

# 2 Business Transaction Protocol

## 3 An OASIS Committee Specification

4  
5  
6 ***CURRENT STATUS : internal committee draft***

7  
8 Version 1.0 [0.9.1.1]

9 DD Mmm 2001 [27 January 2002 19:40]

10  
11

<i>Working draft 0.1 (pre-London)</i>	14 June 2001
<i>Working draft 0.2 (London)</i>	18 June 2001
<i>Working draft 0.3a (circulated)</i>	12 July 2001
<i>Working draft 0.3c (circulated)</i>	20 July 2001
<i>Working draft 0.4 (circulated; incorporates PRF material)</i>	25 July 2001
<i>Working draft 0.6 (State tables)</i>	31 August 2001
<i>Working Draft 0.9</i>	24 October 2001
<i>Working Draft 0.9.0.1 – minor editorials issues applied</i>	16 November 2001
<i>Working Draft 0.9.0.2 – issue resolutions balloting to 10 Dec 2001</i>	4 December 2001
<i>Working Draft 0.9.0.3 – possible solution to msging issues</i>	11 December 2001
<i>Working Draft 0.9.0.4 – issue 79 solution, revise msging issues</i>	12 January 2002
<i>Working Draft 0.9.1 – includes all issues agreed 16 Jan 2002, and 82 (deferred)</i>	18 January 2002
<i>Working Draft 0.9.1.1 – format changes and proposed soln 77,78, 17.</i>	27 January 2002

12  
13 [\*Change marks relative to 0.9.1\*](#)

## 14 Copyright and related notices

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

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

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

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

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

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

54

54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99

## Acknowledgements

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

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

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

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

Additional contributions to its writing were made by

Sanjay Dalal ([sanjay.dalal@bea.com](mailto:sanjay.dalal@bea.com))  
Mark Little ([mark\\_little@hp.com](mailto:mark_little@hp.com))

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

*In memory of Ed Felt*

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

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

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

## 99 **Typographical and Linguistic Conventions and Style**

100

101

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

102

103

104

105

Cancel

106

Participant

107

Application Message

108

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

109

110

### **Coordinator**

111

112

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

113

114

115

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

116

117

BEGIN

118

CONTEXT

119

RESIGN

120

121

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

122

123

BEGIN/atom

124

125

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

126

127

BEGIN/atom & CONTEXT

128

129

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

130

131

ENROL + VOTE

132

133

XML schemata and instances are given in Courier:

134

135

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

136

137

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

138

139

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

140

141

142

143

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

144

145

146

147  
148  
149  
150  
151

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

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

197	ENROLLED .....	40
198	RESIGN .....	41
199	RESIGNED .....	42
200	PREPARE .....	43
201	PREPARED .....	43
202	CONFIRM .....	45
203	CONFIRMED .....	46
204	CANCEL .....	47
205	CANCELLED .....	48
206	CONFIRM_ONE_PHASE .....	49
207	HAZARD .....	50
208	CONTRADICTION .....	51
209	SUPERIOR_STATE .....	51
210	INFERIOR_STATE .....	53
211	REDIRECT .....	55
212	Messages used in control relationships .....	56
213	BEGIN .....	56
214	BEGUN .....	57
215	PREPARE_INFERIORS .....	58
216	CONFIRM_TRANSACTION .....	59
217	TRANSACTION_CONFIRMED .....	61
218	CANCEL_TRANSACTION .....	62
219	CANCEL_INFERIORS .....	63
220	TRANSACTION_CANCELLED .....	64
221	REQUEST_INFERIOR_STATUSES .....	65
222	INFERIOR_STATUSES .....	66
223	Groups – combinations of related messages .....	68
224	CONTEXT & application message .....	69
225	CONTEXT_REPLY & ENROL .....	69
226	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED .....	70
227	CONTEXT_REPLY & ENROL & application message (& PREPARED) .....	71
228	BEGUN & CONTEXT .....	72
229	BEGIN & CONTEXT .....	72
230	Standard qualifiers .....	72
231	Transaction timelimit .....	72
232	Inferior timeout .....	73
233	Minimum inferior timeout .....	74
234	Inferior name .....	74
235	State Tables .....	76
236	Explanation of the state tables .....	76
237	Status queries .....	76
238	Decision events .....	76
239	Disruptions – failure events .....	77
240	Invalid cells and assumptions of the communication mechanism .....	77
241	Meaning of state table events .....	78
242	Persistent information .....	82
243	Failure Recovery .....	95

