
4404 type="btp:additional-information" minOccurs="0"/>

```

```

4405         <element name="sender-address" type="btp:address"
4406 minOccurs="0"/>
4407     </sequence>
4408     <attribute name="id" type="ID" use="optional"/>
4409 </complexType>
4410 </element>
4411
4412 <element name="cancelled" substitutionGroup="btp:message">
4413     <complexType>
4414         <sequence>
4415             <element name="superior-identifier" type="btp:identifier"/>
4416             <element name="inferior-identifier" type="btp:identifier"
4417 minOccurs="0"/>
4418             <element ref="btp:qualifiers" minOccurs="0"/>
4419             <element name="target-additional-information"
4420 type="btp:additional-information" minOccurs="0"/>
4421             <element name="sender-address" type="btp:address"
4422 minOccurs="0"/>
4423         </sequence>
4424         <attribute name="id" type="ID" use="optional"/>
4425     </complexType>
4426 </element>
4427
4428 <element name="confirm-one-phase" substitutionGroup="btp:message">
4429     <complexType>
4430         <sequence>
4431             <element name="inferior-identifier" type="btp:identifier"/>
4432             <element name="report-hazard" type="boolean"/>
4433             <element ref="btp:qualifiers" minOccurs="0"/>
4434             <element name="target-additional-information"
4435 type="btp:additional-information" minOccurs="0"/>
4436             <element name="sender-address" type="btp:address"
4437 minOccurs="0"/>
4438         </sequence>
4439         <attribute name="id" type="ID" use="optional"/>
4440     </complexType>
4441 </element>
4442
4443 <element name="hazard" substitutionGroup="btp:message">
4444     <complexType>
4445         <sequence>
4446             <element name="superior-identifier" type="btp:identifier"/>
4447             <element name="inferior-identifier" type="btp:identifier"/>
4448             <element name="level">
4449                 <simpleType>
4450                     <restriction base="string">
4451                         <enumeration value="mixed"/>
4452                         <enumeration value="possible"/>
4453                     </restriction>
4454                 </simpleType>
4455             </element>
4456             <element ref="btp:qualifiers" minOccurs="0"/>
4457             <element name="target-additional-information"
4458 type="btp:additional-information" minOccurs="0"/>

```

```

4459         <element name="sender-address" type="btp:address"
4460 minOccurs="0"/>
4461     </sequence>
4462     <attribute name="id" type="ID" use="optional"/>
4463 </complexType>
4464 </element>
4465
4466 <element name="contradiction" substitutionGroup="btp:message">
4467 <complexType>
4468 <sequence>
4469 <element name="inferior-identifier" type="btp:identifier"/>
4470 <element ref="btp:qualifiers" minOccurs="0"/>
4471 <element name="target-additional-information"
4472 type="btp:additional-information" minOccurs="0"/>
4473 <element name="sender-address" type="btp:address"
4474 minOccurs="0"/>
4475 </sequence>
4476 <attribute name="id" type="ID" use="optional"/>
4477 </complexType>
4478 </element>
4479
4480 <element name="superior-state" substitutionGroup="btp:message">
4481 <complexType>
4482 <sequence>
4483 <element name="inferior-identifier" type="btp:identifier"/>
4484 <element name="status">
4485 <simpleType>
4486 <restriction base="string">
4487 <enumeration value="active"/>
4488 <enumeration value="prepared-received"/>
4489 <enumeration value="inaccessible"/>
4490 <enumeration value="unknown"/>
4491 </restriction>
4492 </simpleType>
4493 </element>
4494 <element name="response-requested" type="boolean"
4495 minOccurs="0" default="false"/>
4496 <element ref="btp:qualifiers" minOccurs="0"/>
4497 <element name="target-additional-information"
4498 type="btp:additional-information" minOccurs="0"/>
4499 <element name="sender-address" type="btp:address"
4500 minOccurs="0"/>
4501 </sequence>
4502 <attribute name="id" type="ID" use="optional"/>
4503 </complexType>
4504 </element>
4505
4506 <element name="inferior-state" substitutionGroup="btp:message">
4507 <complexType>
4508 <sequence>
4509 <element name="superior-identifier" type="btp:identifier"/>
4510 <element name="inferior-identifier" type="btp:identifier"/>
4511 <element name="status">
4512 <simpleType>
4513 <restriction base="string">

```

```

4514         <enumeration value="active"/>
4515         <enumeration value="inaccessible"/>
4516         <enumeration value="unknown"/>
4517     </restriction>
4518 </simpleType>
4519 </element>
4520     <element name="response-requested" type="boolean"
4521 minOccurs="0" default="false"/>
4522     <element ref="btp:qualifiers" minOccurs="0"/>
4523     <element name="target-additional-information"
4524 type="btp:additional-information" minOccurs="0"/>
4525     <element name="sender-address" type="btp:address"
4526 minOccurs="0"/>
4527 </sequence>
4528     <attribute name="id" type="ID" use="optional"/>
4529 </complexType>
4530 </element>
4531
4532     <element name="redirect" substitutionGroup="btp:message">
4533     <complexType>
4534     <sequence>
4535         <element name="superior-identifier" type="btp:identifier"
4536 minOccurs="0"/>
4537         <element name="inferior-identifier" type="btp:identifier"
4538 />
4539         <element name="old-address" type="btp:address"
4540 maxOccurs="unbounded"/>
4541         <element name="new-address" type="btp:address"
4542 maxOccurs="unbounded"/>
4543         <element ref="btp:qualifiers" minOccurs="0"/>
4544         <element name="target-additional-information"
4545 type="btp:additional-information" minOccurs="0"/>
4546     </sequence>
4547     <attribute name="id" type="ID" use="optional"/>
4548 </complexType>
4549 </element>
4550
4551     <element name="begin" substitutionGroup="btp:message">
4552     <complexType>
4553     <sequence>
4554         <element name="transaction-type" type="btp:superior-type"/>
4555         <element ref="btp:qualifiers" minOccurs="0"/>
4556         <element name="target-additional-information"
4557 type="btp:additional-information" minOccurs="0"/>
4558         <element name="reply-address" type="btp:address"
4559 minOccurs="0"/>
4560     </sequence>
4561     <attribute name="id" type="ID" use="optional"/>
4562 </complexType>
4563 </element>
4564
4565     <element name="begun" substitutionGroup="btp:message">
4566     <complexType>
4567     <sequence>

```