244	Types of failure .....	95
245	Persistent information .....	96
246	Redirection.....	97
247	Terminator:Decider failures.....	98
248	XML representation of Message Set.....	98
249	Addresses .....	99
250	Qualifiers .....	99
251	Identifiers .....	100
252	Message References.....	100
253	Messages.....	100
254	CONTEXT.....	100
255	CONTEXT_REPLY .....	100
256	BEGIN .....	101
257	BEGUN.....	101
258	ENROL .....	101
259	ENROLLED .....	102
260	RESIGN .....	102
261	RESIGNED.....	103
262	PREPARE .....	103
263	PREPARED .....	103
264	CONFIRM .....	103
265	CONFIRMED.....	104
266	CANCEL .....	104
267	CANCELLED.....	104
268	CONFIRM_ONE_PHASE .....	105
269	HAZARD.....	105
270	CONTRADICTION.....	105
271	SUPERIOR_STATE.....	106
272	INFERIOR_STATE.....	106
273	REDIRECT .....	106
274	PREPARE_INFERIORS .....	107
275	CONFIRM_TRANSACTION .....	107
276	TRANSACTION_CONFIRMED .....	108
277	CANCEL_TRANSACTION .....	108
278	CANCEL_INFERIORS .....	108
279	TRANSACTION_CANCELLED.....	109
280	REQUEST_INFERIOR_STATUSES .....	109
281	INFERIOR_STATUSES .....	109
282	REQUEST_STATUS .....	110
283	STATUS .....	110
284	FAULT.....	111
285	Standard qualifiers .....	112
286	Transaction timelimit.....	112
287	Inferior timeout .....	112
288	Minimum inferior timeout .....	112
289	Compounding of Messages.....	112
290	Carrier Protocol Bindings .....	114

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

# Part 1. Purpose and Features of BTP

## Introduction

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

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

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

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

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

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

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

345 **Development and Maintenance of the Specification**

346

347

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

348

349

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

350

351

352

353

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

354

355

356

❑ maintaining the specification in the light of implementation experiences

357

358

❑ coordinating publicity for BTP

359

360

❑ liaising with other standards bodies whose work affects or may be affected by  
BTP

361

362

363

❑ reviewing the appropriate time, in the light of implementation experience and  
user support, to put BTP forward for adoption as a full OASIS standard

364

365

366

367

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

368

369

[bt-spec@lists.oasis-open.org](mailto:bt-spec@lists.oasis-open.org)

370

371

372

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

373

374

375

The main list of the committee is:

376

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

377

378

379

380

381

382

383

## 383 Overview of the Business Transaction Protocol

384  
385 A Business Transaction is a consistent change in the state of a business relationship between  
386 two or more parties. BTP provides means to allow the consistent and coordinated changes in  
387 the relationship as viewed from each party.

388  
389 BTP assumes that for a given business transaction state changes occur, or are desired, in some  
390 set of parties, and that these changes are related in some business-defined manner.

391  
392 Typically business-defined messages (“application messages”) are exchanged between the  
393 parties to the transaction, which result in the performance of some set of operations. These  
394 operations create provisional or tentative state changes (the transaction’s effect). The  
395 provisional changes of each party must either be confirmed (given final effect), or must be  
396 cancelled (counter-effected). Those parties which are confirmed create an atomic unit, within  
397 which the business transaction should have a consistent final effect.

398  
399 The meaning of “effect”, “final effect” and “counter-effect” is specific to each business  
400 transaction and to each party’s role within it. A party may log intended changes (as its effect)  
401 and only process them as visible state changes on confirmation (its final effect). Or it may  
402 make visible state changes and store the information needed to cancel (its effect), and then  
403 simply delete the information needed for cancellation (its final effect). A counter-effect may  
404 be a precise inversion or removal of provisional changes, or it may be the processing of  
405 operations that in some way compensate for, make good, alleviate or supplement their effect.

406  
407 To ensure that confirmation or cancellation of the provisional effect within different parties  
408 can be consistently performed, it is necessary that each party should

- 409  
410  determine whether it is able both to cancel (counter-effect) and to confirm (give final  
411 effect to) its effect
- 412  
413  report its ability or inability to cancel-or-confirm (its preparedness) to a central  
414 coordinating entity

415  
416 After receiving these reports, the coordinating entity is responsible for determining which of  
417 the parties should be instructed to confirm and which should be instructed to cancel.

418  
419 Such a two-phase exchange (ask, instruct) mediated by a central coordinator is required to  
420 achieve a consistent outcome for a set of operations. BTP defines the means for software  
421 agents executing on network nodes to interoperate using a two-phase coordination protocol,  
422 leading either to the abandonment of the entire attempted transaction, or to the selection of an  
423 internally consistent set of confirmed operations.

424  
425 BTP centres on the bilateral relationship between the computer systems of the coordinating  
426 entity and those of one of the parties in the overall business transaction. In that relationship a  
427 software agent within the coordinating entity’s systems plays the BTP role of Superior for a  
428 given transaction and one or more software agents within the systems of the party play the  
429 BTP role of Inferior. Each Inferior has one Superior, therefore, while a single Superior may

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

432  
433 An Inferior is associated with some set of operation invocations that creates effect  
434 (provisional or tentative changes) within the party, for a given business transaction. The  
435 Inferior is responsible for reporting to its related Superior whether its associated operations'  
436 effect can be confirmed/cancelled. A Superior is responsible for gathering the reports of all of  
437 its Inferiors, in order to ascertain which should be cancelled or confirmed. For example, if a  
438 Superior is acting as an atomic Coordinator it will treat any Inferior which cannot prepare to  
439 cancel/confirm as having veto power over the whole business transaction, causing the  
440 Superior to instruct all its Inferiors to cancel. A Superior may, under the dictates of a  
441 controlling application, increase or reduce the set of Inferiors to which a common confirm or  
442 cancel outcome may be delivered. Thus, the set of prepared Inferiors may be larger than the  
443 set of confirmed Inferiors.

444  
445 An Inferior:Superior relationship is typically established in relation to one or more  
446 application messages sent from one part of the application (linked to the Superior) to some  
447 other part of the application to request the performance of operations that are to be subject to  
448 the confirm or cancel decision of the Superior. If an application is divided between a client  
449 and a service, which use RPCs to communicate application requests and responses, then the  
450 client would typically be associated with the Superior and the service would typically host the  
451 Inferior(s). (BTP does not mandate such an application topology nor does it require the use of  
452 RPC or any other application communication paradigm.)

453  
454 BTP defines a CONTEXT message that can be sent "in relation to" such application  
455 messages. On receipt of a CONTEXT, one or more Inferiors may be created and "enrolled"  
456 with the Superior, establishing the Superior:Inferior relationships. The particular mechanisms  
457 by which a CONTEXT is "related" to application messages is an issue for the application  
458 protocol and its binding to carrier mechanisms. BTP does not require that the enrolment is  
459 requested by any particular entity – in a particular implementation this may be done by the  
460 Inferior itself, by parts of the application or by other entities involved in the transmission of  
461 the CONTEXT and the application messages. BTP defines a CONTEXT\_REPLY message  
462 that can be sent on the return path of the CONTEXT to indicate whether the enrolment was  
463 successful. Without CONTEXT\_REPLY it would be possible for a Superior to have an  
464 incorrect view of which Inferiors it was supposed to involve in its confirm decision.

465  
466 It should be noted that this BTP specification recognises that:

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

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

## Part 2. Normative Specification of BTP

### Actors, Roles and Relationships

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

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

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

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

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

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

---

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

---

### Relationships

There are two primary relationships in BTP.

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

533

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

536

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

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

559

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

565

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

569

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

573

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

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

## 588 Roles involved in the outcome relationships

### 589 Superior

590  
591  
592 Accepts enrolments from Inferiors, establishing a Superior:Inferior relationship with each. In  
593 cooperation with other actors and constrained by the messages exchanged with the Inferior,  
594 the Superior determines the **Outcome** applicable to the Inferior and informs the Inferior by  
595 sending CONFIRM or CANCEL. This outcome can be confirm only if a PREPARED  
596 message is received from the Inferior, and if a record, identifying the Inferior can be  
597 persisted. (Whether this record is also a record of a confirm decision depends on the  
598 Superior’s position in the business transaction as a whole.). The Superior must retain this  
599 persistent record until it receives a CONFIRMED (or, in exceptional cases, CANCELLED or  
600 HAZARD) from the Inferior.  
601

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

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

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

613 A Superior receives

614 ENROL

615  
616 to enrol a new Inferior, establishing a new Superior:Inferior relationship.  
617

618 A Superior sends  
619  
620

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

625 A Superior sends

626  
627 PREPARE  
628 CONFIRM  
629 CANCEL  
630 RESIGNED  
631 CONFIRM\_ONE\_PHASE  
632 SUPERIOR\_STATE

633  
634 to an enrolled Inferior.

635  
636 A Superior receives

637  
638 PREPARED  
639 CANCELLED  
640 CONFIRMED  
641 HAZARD  
642 RESIGN  
643 INFERIOR\_STATE

644  
645 from an enrolled Inferior.

#### 646 647 **Inferior**

648  
649 Responsible for applying the Outcome to some set of associated operations – the application  
650 determines which operations are the responsibility of a particular Inferior.

651  
652 An Inferior is **Enrolled** with a single Superior (hereafter referred to as “its Superior”),  
653 establishing a Superior:Inferior relationship. If the Inferior is able to ensure that either a  
654 confirm or cancel decision can be applied to the associated operations, and can persist  
655 information to retain that condition, it sends a PREPARED message to the Superior. When  
656 the Outcome is received from the Superior, the Inferior applies it, deletes the persistent  
657 information, and replies with CANCELLED or CONFIRMED as appropriate.