```

4568         <element name="decider-address" type="btp:address"
4569 minOccurs="0" maxOccurs="unbounded"/>
4570         <element name="inferior-address" type="btp:address"
4571 minOccurs="0" maxOccurs="unbounded"/>
4572         <element name="transaction-identifier"
4573 type="btp:identifier" minOccurs="0"/>
4574         <element ref="btp:qualifiers" minOccurs="0"/>
4575         <element name="target-additional-information"
4576 type="btp:additional-information" minOccurs="0"/>
4577     </sequence>
4578     <attribute name="id" type="ID" use="optional"/>
4579 </complexType>
4580 </element>
4581
4582     <element name="prepare-inferiors" substitutionGroup="btp:message">
4583     <complexType>
4584     <sequence>
4585     <element name="transaction-identifier"
4586 type="btp:identifier"/>
4587     <element name="inferiors-list" minOccurs="0">
4588     <complexType>
4589     <sequence>
4590     <element name="inferior-identifier"
4591 type="btp:identifier" maxOccurs="unbounded"/>
4592     </sequence>
4593     </complexType>
4594     </element>
4595     <element ref="btp:qualifiers" minOccurs="0"/>
4596     <element name="target-additional-information"
4597 type="btp:additional-information" minOccurs="0"/>
4598     <element name="reply-address" type="btp:address"
4599 minOccurs="0"/>
4600     </sequence>
4601     <attribute name="id" type="ID" use="optional"/>
4602     </complexType>
4603 </element>
4604
4605     <element name="confirm-transaction" substitutionGroup="btp:message">
4606     <complexType>
4607     <sequence>
4608     <element name="transaction-identifier"
4609 type="btp:identifier"/>
4610     <element name="inferiors-list" minOccurs="0">
4611     <complexType>
4612     <sequence>
4613     <element name="inferior-identifier"
4614 type="btp:identifier" maxOccurs="unbounded"/>
4615     </sequence>
4616     </complexType>
4617     </element>
4618     <element name="report-hazard" type="boolean"/>
4619     <element ref="btp:qualifiers" minOccurs="0"/>
4620     <element name="target-additional-information"
4621 type="btp:additional-information" minOccurs="0"/>

```

```

4622         <element name="reply-address" type="btp:address"
4623 minOccurs="0"/>
4624     </sequence>
4625     <attribute name="id" type="ID" use="optional"/>
4626 </complexType>
4627 </element>
4628
4629 <element name="transaction-confirmed" substitutionGroup="btp:message">
4630 <complexType>
4631 <sequence>
4632 <element name="transaction-identifier"
4633 type="btp:identifier"/>
4634 <element ref="btp:qualifiers" minOccurs="0"/>
4635 <element name="target-additional-information"
4636 type="btp:additional-information" minOccurs="0"/>
4637 </sequence>
4638 <attribute name="id" type="ID" use="optional"/>
4639 </complexType>
4640 </element>
4641
4642 <element name="cancel-transaction" substitutionGroup="btp:message">
4643 <complexType>
4644 <sequence>
4645 <element name="transaction-identifier"
4646 type="btp:identifier"/>
4647 <element name="report-hazard" type="boolean"/>
4648 <element ref="btp:qualifiers" minOccurs="0"/>
4649 <element name="target-additional-information"
4650 type="btp:additional-information" minOccurs="0"/>
4651 <element name="reply-address" type="btp:address"
4652 minOccurs="0"/>
4653 </sequence>
4654 <attribute name="id" type="ID" use="optional"/>
4655 </complexType>
4656 </element>
4657
4658 <element name="cancel-inferiors" substitutionGroup="btp:message">
4659 <complexType>
4660 <sequence>
4661 <element name="transaction-identifier"
4662 type="btp:identifier" minOccurs="0"/>
4663 <element name="inferiors-list">
4664 <complexType>
4665 <sequence>
4666 <element name="inferior-identifier"
4667 type="btp:identifier" maxOccurs="unbounded"/>
4668 </sequence>
4669 </complexType>
4670 </element>
4671 <element ref="btp:qualifiers" minOccurs="0"/>
4672 <element name="target-additional-information"
4673 type="btp:additional-information" minOccurs="0"/>
4674 <element name="reply-address" type="btp:address"
4675 minOccurs="0"/>
4676 </sequence>

```

```

4677         <attribute name="id" type="ID" use="optional"/>
4678     </complexType>
4679 </element>
4680
4681     <element name="transaction-cancelled" substitutionGroup="btp:message">
4682         <complexType>
4683             <sequence>
4684                 <element name="transaction-identifier"
4685 type="btp:identifier"/>
4686                 <element ref="btp:qualifiers" minOccurs="0"/>
4687                 <element name="target-additional-information"
4688 type="btp:additional-information" minOccurs="0"/>
4689             </sequence>
4690             <attribute name="id" type="ID" use="optional"/>
4691         </complexType>
4692     </element>
4693
4694     <element name="request-inferior-statuses"
4695 substitutionGroup="btp:message">
4696         <complexType>
4697             <sequence>
4698                 <element name="target-identifier" type="btp:identifier"/>
4699                 <element name="inferiors-list" minOccurs="0">
4700                     <complexType>
4701                         <sequence>
4702                             <element name="inferior-identifier"
4703 type="btp:identifier" maxOccurs="unbounded"/>
4704                         </sequence>
4705                     </complexType>
4706                 </element>
4707                 <element ref="btp:qualifiers" minOccurs="0"/>
4708                 <element name="target-additional-information"
4709 type="btp:additional-information" minOccurs="0"/>
4710                 <element name="reply-address" type="btp:address"
4711 minOccurs="0"/>
4712             </sequence>
4713             <attribute name="id" type="ID" use="optional"/>
4714         </complexType>
4715     </element>
4716
4717     <element name="inferior-statuses" substitutionGroup="btp:message">
4718         <complexType>
4719             <sequence>
4720                 <element name="responders-identifier"
4721 type="btp:identifier"/>
4722                 <element name="status-list">
4723                     <complexType>
4724                         <sequence>
4725                             <element name="status-item" maxOccurs="unbounded">
4726                                 <complexType>
4727                                     <sequence>
4728                                         <element name="inferior-identifier"
4729 type="btp:identifier"/>
4730                                         <element name="status">
4731                                             <simpleType>

```

```

4732         <restriction base="string">
4733             <enumeration value="active"/>
4734             <enumeration value="resigned"/>
4735             <enumeration value="preparing"/>
4736             <enumeration value="prepared"/>
4737             <enumeration value="autonomously-
4738 confirmed"/>
4739             <enumeration value="autonomously-
4740 cancelled"/>
4741             <enumeration value="confirming"/>
4742             <enumeration value="confirmed"/>
4743             <enumeration value="cancelling"/>
4744             <enumeration value="cancelled"/>
4745             <enumeration value="cancel-
4746 contradiction"/>
4747             <enumeration value="confirm-
4748 contradiction"/>
4749             <enumeration value="hazard"/>
4750             <enumeration value="invalid"/>
4751         </restriction>
4752     </simpleType>
4753 </element>
4754     <element ref="btp:qualifiers" minOccurs="0"/>
4755 </sequence>
4756 </complexType>
4757 </element>
4758 </sequence>
4759 </complexType>
4760 </element>
4761     <element ref="btp:qualifiers" minOccurs="0"/>
4762     <element name="target-additional-information"
4763 type="btp:additional-information" minOccurs="0"/>
4764 </sequence>
4765     <attribute name="id" type="ID" use="optional"/>
4766 </complexType>
4767 </element>
4768
4769 </schema>

```

## 4770 10.5.2 XML schema for standard qualifiers