658  
659 If an Inferior is unable to come to a prepared state, it cancels the associated operations and  
660 informs the Superior with a CANCELLED message. If it is unable to either come to a  
661 prepared state, or to cancel the associated operations, it informs the Superior with a  
662 HAZARD message.

663  
664 An Inferior that has become prepared may, exceptionally, make an autonomous decision to be  
665 applied to the associated operations, without waiting for the Outcome from the Superior. It is  
666 required to persist this autonomous decision and report it to the Superior with CONFIRMED  
667 or CANCELLED as appropriate. If, when CONFIRM or CANCEL is received, the

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

671

672 An Inferior receives

673

674 PREPARE  
675 CONFIRM  
676 CANCEL  
677 RESIGNED  
678 CONFIRM\_ONE\_PHASE  
679 SUPERIOR\_STATE

680

681 from its Superior.

682

683 An Inferior sends

684

685 PREPARED  
686 CANCELLED  
687 CONFIRMED  
688 HAZARD  
689 RESIGN  
690 INFERIOR\_STATE

691

692 to its Superior.

693

694

695 **Enroller**

696

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

702

703 An Enroller sends

704

705 ENROL

706

707 to a Superior.

708

709 An Enroller receives

710

711 ENROLLED

712

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

714

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

## 721 Participant

722  
723 An Inferior which is specialized for the purposes of an application. Some application  
724 operations are associated directly with the Participant, which is responsible for determining  
725 whether a prepared condition is possible for them, and for applying the outcome. (“associated  
726 directly” as opposed to involving another BTP Superior:Inferior relationship, in which this  
727 actor is the Superior).  
728

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

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

---

737 Note – Possible approaches are:

---

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

## 750 Sub-coordinator

751  
752 An Inferior which is also an Atomic Superior.  
753

754  
755 A sub-coordinator is the Inferior in one Superior:Inferior relationship and the Superior in one  
756 or more Superior:Inferior relationships.

757  
758 From the perspective of its Superior (the one the sub-coordinator is Inferior to), there is no  
759 difference between a sub-coordinator and any other Inferior. From this perspective, the  
760 “associated operations” of the sub-coordinator as an Inferior include the relationships with its  
761 Inferiors.

762  
763 A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and  
764 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is  
765 propagated to all Inferiors.

### 766 **Sub-composer**

767  
768  
769 An Inferior which is also a Cohesive Superior.

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

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

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

780  
781

## 782 **Roles involved in the control relationships**

### 783 **Decider**

784  
785  
786 A Superior that is not also the Inferior on a Superior:Inferior relationship. It is the top-node in  
787 the transaction tree and receives requests from a Terminator as to the desired outcome for the  
788 business transaction. If the Terminator asks the Decider to confirm the business transaction, it  
789 is the responsibility of the Decider to finally take the confirm decision. The taking of the  
790 decision is synonymous with the persisting of information identifying the Inferiors that are to  
791 be confirmed. An Inferior cannot be confirmed unless PREPARED has been received from it.

792  
793 A Decider is instructed to cancel by receiving CANCEL\_TRANSACTION.

794  
795 A Decider that is an Atomic Superior (all Inferiors will have the same outcome) is a  
796 Coordinator. A Decider that is a Cohesive Superior (some Inferiors may cancel, some  
797 confirm) is a Cohesion.

798  
799 All Deciders receive  
800 CONFIRM\_TRANSACTION  
801 CANCEL\_TRANSACTION  
802 REQUEST\_INFERIOR\_STATUSES  
803

804 All Deciders send  
805 CONFIRM\_COMPLETE  
806 CANCEL\_COMPLETE  
807 INFERIOR\_STATUSES  
808  
809

### 810 **Coordinator**

811  
812 A Decider that is an Atomic Superior. The same outcome decision will be applied to all  
813 Inferiors (excluding any from which RESIGN is received).  
814  
815 PREPARED must be received from all remaining Inferiors for a confirm decision to be taken.  
816  
817 A Coordinator must make a cancel decision if  
818 it is instructed to cancel by the Terminator  
819 if CANCELLED is received from any Inferior  
820 if it is unable to persist a confirm decision  
821