```

4771 <?xml version="1.0"?>
4772 <schema
4773     xmlns="http://www.w3.org/2001/XMLSchema"
4774     targetNamespace="urn:oasis:names:tc:BTP:1.0:qualifiers"
4775     xmlns:btpq="urn:oasis:names:tc:BTP:1.0:qualifiers"
4776     xmlns:btp="urn:oasis:names:tc:BTP:1.0:core"
4777     elementFormDefault="qualified">
4778
4779     <import namespace="urn:oasis:names:tc:BTP:1.0:core"/>
4780
4781     <element name="transaction-timelimit"
4782 substitutionGroup="btp:qualifier">
4783         <complexType>
4784             <complexContent>

```

```

4785         <extension base="btp:qualifier-type">
4786             <sequence>
4787                 <element name="timelimit"
4788 type="nonNegativeInteger"/>
4789             </sequence>
4790         </extension>
4791     </complexContent>
4792 </complexType>
4793 </element>
4794
4795     <element name="inferior-timeout" substitutionGroup="btp:qualifier">
4796         <complexType>
4797             <complexContent>
4798                 <extension base="btp:qualifier-type">
4799                     <sequence>
4800                         <element name="timelimit"
4801 type="nonNegativeInteger"/>
4802                         <element name="intended-decision">
4803                             <simpleType>
4804                                 <restriction base="string">
4805                                     <enumeration value="confirm"/>
4806                                     <enumeration value="cancel"/>
4807                                 </restriction>
4808                             </simpleType>
4809                         </element>
4810                     </sequence>
4811                 </extension>
4812             </complexContent>
4813         </complexType>
4814     </element>
4815
4816     <element name="minimum-inferior-timeout"
4817 substitutionGroup="btp:qualifier">
4818         <complexType>
4819             <complexContent>
4820                 <extension base="btp:qualifier-type">
4821                     <sequence>
4822 type="nonNegativeInteger"/>
4823                         </sequence>
4824                     </extension>
4825                 </complexContent>
4826             </complexType>
4827     </element>
4828
4829
4830     <element name="inferior-name" substitutionGroup="btp:qualifier">
4831         <complexType>
4832             <complexContent>
4833                 <extension base="btp:qualifier-type">
4834                     <sequence>
4835                         <element name="inferior-name" type="string"/>
4836                     </sequence>
4837                 </extension>
4838             </complexContent>
4839         </complexType>

```