### 822 **Composer**

823  
824 A Decider that is a Cohesive Superior. If the Terminator requests confirmation of the  
825 Cohesion, that request will determine the confirm-set of the Cohesion.  
826  
827 PREPARED must be received from all Inferiors in the confirm-set (excluding any from  
828 which RESIGN is received) for a confirm decision to be taken.  
829  
830 A Composer must make a cancel decision (applying to all Inferiors) if  
831 it is instructed to cancel by the Terminator  
832 if CANCELLED is received from any Inferior in the confirm-set  
833 if it is unable to persist a confirm decision  
834  
835 A Composer may be asked to prepare some or all of its Inferiors by receiving  
836 PREPARE\_INFERIORS. It issues PREPARE to any of those Inferiors from which none of  
837 PREPARED, CANCELLED or RESIGN have been received, and replies to the  
838 PREPARE\_INFERIORS with INFERIOR\_STATUSES.  
839  
840 A Composer may be asked to cancel some of its Inferiors, but not itself, by receiving  
841 CANCEL\_INFERIORS.  
842

### 843 **Terminator**

844  
845  
846 Asks a Decider to confirm the business transaction, or instructs it to cancel all or (for a  
847 Cohesion) part of the business transaction.  
848  
849 All communications between Terminator and Decider are initiated by the Terminator. A  
850 Terminator is usually an application element.

851  
852 A request to confirm is made by sending CONFIRM\_TRANSACTION to the target Decider.  
853 If the Decider is a Cohesion Composer, the Terminator may select which of the Composer's  
854 Inferiors are to be included in the confirm-set. If the Decider is an Atom Coordinator, all  
855 Inferiors are included. After applying the decision, the Decider replies with  
856 CONFIRM\_COMPLETE, CANCEL\_COMPLETE or (in the case of problems)  
857 INFERIOR\_STATUSES.

858  
859 A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its  
860 Inferiors with PREPARE\_INFERIORS. The Composer replies with  
861 INFERIOR\_STATUSES.

862  
863 A Terminator may send CANCEL\_TRANSACTION to instruct the Decider to cancel the  
864 whole business transaction.,. The Decider replies with CANCEL\_COMPLETE if all Inferiors  
865 cancel successfully, and with INFERIOR\_STATUSES in the case of problems.. If the  
866 Decider is a Cohesion Composer, the Terminator may send CANCEL\_INFERIORS to cancel  
867 some of the Inferiors; the Decider always replies with INFERIOR\_STATUSES.

868  
869 A Terminator may check the status of the Inferiors of the Decider by sending  
870 REQUEST\_INFERIOR\_STATUSES. The Decider replies with INFERIOR\_STATUSES.

871  
872 A Terminator sends  
873 CONFIRM\_TRANSACTION  
874 CANCEL\_TRANSACTION  
875 CANCEL\_INFERIORS  
876 PREPARE\_INFERIORS  
877 REQUEST\_INFERIOR\_STATUSES

878  
879 A Terminator receives  
880 CONFIRM\_COMPLETE  
881 CANCEL\_COMPLETE  
882 INFERIOR\_STATUSES

## 884 Initiator

885  
886 Requests a **Factory** to create a Superior – this will either be a Decider (representing a new  
887 top-level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an  
888 existing business transaction.

889 An Initiator sends

890 BEGIN  
891 BEGIN & CONTEXT

892  
893 to a Factory, and receives in reply

894  
895 BEGUN & CONTEXT

898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944

## Factory

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

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

A Factory receives

BEGIN  
BEGIN & CONTEXT

and replies with

BEGUN & CONTEXT

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

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

## Other roles

### Redirector

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

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

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

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

946

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

949

## 950 **Status Requestor**

951

952 Requests and receives the current status of a transaction tree node – any of an Inferior,  
953 Superior or Decider, or the current status of the nodes relationships with its Inferiors, if any.

954 The role of Status Requestor has no responsibilities – it is just a name for where the

955 REQUEST\_STATUS and REQUEST\_INFERIOR\_STATUSES comes from

956 (REQUEST\_INFERIOR\_STATUSES is also issued by a Terminator to a Decider).

957

958 A Status Requestor sends

959

960 REQUEST\_STATUS

961 REQUEST\_INFERIOR\_STATUSES

962

963 and receives

964

965 STATUS

966 INFERIOR\_STATUSES

967

968 in response.

969

970 The receiver of the request can refuse to provide the status information by replying with

971 FAULT(StatusRefused). The information returned in STATUS will always relate to the

972 transaction tree node as a whole (e.g. as an Inferior, even if it is also a Superior).

973

## 974 **Abstract Messages and Associated Contracts**

975

976 BT Protocol Messages are defined in this section in terms of the abstract information that has

977 to be communicated. These abstract messages will be mapped to concrete messages

978 communicated by a particular carrier protocol (there can be several such mappings defined).

979

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

981

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

984

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

987

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

989

990 and

991

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

993

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

996

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

1003

1004

## 1005                   **Addresses**

1006

1007                   All of the messages except CONTEXT and CONTEXT\_REPLY have a “target address”  
1008                   parameter and many also have other address parameters. These latter identify the desired  
1009                   target of other messages in the set. In all cases, the exact value will invariably have been  
1010                   originally determined by the implementation that is the target or desired future target.

1011

1012                   The detailed format of the address will depend on the particular carrier protocol, but at this  
1013                   abstract level is considered to have three parts. The first part, the “binding name”, identifies  
1014                   the binding to a particular carrier protocol – some bindings are specified in this document,  
1015                   others can be specified elsewhere. The second part of the address, the “binding address”, is  
1016                   meaningful to the carrier protocol itself, which will use it for the communication (i.e. it will  
1017                   permit a message to be delivered to a receiver). The third part, “additional information”, is  
1018                   not used or understood by the carrier protocol. The “additional information” may be a  
1019                   structured value.

1020

1021                   When a message is actually transmitted, the “binding name” of the target address will identify  
1022                   which carrier protocol is in use and the “binding address” will identify the destination, as  
1023                   known to the carrier protocol. The entire binding address is considered to be “consumed” by  
1024                   the carrier protocol implementation. All of it may be used by the sending implementation, or  
1025                   some of it may be transmitted in headers, or as part of a URL in the carrier protocol, but then  
1026                   used or consumed by the receiving implementation of the carrier protocol to direct the BTP  
1027                   message to a BTP-aware entity (BTP-aware in that it is capable of interpreting the BTP  
1028                   messages). The “additional information” of the target address will be part of the BTP  
1029                   message itself and used in some way by the receiving BTP-aware entity (it could be used to  
1030                   route the message on to some other BTP entity). Thus, for the target address, only the  
1031                   “additional information” field is transmitted in the BTP message and the “additional  
1032                   information” is opaque to parties other than the recipient.

1033

1034                   For other addresses in BTP messages, all three components will be within the message.

1035

1036                   All messages that concern a particular Superior:Inferior relationship have an identifier  
1037                   parameter for the target side as well as the ~~compound~~-target address. This allows full  
1038                   flexibility for implementation choices – an implementation can:

1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085

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

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

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

### **Request/response pairs**

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

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

### **Compounding messages**

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

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

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

1086 the transmission of the bundle "A+B" is semantically identical to the transmission of A  
1087 followed by the transmission of B.

1088  
1089 Only certain combinations of messages are possible in a group, and the meaning of the  
1090 relation is specifically defined for each such combination in the next section. A particular  
1091 group is treated as a unit for transmission – it has a single target address. This is usually that  
1092 of one of the messages in the group – the specification for the group defines which.

1093  
1094 A “bundle” of messages may contain both unrelated messages and groups of related  
1095 messages. The only constraint on which messages and groups can be bundled is that all have  
1096 the same binding address, but may have different “additional information” values. (Messages  
1097 within a related group may have different addresses, where the rules of their relatedness  
1098 permit this). Unless constrained by the binding, any messages or groups that are to be sent to  
1099 the same binding address may be bundled – the fact that the binding addresses are the same is  
1100 a necessary and sufficient condition for the sender to determine that the messages can be  
1101 bundled.

1102  
1103 A particular and important case of related messages is where a BTP CONTEXT message is  
1104 sent related to an application message. In this case, the target of the application message  
1105 defines the destination of the CONTEXT message. The receiving implementation may in fact  
1106 remove the CONTEXT before delivering the application message to the application (Service)  
1107 proper, but from the perspective of the sender, the two are sent to the same place.  
1108 The compounding mechanisms, and the multi-part address structures, support the “one-wire”  
1109 and “one-shot” communication patterns.

1110  
1111 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,  
1112 including the associated application messages, pass via the same “endpoints”. These  
1113 “endpoints” may in fact be relays, routing messages on to particular actors within their  
1114 domain. The onward routing will require some further addressing, but this has to be opaque to  
1115 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors  
1116 in its domain have the relay’s address as their binding address, and any routing information it  
1117 will need in its own domain is placed in the additional information. (This may involve the  
1118 relay changing addresses in messages as they pass through it on the way out). On receiving a  
1119 message, it determines the within-domain destination from the received additional  
1120 information (which is thus rewritten) and forwards the message appropriately. The sender is  
1121 unaware of this, and merely sees addresses with the same binding address, which it is  
1122 permitted to bundle. The content of the “additional information” is a matter only for the relay  
1123 – it could put an entire BTP address in there, or other implementation-defined information.  
1124 Note that a quite different one-wire implementation can be constructed where there is no  
1125 relaying, but the receiving entity effectively performs all roles, using the received identifiers  
1126 to locate the appropriate state.