```
4840     </element>
4841
4842 </schema>
4843
```

## 4844 11 Carrier Protocol Bindings

4845 The notion of bindings is introduced to act as the glue between the BTP messages and an  
4846 underlying transport. A binding specification must define various particulars of how the BTP  
4847 messages are carried and some aspects of how the related Application Messages are carried. This  
4848 document specifies two bindings: a SOAP binding and a SOAP + Attachments binding. However,  
4849 other bindings could be specified by the Oasis BTP technical committee or by a third party. For  
4850 example, in the future a binding might exist to put a BTP message directly on top of HTTP  
4851 without the use of SOAP, or a closed community could define their own binding. To ensure that  
4852 such specifications are complete, the Binding Proforma defines the information that must be  
4853 included in a binding specification.

4854 A registry of bindings, with links to the binding specifications is maintained on the OASIS  
4855 website, linked from the BTP page (<http://www.oasis-open.org/committees/business->  
4856 [transactions](http://www.oasis-open.org/committees/business-)). Any party may submit a binding specification and request its addition to this  
4857 registry. The presence of an entry in the registry does not, of itself, imply ratification or approval  
4858 by OASIS or the BTP Technical Committee.

### 4859 11.1 Carrier Protocol Binding Proforma

4860 A BTP carrier binding specification should provide the following information:

4861 **Binding name:** A name for the binding, as used in the “binding name” field of BTP Addresses  
4862 (and available for declaring the capabilities of an implementation). Binding specified in this  
4863 document, and future revisions of this document have binding names that are simple strings of  
4864 letters, numbers and hyphens (and, in particular, do not contain colons). Bindings specified  
4865 elsewhere shall have binding names that are URIs. Bindings specified in this document use  
4866 numbers to identify the version of the binding, not the version(s) of the Carrier Protocol.

4867 **Binding address format:** This section states the format of the “binding address” field of a BTP  
4868 Address for this binding. For many bindings, this will be a URL of some kind; for other bindings  
4869 it may be some other form

4870 **BTP message representation:** This section will define how BTP messages are represented. For  
4871 many bindings, the BTP message syntax will be as specified in the XML schema defined in this  
4872 document, and the normal string encoding of that XML will be used.

4873 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that are  
4874 not related to Application Messages are sent in either direction between Superior and Inferior.  
4875 (i.e. those messages sent directly between BTP Actors). This mapping need not be symmetric (i.e.  
4876 Superior to Inferior may differ to some degree to Inferior to Superior). The mapping may define  
4877 particular rules for particular BTP messages, or messages with particular parameter values (e.g.  
4878 the FAULT message with “fault-type” “CommunicationFailure” will typically not be sent as a  
4879 BTP message). The mapping states any constraints or requirements on which BTP may or must  
4880 be bundled together by compounding.

4881 **Mapping for BTP messages related to Application Messages:** This section will define how  
4882 BTP messages that are related to Application Messages are sent. A binding specification may  
4883 defer details of this to a particular application (e.g. a mapping specification could just say “the  
4884 CONTEXT may be carried as a parameter of an application invocation”). Alternatively, the  
4885 binding may specify a general method that represents the relationship between application and  
4886 BTP messages.

4887 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly but  
4888 are treated as implicit in carrier-protocol mechanisms, Application Messages or other BTP  
4889 messages. This may depend on particular parameter values of the BTP messages or the  
4890 Application Messages.

4891 **Faults:** The relationship between the fault and exception reporting mechanisms of the Carrier  
4892 Protocol and of BTP shall be defined. This may include definition of which Carrier Protocol  
4893 exceptions are equivalent to a FAULT/communication-failure message.

4894 **Relationship to other bindings:** Any relationship to other bindings is defined in this section. If  
4895 BTP Addresses with different bindings are be considered to match (for purposes of identifying  
4896 the Peer Superior/Inferior and redirection), this should be specified here.

4897 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are imposed  
4898 by use of this binding should be listed. This would include limitations on which messages can be  
4899 sent, which event sequences are supported and restrictions on parameter values. Such limitations  
4900 may reduce the usefulness of an implementation, but may be appropriate in certain environments.

4901 **Other:** Other features of the binding, especially any that will potentially affect interoperation  
4902 should be specified here. This may include restrictions or requirements on the use or support of  
4903 optional carrier parameters or mechanisms or use of standard or other qualifiers.

## 4904 **11.2 Bindings for request/response Carrier Protocols**

4905 BTP does not generally follow a request/response pattern. In particular, on the Outcome  
4906 Relationship either side may initiate a message – this is an essential part of the presume-abort  
4907 recovery paradigm although it is not limited to recovery cases. However, there are some BTP  
4908 messages, especially in the Control Relationship, that do have a request/response pattern. Many  
4909 (potential) Carrier Protocols (e.g. HTTP) do have a request/response pattern. The specification of  
4910 a binding specification to a request/response Carrier Protocol needs to state what rules apply –  
4911 which messages can be carried by requests, which by responses. The simplest rule is to send all  
4912 BTP messages on requests, and let the carrier responses travel back empty. This would be  
4913 inefficient in use of network resources, and possibly inconvenient when used for the BTP  
4914 request/response pairs.

4915 This section defines a set of rules that allow more efficient use of the carrier, while allowing the  
4916 initiator of a BTP request/response pair to ensure the BTP response is sent back on the carrier  
4917 response. These rules are specified in this section to enable binding specifications to reference  
4918 them, without requiring each binding specification to repeat similar information. These rules also  
4919 allow the receiver of a message between Superior and Inferior (in either direction) on a Carrier  
4920 Protocol request to send any reply message on the carrier response – the “sender-address” field is  
4921 implicitly considered to be that of the sender of the carrier request.

4922 A binding to a request/response carrier is not required to use these rules. It may define other rules.

### 4923 11.2.1 Request/response exploitation rules

4924 These rules allow implementations to use the request and response of the Carrier Protocol  
4925 efficiently, and, when a BTP request/response exchange occurs, to either treat the  
4926 request/response exchanges of the Carrier Protocol and of BTP independently, if both sides wish,  
4927 or allow either side to map them closely.

4928 Under these rules, an implementation sending a BTP request (i.e. a message, other than  
4929 CONTEXT, which has “reply-address” as a parameter in the abstract message definition), can  
4930 ensure that it and the reply map to a carrier request/response by supplying no value for the “reply-  
4931 address”. An implementation receiving such a request is required to send the BTP response on the  
4932 carrier response.

4933 Conversely, if an implementation does supply a “reply-address” value on the request, the receiver  
4934 has the option of sending the BTP response back on the carrier response, or sending it on a new  
4935 carrier request.

4936 Within the Outcome Relationship, apart from ENROL, there is no “reply-address”, and the parties  
4937 normally know each other’s “superior-address” and “inferior-address”. However, these messages  
4938 have a “sender-address”, which is used when the receiver does not have knowledge of the Peer. In  
4939 this case, the “sender-address” is treated as the “reply-address” of the other messages – if the field  
4940 is absent in a message on a carrier request, the “sender-address” is implicitly that of the request  
4941 sender. Any message for the Peer (including the three messages mentioned, FAULT but also any  
4942 other valid message in the Superior:Inferior relationship) may be sent on the carrier response.  
4943 Apart from this, both sides are permitted to treat the carrier request/response exchanges as  
4944 opportunities for sending messages to the appropriate destination.

4945 The rules:

4946 a) A BTP Actor **may** bundle one or more BTP messages and related groups that  
4947 have the same binding address for their target in a single btp:messages and  
4948 transmit this btp:messages element on a Carrier Protocol request. There is no  
4949 restriction on which combinations of messages and groups may be so bundled,  
4950 other than that they have the same binding address, and that this binding address  
4951 is usable as the destination of a Carrier Protocol request.

4952 b) A BTP Actor that has received a Carrier Protocol request to which it has not yet  
4953 responded, and which has one or more BTP messages and groups whose binding  
4954 address for the target matches the origin of the carrier request **may** bundle such  
4955 BTP messages in a single btp:messages element and transmit that on the Carrier  
4956 Protocol response.

4957 c) A BTP Actor that has received, on a Carrier Protocol request, one or more BTP  
4958 messages or related groups that require a BTP response and for which no “reply-  
4959 address” was supplied, **must** bundle the responding BTP message and groups in a  
4960 btp:messages element and transmit this element on the Carrier Protocol response  
4961 to the request that carried the BTP request.

- 4962 d) A BTP Actor that has received, on a Carrier Protocol request, one or more BTP  
4963 messages or related groups that, as abstract messages, have a “sender-address”  
4964 parameter but no “reply-address” was supplied and does not have knowledge of  
4965 the Peer address, **must** bundle the responding BTP message and groups in a  
4966 btp:messages element and transmit this element on the Carrier Protocol response  
4967 to the request that carried the BTP request. If the Actor does have knowledge of  
4968 the Peer address it **may** send one or messages for the Peer in the Carrier Protocol  
4969 response, regardless of whether the binding address of the Peer matches the  
4970 address of the Carrier Protocol requestor.
- 4971 e) Where only one message or group is to be sent, it shall be contained within a  
4972 btp:messages element, as a bundle of one element.
- 4973 f) A BTP Actor that receives a Carrier Protocol request carrying BTP messages that  
4974 do have a “reply-address”, or which initiate processing that produces BTP  
4975 messages whose target binding address matches the origin of the request, **may**  
4976 freely choose whether to use the Carrier Protocol response for the replies, or to  
4977 send back an “empty Carrier Protocol response”, and send the BTP replies in a  
4978 separately initiated Carrier Protocol request. The characteristics of an “empty  
4979 Carrier Protocol response” shall be stated in the particular binding specification.
- 4980 g) A BTP Actor that sends BTP messages on a Carrier Protocol request **must** be  
4981 able to accept returning BTP messages on the corresponding Carrier Protocol  
4982 response and, if the Actor has offered an address on which it will receive carrier  
4983 requests, must be able to accept “replying” BTP messages on a separate Carrier  
4984 Protocol request.

### 4985 11.3 SOAP Binding

4986 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)  
4987 specification, using the SOAP literal messaging style conventions. If no Application Message is  
4988 sent at the same time, the BTP messages are contained within the SOAP Body element. If  
4989 Application Messages are sent, the BTP messages are contained in the SOAP Header element.

4990 **Binding name:** soap-http-1

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

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

4996 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be  
4997 used.

4998 BTP messages sent on an HTTP request or HTTP response which is not carrying an Application  
4999 Message, the messages are contained in a single btp:messages element which is the immediate  
5000 child element of the SOAP Body element.

5001 An “empty Carrier Protocol response” sent after receiving an HTTP request containing a  
5002 btp:messages element in the SOAP Body when the implementation chooses just to reply at the  
5003 lower level (and when the request/response exploitation rules allow an empty Carrier Protocol  
5004 response), shall be any of:

5005 a) an empty HTTP response

5006 b) an HTTP response containing an empty SOAP Envelope

5007 c) an HTTP response containing a SOAP Envelope containing a single, empty  
5008 btp:messages element.

5009 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they have  
5010 no effect on the BTP sequence (other than indicating that the earlier sending did not cause a  
5011 communication failure.)

5012 If an Application Message is being sent at the same time, the mapping for related messages shall  
5013 be used, as if the BTP messages were related to the Application Message. (There is no ambiguity  
5014 in whether the BTP messages are related, because only CONTEXT and ENROL can be related to  
5015 an Application Message.)

5016 **Mapping for BTP messages related to Application Messages:** All BTP messages sent with an  
5017 Application Message, whether related to the Application Message or not, shall be sent in a single  
5018 btp:messages element in the SOAP Header. There shall be precisely one btp:messages element in  
5019 the SOAP Header.

5020 The “request/response exploitation” rules shall apply to the BTP messages carried in the SOAP  
5021 Header, as if they had been carried in a SOAP Body, unrelated to an Application Message, sent to  
5022 the same binding address.

5023 *Note – The Application Protocol itself (which is using the SOAP Body) may use the SOAP*  
5024 *RPC or document approach – this is determined by the application.*

5025 Only CONTEXT and ENROL messages are related (&) to Application Messages. If there is only  
5026 one CONTEXT or one ENROL message present in the SOAP Header, it is assumed to be related  
5027 to the whole of the Application Message in the SOAP Body. If there are multiple CONTEXT or  
5028 ENROL messages, any relation of these BTP messages shall be indicated by application specific  
5029 means.

5030 *Note 1 – An Application Protocol could use references to the ID values of the*  
5031 *BTP messages to indicate relation between BTP CONTEXT or ENROL*  
5032 *messages and the Application Message.*

5033 *Note 2 -- However indicated, what the relatedness means, or even whether it has*  
5034 *any significance at all, is a matter for the application.*

5035 **Implicit messages:** A SOAP FAULT, or other communication failure received in response to a  
5036 SOAP request that had a CONTEXT in the SOAP Header shall be treated as if a  
5037 CONTEXT\_REPLY/repudiated had been received. See also the discussion under “other” about  
5038 the SOAP mustUnderstand attribute.

5039 **Faults:** A SOAP FAULT or other communication failure shall be treated as  
5040 FAULT/communication-failure.

5041 **Relationship to other bindings:** A BTP Address for Superior or Inferior that has the binding  
5042 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-  
5043 http-1” if the binding address and additional information fields match.

5044 **Limitations on BTP use:** None

5045 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor  
5046 attribute. The SOAPAction HTTP header is left to be application specific when there are  
5047 Application Messages in the SOAP Body, as an already existing web service that is being  
5048 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP  
5049 header shall contain no value when the SOAP message carries only BTP messages in the SOAP  
5050 Body.

5051 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP  
5052 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to determine  
5053 whether any enrolments are necessary and replies with CONTEXT\_REPLY as appropriate. The  
5054 sender of the CONTEXT (and related Application Message) can use this to ensure that the  
5055 application work is performed as part of the Business Transaction, assuming the receiver’s SOAP  
5056 implementation supports the mustUnderstand attribute. If mustUnderstand if false, a receiver can  
5057 ignore the CONTEXT (if BTP is not supported there), and no CONTEXT\_REPLY will be  
5058 returned. It is a local option on the sender (Client) side whether the absence of a  
5059 CONTEXT\_REPLY is assumed to be equivalent to aCONTEXT\_REPLY/ok (and the Business  
5060 Transaction allowed to proceed to confirmation).

5061 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to  
5062 enforce these requirements.

### 5063 11.3.1 Example scenario using SOAP binding

5064 The example below shows an application request with CONTEXT message sent from  
5065 client.example.com (which includes the Superior) to services.example.com (Service).

```
5066 <soap:Envelope  
5067   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
5068   soap:encodingStyle="">  
5069   <soap:Header>  
5070     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core">  
5071       <btp:context superior-type="atom">  
5072         <btp:superior-address>  
5073           <btp:binding>soap-http-1</btp:binding>  
5074           <btp:binding-  
5075             address>http://client.example.com/soaphandler</btp:binding-  
5076             address>  
5077           <btp:additional-information>btpengine</btp:additional-  
5078             information>  
5079         </btp:superior-address>  
5080       </btp:messages>
```

```

5081         <btp:superior-
5082 identifier>http://example.com/1001</btp:superior-identifier>
5083         <btp:qualifiers>
5084             <btpq:transaction-timelimit
5085 xmlns:btpq="urn:oasis:names:tc:BTP:1.0:qualifiers"><btpq:timelimit
5086 >1800</btpq:timelimit></btpq:transaction-timelimit>
5087             </btp:qualifiers>
5088         </btp:context>
5089     </btp:messages>
5090 </soap:Header>
5091 <soap:Body>
5092     <ns1:orderGoods
5093 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
5094         <custID>ABC8329045</custID>
5095         <itemID>224352</itemID>
5096         <quantity>5</quantity>
5097     </ns1:orderGoods>
5098 </soap:Body>
5099 </soap:Envelope>
5100

```

5101 The example below shows CONTEXT\_REPLY and a related ENROL message sent from  
5102 services.example.com to client.example.com, in reply to the previous message. There is no  
5103 application response, so the BTP messages are in the SOAP Body. The ENROL message does not  
5104 contain the target-additional-information, since the grouping rules for CONTEXT\_REPLY &  
5105 ENROL omit the "target-address" (the receiver of this example remembers the superior address  
5106 from the original CONTEXT)

```

5107 <soap:Envelope
5108     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
5109     soap:encodingStyle="">
5110 <soap:Header>
5111 </soap:Header>
5112 <soap:Body>
5113     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core">
5114         <btp:related-group>
5115             <btp:context-reply>
5116                 <btp:target-additional-information>btpeengine</btp:target-
5117 additional-information>
5118                 <btp:superior-
5119 identifier>http://example.com/1001</btp:superior-identifier>
5120                 <completion-status>related</completion-status>
5121                 </btp:context-reply>
5122                 <btp:enrol response-requested="false">
5123                     <btp:target-additional-
5124 information>btpeengine</btp:target-additional-information>
5125                     <btp:superior-
5126 identifier>http://example.com/1001</btp:superior-identifier>
5127                     <btp:inferior-address>
5128                         <btp:binding>soap-http-1</btp:binding>
5129                         <btp:binding-address>
5130                             http://services.example.com/soaphandler
5131                         </btp:binding-address>
5132                     </btp:inferior-address>

```

```
5133     <btpr:inferior-identifier>
5134         http://example.com/AAAB
5135     </btpr:inferior-identifier>
5136     </btpr:enrol>
5137     </btpr:related-group>
5138     </btpr:messages>
5139     </soap:Body>
5140 </soap:Envelope>
5141
```

#### 5142 **11.4 SOAP + Attachments Binding**

5143 This binding describes how BTP messages will be carried using SOAP as in the [SOAP Messages](#)  
5144 [with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-http-1. The two  
5145 bindings only differ when Application Messages are sent.

5146 **Binding name:** soap-attachments-http-1

5147 **Binding address format:** as for soap-http-1

5148 **BTP message representation:** As for soap-http-1

5149 **Mapping for BTP messages (unrelated):** As for “soap-http-1”, except the SOAP Envelope  
5150 containing the SOAP Body containing the BTP messages shall be in a MIME body part, as  
5151 specified in [SOAP Messages with Attachments](#) specification. If an Application Message is being  
5152 sent at the same time, the mapping for related messages for this binding shall be used, as if the  
5153 BTP messages were related to the Application Message(s).

5154 **Mapping for BTP messages related to Application Messages:** MIME packaging shall be used.  
5155 One of the MIME multipart/related parts shall contain a SOAP Envelope, whose SOAP Headers  
5156 element shall contain precisely one btpr:messages element, containing any BTP messages. Any  
5157 BTP CONTEXT in the btpr:messages is considered to be related to the Application Message(s) in  
5158 the SOAP Body, and to also any of the MIME parts referenced from the SOAP Body (using the  
5159 “href” attribute).

5160 **Implicit messages:** As for soap-http-1.

5161 **Faults:** As for soap-http-1.

5162 **Relationship to other bindings:** A BTP Address for Superior or Inferior that has the binding  
5163 string “soap-http-1” is considered to match one that has the binding string “soap-attachements-  
5164 http-1” if the binding address and additional information fields match.

5165 **Limitations on BTP use:** None

5166 **Other:** As for soap-http-1

## 5167 11.4.1 Example using SOAP + Attachments binding