1127  
1128 “One-shot” communication makes it possible to send an application message, receive the  
1129 application reply, enrol an Inferior to be responsible for the confirm/cancel of the operations  
1130 of those message and inform the Superior that the Inferior is prepared, all in one two-way  
1131 exchange across the network (e.g. one request/reply of a carrier protocol).. The application  
1132 request is sent with a related CONTEXT message. The application response is sent with a

1133 relation group of CONTEXT\_REPLY/related, ENROL/no-rsp-req message and a  
1134 PREPARED message. This is possible even if the Superior address is different from the  
1135 address of the application element that sends the original message (if the application  
1136 exchange is request/reply, there may not even be an identifiable address for the application  
1137 element). The target addresses of the ENROL and PREPARED (the Superior address) are not  
1138 transmitted; the actor that was originally responsible for adding the CONTEXT to the  
1139 outbound application message remembers the Superior address and forwards the ENROL and  
1140 PREPARED appropriately.

1141  
1142 With “one-shot”, if there are multiple Inferiors created as a result of a single application  
1143 message, there is an ENROL and PREPARED message for each sent related to the  
1144 CONTEXT\_REPLY. If an operation fails, a CANCELLED message is sent instead of a  
1145 PREPARED.

1146  
1147 If the CONTEXT has “superior-type” of “atom”, then subsequent messages to the same  
1148 Service, with the same related CONTEXT/atom, can have their associated operations put  
1149 under the control of the same Inferior, and only a CONTEXT\_REPLY/completed is sent back  
1150 with the response (if the new operations fail, it will be necessary to send back  
1151 CONTEXT\_REPLY/repudiated, or send CANCELLED). If the “superior type” on the  
1152 CONTEXT is “cohesive”, each operation will require separate enrolment.

1153  
1154 Whether the “one-shot” mechanism is used is determined by the implementation on the  
1155 responding (Inferior) side. This may be subject to configuration and may also be constrained  
1156 by the application or by the binding in use.

1157

## 1158 **Extensibility**

1159

1160 To simplify interoperation between implementations of this edition of BTP with  
1161 implementations of future editions, the “must-be-understood” sub-parameter as specified for  
1162 Qualifiers may be defined for use with any parameter added to an existing message in a future  
1163 revision of this specification. The default for “must-be-understood” shall be “true”, so an  
1164 implementation receiving an unrecognised parameter without a “false” value for “must-be-  
1165 understood” shall not accept it (the FAULT value “UnrecognisedParameter” is available, but  
1166 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If  
1167 “must-be-understood” with the value “false” is present as a sub-parameter of a parameter in  
1168 any message, a receiving implementation **should** ignore the parameter.

1169

1170 How the sub-parameter is associated with the new parameter is determined by the particular  
1171 binding.

1172

1173 No special mechanism is provided to allow for the introduction of completely new messages.

1174

## 1175 **Inferior handle**

1176

1177 ~~Some of the messages exchanged between a Terminator and a Decider are concerned with the~~  
1178 ~~individual Inferiors enrolled with the Decider, and not with the business transaction as a~~

1179 whole. These messages distinguish the Inferiors of Decider using an “inferior handle”. This is  
1180 created by the Decider and is unambiguous within the scope of the Decider.

1181  
1182 The “inferior handle” is distinct from the “inferior identifier” passed on an ENROL message  
1183 (among other places). The latter is created by the Inferior (or its enroller) and is required to be  
1184 unambiguous within the scope of the address as inferior on the ENROL (and unambiguous  
1185 within any of the individual addresses in that set of BTP addresses—the identifier must  
1186 identify the Inferior across all the places it might migrate to or that have recovery  
1187 responsibility for it).

1188  
1189 The “inferior handle” is only used by the Terminator to refer to the inferiors of the Decider.  
1190 In messages between the Decider and its Inferiors, the address as inferior and inferior  
1191 identifier are used.

1192

## 1193 Messages

1194

### 1195 Qualifiers

1196

1197 All messages have a Qualifiers parameter which contains zero or more Qualifier values. A  
1198 Qualifier has sub-parameters:

1199

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

1200

1201 **Qualifier group** ensures the Qualifier name is unambiguous. Qualifiers in the  
1202 same group need not have any functional relationship. The qualifier group will  
1203 typically be used to identify the specification that defines the qualifier’s meaning  
1204 and use. Qualifiers may be defined in this or other standard specifications, in  
1205 specifications of a particular community of users or of implementations or by  
1206 bilateral agreement.

1207

1208 **Qualifier name** this identifies the meaning and use of the Qualifier, using a name  
1209 that is unambiguous within the scope of the Qualifier group.

1210

1211 **Must-be-understood** if this has the value “true” and the receiving entity does  
1212 not recognise the Qualifier type (or does not implement the necessary  
1213 functionality), a FAULT “UnsupportedQualifier” shall be returned and the  
1214 message shall not be processed. Default is “true”.