```
5168 Content-Type: Multipart/Related; boundary=MIME_boundary;
5169 type=text/xml;
5170     start="someID"
5171 --MIME_boundary
5172 Content-Type: text/xml; charset=UTF-8
5173 Content-ID: someID
5174 <?xml version='1.0' ?>
5175 <soap:Envelope
5176     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
5177     soap:encodingStyle=" " >
5178   <soap:Header>
5179     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core">
5180       <btp:context superior-type="atom">
5181         <btp:superior-address>
5182           <btp:binding>soap-http-1</btp:binding>
5183           <btp:binding-address>
5184             http://client.example.com/soaphandler
5185           </btp:binding-address>
5186           </btp:superior-address>
5187           <btp:superior-
5188 identifier>http://example.com/1001</btp:superior-identifier>
5189         </btp:context>
5190       </btp:messages>
5191     </soap:Header>
5192     <soap:Body>
5193       <orderGoods href="cid:anotherID"/>
5194     </soap:Body>
5195   </soap:Envelope>
5196 --MIME_boundary
5197 Content-Type: text/xml
5198 Content-ID: anotherID
5199   <ns1:orderGoods
5200 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
5201     <custID>ABC8329045</custID>
5202     <itemID>224352</itemID>
5203     <quantity>5</quantity>
5204   </ns1:orderGoods>
5205
5206 --MIME_boundary--
```

## 5207 12 Conformance

5208 A BTP implementation need not implement all aspects of the protocol to be useful. The level of  
5209 conformance of an implementation is defined by which roles it can support using the specified  
5210 messages and Carrier Protocol bindings for interoperation with other implementations.

5211 An implementation may implement some roles and relationships in accordance with this  
5212 specification, while providing the (approximate) functionality of other roles in some other  
5213 manner. (For example, an implementation might provide an equivalent of the Control  
5214 Relationships using a language-specific API, but support roles involved in the Outcome  
5215 Relationships using standard BTP messages.) Such an implementation is conformant in respect of  
5216 the roles it does implement in accordance with this specification.

5217 An implementation can state which aspects of the BTP specification it conforms to in terms of  
 5218 which Roles it supports. Since most Roles cannot usefully be supported in isolation, the following  
 5219 Role Groups can be used to describe implementation capabilities:.

<b>Role Group</b>	<b>Roles</b>
<b>Initiator/Terminator</b>	Initiator Terminator
<b>Cohesive Hub</b>	Factory Composer (as Decider and Superior) Coordinator (as Decider and Superior) Sub-composer Sub-coordinator
<b>Atomic Hub</b>	Factory Coordinator Sub-coordinator
<b>Cohesive Superior</b>	Composer (as Superior only) Sub-Composer Coordinator (as Superior only) Sub-coordinator
<b>Atomic Superior</b>	Coordinator (as Superior only)) Sub-coordinator
<b>Participant</b>	Inferior Enroller

5220  
 5221 The Role Groups occupy different positions within a Business Transaction Tree and thus require  
 5222 presence of implementations supporting other Role Groups:

5223 Initiator/Terminator uses Control Relationship to Atomic Hub or Cohesive Hub to initiate  
 5224 and control Atoms or Cohesions. Initiator/Terminator would typically be a library linked  
 5225 with application software.

5226 Atomic Hub and Cohesive Hub would often be standalone servers.

5227 Cohesive Superior and Atomic Superior would provide the equivalent of  
 5228 Initiator/Terminator functionality by internal or proprietary means.

5229 Cohesive Hubs, Atomic Hubs, Cohesive Superior and Atomic Superior use Outcome  
 5230 Relationships to Participants and to each other.

5231 Participants will establish Outcome Relationships to implementations of any of the other  
 5232 Role Groups except Initiator/Terminator. A Participant “covers” a resource or application

5233 work of some kind. It should be noted that a Participant is unaffected by whether it is  
5234 enrolled in an Atom or Cohesion – it gets only a single outcome.

5235 An implementation may support one or more Role Groups. The following combinations are  
5236 defined as commonly expected conformance profiles, although other combinations or selections  
5237 are equally possible.

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

5238

5239 BTP has several features, such as optional parameters, that allow alternative implementation  
5240 architectures. Implementations should pay particular attention to avoid assuming their peers have  
5241 made the same implementation options as they have (e.g. an implementation that always sends  
5242 ENROL with the same inferior address and with the “reply-address” absent (because the Inferior  
5243 in all transactions are dealt with by the same addressable entity), must not assume that the same is  
5244 true of received ENROLs)

5245

<b>Actor</b>	An entity that executes procedures, a software agent. (See also BTP Actor)
<b>Address</b>	An identifier for an endpoint.
<b>Application</b>	<p>An Actor, which uses the Business Transaction Protocol (in the context of this specification).</p> <p>Also, a group of such Actors, which may be distributed, that perform a common purpose.</p> <p>(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.)</p>
<b>Application Element</b>	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.
<b>Application Message</b>	A message produced by an Application Element and consumed by an Application Element.
<b>Application Operation</b>	An operation, which is started when an Application Message arrives.
<b>Appropriate</b>	In accordance with a pertinent contract or specification.
<b>Atom</b>	A set of participants, which are the direct inferiors of a BTP 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.)

<b>Atomic Business Transaction</b>	A complete Business Transaction that follows the atom rules for every BTP 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.)
<b>Become Prepared</b>	Ensure that of a set of procedures is capable of being successfully instructed to Cancel or to Confirm.
<b>BTP Actor</b>	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.
<b>BTP Element</b>	A BTP Actor that supports an Application Element (or elements) but is not itself concerned with Application Messages or semantics.
<b>(Business) Application Protocol</b>	The messages, their meanings and their permitted sequences used to effect a change in the state of a business relationship.
<b>(Business) Application System</b>	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.
<b>Business relationship</b>	A <i>business relationship</i> is any distributed state held by the parties, which is subject to contractual constraints agreed by those parties.
<b>Business Transaction Protocol (BTP)</b>	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.

<b>BTP Address</b>	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.
<b>Business Transaction</b>	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 <i>Business Transaction</i> is subject to, and a part of, a <i>business relationship</i> . (BTP assumes that the parties involved in a <i>Business Transaction</i> 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.)
<b>Cancel</b>	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.
<b>Carrier Protocol</b>	A protocol, which defines how the transmission of BTP messages occur.
<b>Client</b>	An Actor, which sends Application Messages to services.
<b>Cohesion</b>	A set of participants, which are the direct inferiors of a BTP 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.

<b>Cohesive Business Transaction</b>	A complete Business Transaction for which at least one BTP Node over space and time follows the cohesion rules. The other BTP Nodes in the Transaction Tree of a Cohesive Business Transaction may follow either the cohesion rules or the atom rules.
<b>Confirm</b>	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.
<b>Contract</b>	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.
<b>Control Relationship</b>	The Application Element:BTP Element relationships that create the nodes of the Transaction Tree (Initiator:Factory) and drive the completion (Terminator:Decider).
<b>Coordinator</b>	A BTP Actor, which is the top BTP 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.
<b>Counter-effect</b>	An appropriate effect intended to counteract a Provisional Effect.
<b>Decider</b>	The top BTP 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.
<b>Delivery Parameter</b>	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.
<b>Endpoint</b>	A sender or receiver.
<b>Enroller</b>	The BTP Actor Role that informs a superior of the existence of an inferior.

<b>Factory</b>	The BTP Actor Role that creates transaction contexts and deciders.
<b>Final Effect</b>	An appropriate effect intended to complete and finalise a Provisional Effect
<b>Inferior</b>	The 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.
<b>Inferior-Address</b>	The address used to communicate with an Actor playing the Role of an Inferior.
<b>Inferior-identifier</b>	A globally unambiguous identification of a particular Inferior within a single transaction (represented as an URI or equivalent).
<b>Initiator</b>	The BTP Actor Role (an Application Element) that starts a transaction.
<b>Intermediate</b>	A BTP Node that is a sub-composer or a sub-coordinator. An alternative term to interposed.
<b>Interposed</b>	A BTP Node that is a sub-composer or a sub-coordinator. An alternative term to intermediate.
<b>Message</b>	A datum, which is produced and then consumed.
<b>Node</b>	BTP Node, Business Transaction Tree Node, Transaction Tree Node: A logical entity that is associated with a single transaction. A BTP Node is a composer, a coordinator, a sub-coordinator, a sub-composer, or a participant.  Network Node: A computer system or program that hosts one or more BTP Actors (and thus, often, BTP Nodes)
<b>Operation</b>	A procedure, which is started by a receiver when a message arrives at it.
<b>Outcome</b>	A decision to either Cancel or Confirm.
<b>Outcome Relationship</b>	The Superior:Inferior relationship (i.e. between BTP Actors within the Transaction Tree) and the Enroller:Superior relationship used in establishing it.

<b>Participant</b>	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.
<b>Payload Parameter</b>	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.
<b>Peer</b>	The other party in a two-party relationship, as in Superior to Inferior, or Sender to Receiver.
<b>Provisional Effect</b>	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.
<b>Receiver</b>	The consumer of a message.
<b>Responders-identifier</b>	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.
<b>Role</b>	The participation of a software agent in a particular relationship in a particular Business Transaction. The software agent performing a Role is termed an <b>Actor</b> .
<b>Sender</b>	The producer of a message.
<b>Service</b>	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.
<b>Status Requestor</b>	The BTP Actor Role that requests the status of another BTP Actor.

<b>Sub-composer</b>	An Actor, which is not the top BTP 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.
<b>Sub-coordinator</b>	An Actor, which is not the top BTP 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 BTP Nodes can send BTP messages.
<b>Superior</b>	<p>The BTP Role that will accept enrolments of Inferiors and subsequently inform the Inferior of the Outcome applicable to it.</p> <p>A Superior will be one of Composer, Coordinator, Sub-composer, or Sub-coordinator.</p> <p>A Superior is considered to be a Superior even if it currently has no enrolled Inferiors.</p>
<b>Superior-address</b>	The set of BTP addresses used to communicate with an Actor playing the Role of a Superior.
<b>Superior-identifier</b>	A globally unambiguous identifier of a particular Superior within a particular transaction (represented as an URI or equivalent).
<b>Target-identifier</b>	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.
<b>Terminator</b>	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.

<b>Transaction</b>	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).
<b>Transaction Tree</b>	A pattern of BTP Nodes that provides the coordination of a distributed application transaction. There is single top BTP Node (a Decider) that interacts with the initiating application (which is a part of a distributed application). The Decider BTP Node has one, or more Outcome Relationships with other BTP Nodes (sub-composer, sub-coordinator, or participant BTP Nodes). Any intermediate BTP 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.
<b>Transaction-identifier</b>	A globally unambiguous identifier for a particular a Decider(represented as an URI or equivalent). A Decider is the top BTP 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.
<b>Transmission</b>	The passage of a message from a sender to a receiver.

5247

5248

## 5248 Part 4. Annexes

### 5249 **13 Informational annex A Node State Information** 5250 **Serialisation**

5251 This Annex provides a simple, but standardised format for the serialised essential state  
5252 information of a BTP Node. It does not specify the events that would cause serialisation to take  
5253 place, nor does it specify how this serialisation format is extracted from a BTP Node and  
5254 transferred elsewhere. The format is specified in abstract form and as an XML Schema.

#### 5255 **13.1 NODE STATE INFORMATION**

##### 5256 **13.1.1 Abstract Format for Node State Information**

5257 The node state information represents the BTP state information for a single BTP Node in some  
5258 Transaction Tree. It contains information for a single transaction that was extant at the BTP Node  
5259 at the time the serialisation was performed.

Parameter	Sub-Parameter	Type
date and time		Date and Time
Role		composer/coordinator/sub-composer/sub-coordinator/participant
own information	transaction type	cohesion/atom
	own-identifier	Identifier
	own-address	Set of BTP Addresses
information as inferior	transaction type	cohesion/atom
	inferior-state-identification	State identifier
	superior's identifier	Identifier
	superior's address	Set of BTP Addresses
	Qualifiers	List of qualifiers
Set of information as superior	superior-state-identification	State identifier
	inferior's identifier	Identifier
	inferior's address	Set of BTP Addresses
	Qualifiers	List of qualifiers

5260

5261 **date and time** the date and time that this node state information was generated to an  
5262 agreed resolution and accuracy. The presence of this information is optional.

5263 **role** the type of the BTP Node. Its value is one of composer / coordinator / sub-composer /  
5264 sub-coordinator / participant.

5265 **own information** identification information for this BTP Node. This information is  
5266 required. It consists of the following information:

5267 **transaction type** the type of this part of the transaction propagated to inferiors. Its  
5268 value is one of cohesion or atom.

5269 **own identifier** identifies this BTP Node. This may be the superior identifier from the  
5270 CONTEXT for the node and/or the inferior identifier on the ENROL for the node.  
5271 This shall be globally unambiguous.

5272 **own address** the address at which this BTP Node may be accessible. This can be a  
5273 set of alternative addresses.

5274 **information as inferior** information relevant to the BTP Node's Role as an inferior.  
5275 Should be present, once only, if the BTP Node is a sub-composer or a sub-coordinator  
5276 or a participant, otherwise absent. It includes information about the superior of this  
5277 BTP Node and consists of the following information:

5278 **transaction type** the type of this part of the transaction that applies to the BTP Node  
5279 acting as an inferior as indicated in the CONTEXT for the BTP Node. Its value is one  
5280 of cohesion or atom.

5281 **inferior-state-identification** identifies the state of the inferior state machine at this  
5282 BTP Node. This is represented as a small letter followed by a number, which  
5283 designates the inferior state. Refer to the section on 'State Tables' and in particular  
5284 Tables 6 and 11 - 14.

5285 **superior's identifier** identifies the Superior of this BTP Node. This shall be globally  
5286 unambiguous.

5287 **superior's address** the address to which ENROL and other messages from this  
5288 enrolled Inferior were sent. This can be a set of alternative addresses.

5289 **qualifiers** list of the qualifiers and their values in force for this node as an inferior.

5290 **set of information as superior** information relevant to the node's Role as superior.  
5291 Should be present, if the BTP Node is a composer, coordinator, sub-composer, or a  
5292 sub-coordinator, and shall be absent if the BTP Node is a participant. It may be  
5293 present multiple times, once for each inferior that this BTP Node has a relationship  
5294 with. It includes information about an inferior of this node and consists of the  
5295 following information:

5296 **superior-state-identification** identifies the state of the superior state machine for this  
5297 particular inferior. This is represented as a capital letter followed by a number, which  
5298 designates the superior state. Refer to the section on 'State Tables' and in particular  
5299 Tables 7 and 7 - 10.

5300 **inferior's identifier** identifies an Inferior of this BTP Node. This shall be globally  
5301 unambiguous.

5302 **inferior's address** the address to which PREPARE, CONFIRM, CANCEL and  
5303 SUPERIOR\_STATE messages for this Inferior have been or are to be sent. This can  
5304 be a set of alternative addresses.

5305 **qualifiers** list of the qualifiers and their values in force for this BTP Node as superior  
5306 to this inferior.

### 5307 13.1.2 Informal XML for Node State Information

```
5308 <btpst:node-information>
5309
5310   <btpst:date-time>2002-05-31T13:20:00.000-05:00</btpst:date-time>?
5311
5312   <btpst:role>composer|coordinator|sub-composer|sub-
5313 coordinator|participant</btpst:role>?
5314
5315   <btpst:own-information>
5316     <btpst:trx-type>cohesion|atom</btpst:trx-type>
5317     <btpst:own-identifier>...URI...</btpst:own-identifier>
5318     <btpst:own-address> +
5319       <btp:binding-name>...carrier binding name...</btp:binding-name>
5320       <btp:binding-address>...carrier specific address...</btp:binding-
5321 address>
5322       <btp:additional-information>...optional additional addressing
5323 information...</btp:additional-information> ?
5324     </btpst:own-address>
5325   </btpst:own-information>
5326
5327   <btpst:information-as-inferior> ?
5328     <btpst:trx-type>cohesion|atom</btpst:trx-type>
5329     <btpst:I_state>.. statename from inferior state table e.g.
5330 d1..</btpst:I_state>
5331     <btpst:superiors-identifier>...URI...</btpst:superiors-identifier>
5332     <btpst:superiors-address> +
5333       <btp:binding-name>...carrier binding name...</btp:binding-name>
5334       <btp:binding-address>...carrier specific address...</btp:binding-
5335 address>
5336       <btp:additional-information>...optional additional addressing
5337 information...</btp:additional-information> ?
5338     </btpst:superiors-address>
5339     <btp:qualifiers> ...qualifiers... </btp:qualifiers> ?
5340   </btpst:information-as-inferior>
5341
5342   <btpst:information-as-superior> +
5343     <btpst:S_state>.. statename from superior state table e.g.
5344 D1..</btpst:S_state>
5345     <btpst:inferiors-identifier>...URI...</btpst:inferiors-identifier>
5346     <btpst:inferiors-address> +
5347       <btp:binding-name>...carrier binding name...</btp:binding-name>
5348       <btp:binding-address>...carrier specific address...</btp:binding-
5349 address>
5350       <btp:additional-information>...optional additional addressing
5351 information...</btp:additional-information> ?
5352     </btpst:inferiors-address>
5353     <btp:qualifiers> ...qualifiers... </btp:qualifiers> ?
5354   </btpst:information-as-superior>
5355
5356 </btpst:node-information>
```

### 5357 13.1.3 XML schema for Node State Information

```
5358 <?xml version="1.0" encoding="UTF-8"?>
5359 <schema
5360     xmlns="http://www.w3.org/2001/XMLSchema"
5361     targetNamespace="urn:oasis:names:tc:BTP:1.0:node_state_information"
5362     xmlns:btst="urn:oasis:names:tc:BTP:1.0:node_state_information"
5363     xmlns:btpq="urn:oasis:names:tc:BTP:1.0:qualifiers"
5364     xmlns:btp="urn:oasis:names:tc:BTP:1.0:core"
5365     elementFormDefault="qualified">
5366
5367 <import namespace="urn:oasis:names:tc:BTP:1.0:qualifiers"/>
5368 <import namespace="urn:oasis:names:tc:BTP:1.0:core"/>
5369
5370
5371 <!-- Main node - information element definition -->
5372
5373 <element name="node-information">
5374     <complexType>
5375         <sequence>
5376
5377         <element name="date-time" type="dateTime" minOccurs="0"/>
5378
5379         <element name="role" minOccurs="0">
5380             <simpleType>
5381                 <restriction base="string">
5382                     <enumeration value="composer"/>
5383                     <enumeration value="coordinator"/>
5384                     <enumeration value="sub-Composer"/>
5385                     <enumeration value="sub-Coordinator"/>
5386                     <enumeration value="participant"/>
5387                 </restriction>
5388             </simpleType>
5389         </element>
5390
5391         <element name="own-information">
5392             <complexType>
5393                 <sequence>
5394                     <element ref="btst:trx-type"/>
5395                     <element name="own-identifier" type="btp:identifier"/>
5396                     <element name="own-address" type="btp:address" minOccurs="1"
5397 maxOccurs="unbounded"/>
5398                 </sequence>
5399             </complexType>
5400         </element>
5401
5402         <element name="information-as-inferior" minOccurs="0">
5403             <complexType>
5404                 <sequence>
5405                     <element ref="btst:trx-type"/>
5406                     <element name="I_state">
5407                         <simpleType>
5408                             <restriction base="string">
5409                                 <pattern value="[a-z][0-9]"/>
5410                             </restriction>

```

```

5411     </simpleType>
5412 </element>
5413 <element name="superiors-identifier" type="btp:identifier"/>
5414 <element name="superiors-address" type="btp:address" minOccurs="1"
5415 maxOccurs="unbounded"/>
5416 <element ref="btp:qualifiers" minOccurs="0"/>
5417 </sequence>
5418 </complexType>
5419 </element>
5420
5421 <element name="information-as-superior" minOccurs="0"
5422 maxOccurs="unbounded">
5423 <complexType>
5424 <sequence>
5425 <element name="S_state">
5426 <simpleType>
5427 <restriction base="string">
5428 <pattern value="[A-Z][0-9]"/>
5429 </restriction>
5430 </simpleType>
5431 </element>
5432 <element name="inferiors-identifier" type="btp:identifier"/>
5433 <element name="inferiors-address" type="btp:address" minOccurs="1"
5434 maxOccurs="unbounded"/>
5435 <element ref="btp:qualifiers" minOccurs="0"/>
5436 </sequence>
5437 </complexType>
5438 </element>
5439
5440 </sequence>
5441 </complexType>
5442 </element>
5443
5444 <!-- Common elements and datatypes -->
5445
5446 <element name="trx-type">
5447 <simpleType>
5448 <restriction base="string">
5449 <enumeration value="atom"/>
5450 <enumeration value="cohesion"/>
5451 </restriction>
5452 </simpleType>
5453 </element>
5454
5455 </schema>
5456

```