1215

1216 **To-be-propagated** if this has the value “true” and the receiving entity passes the  
1217 BTP message (which may be a CONTEXT, but can be other messages) onwards  
1218 to other entities, the same Qualifier value shall be included. If the value is  
1219 “false”, the Qualifier shall not be automatically included if the BTP message is  
1220 passed onwards. (If the receiving entity does support the qualifier type, it is  
1221 possible a propagated message may contain another instance of the same type,  
1222 even with the same Content – this is not considered propagation of the original  
1223 qualifier.). Default is “false”.

1224  
1225 **Content** the type (which may be structured) and meaning of the content is  
1226 defined by the specification of the Qualifier.

1227  
1228

### 1229 **Messages not restricted to outcome or control relationships.**

1230  
1231 The messages in this section are used between various roles. CONTEXT message is used in  
1232 the Initiator:Factory relationship (when it is related to BEGIN or to BEGUN), and related to  
1233 an application ‘message’ to propagate the business transaction between parts of the  
1234 application. CONTEXT\_REPLY is used as the reply to a CONTEXT.REQUEST\_STATUS  
1235 can be issued to, and STATUS returned by any of Decider, Superior or Inferior. FAULT can  
1236 be used on any relationship to indicate an error condition back to the sender of a message.

1237  
1238

### CONTEXT

1239  
1240 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more  
1241 application messages. (The means by which this relationship is represented is determined by  
1242 the binding and the binding mechanisms of the application protocol.) The “superior type”  
1243 parameter identifies whether the Superior will apply the same decision to all Inferiors  
1244 enrolled using the same superior identifier (“superior type” is “atom”) or whether it may  
1245 apply different decisions (“superior type” is “cohesion”).

1246

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

1247  
1248

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

1251

1252 **superior identifier** identifies the Superior. This shall be globally unambiguous.  
1253 within the scope of the address-as-superior

1254  
1255 **reply-address** the address to which a replying CONTEXT\_REPLY is to be sent.  
1256 This may be different each time the CONTEXT is transmitted – it refers to the  
1257 destination of a replying CONTEXT\_REPLY for this particular transmission of  
1258 the CONTEXT.

1259  
1260 **superior type** identifies whether the CONTEXT refers to a Cohesion or an  
1261 Atom. Default is atom.

1262  
1263  
1264 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction  
1265 timelimit” is carried by CONTEXT.

1266  
1267 There is no target address parameter for CONTEXT as it is only transmitted in relation to the  
1268 application messages, BEGIN and BEGUN.

1269  
1270 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the  
1271 superior type with the appropriate value.

1272  
1273

## 1274 CONTEXT\_REPLY

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

1283

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

1284  
1285 **target-address** the address to which the CONTEXT\_REPLY is sent. This shall  
1286 be the “reply-address” from the CONTEXT.

1287  
1288 ~~superior-address~~ one of the addresses from the address as superior from the  
1289 CONTEXT. (The parameter is present in CONTEXT\_REPLY to disambiguate  
1290 the superior identifier.)

1291

1292 **superior identifier** the superior identifier from the CONTEXT

1293

1294 **completion\_status:** reports whether all enrol operations made necessary by the  
1295 receipt of the earlier CONTEXT message have completed. Values are

1296

Value	meaning
<i>completed</i>	All enrolments (if any) have succeeded already
<i>related</i>	At least some enrolments are to be performed by ENROL messages related to the CONTEXT_REPLY. All other enrolments (if any) have succeeded already.
<i>repudiated</i>	At least one enrolment has failed. The implications of receiving the CONTEXT have <b>not</b> been honoured.

1297

1298 **qualifiers** standardised or other qualifiers.

1299

1300 The form CONTEXT\_REPLY/completed, CONTEXT\_REPLY/related and  
1301 CONTEXT\_REPLY/repudiated refer to CONTEXT\_REPLY messages with status having the  
1302 appropriate value. The form CONTEXT\_REPLY/ok refers to either of  
1303 CONTEXT\_REPLY/completed or CONTEXT\_REPLY/related.

1304

1305 If there are no necessary enrolments (e.g. the application messages related to the received  
1306 CONTEXT did not require the enrolment of any Inferiors), then  
1307 CONTEXT\_REPLY/completed is used.

1308

1309 If a CONTEXT\_REPLY/repudiated is received, the receiving implementation **must** ensure  
1310 that the business transaction will not be confirmed.

1311

1312

## 1313 REQUEST\_STATUS

1314

1315 Sent to an Inferior, Superior or to a Decider to ask it to reply with STATUS. The receiver  
1316 may reject the request with a FAULT(StatusRefused).

1317

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

1318

1319 **target address** the address to which the REQUEST\_STATUS message is sent.  
1320 This can be any of address-as-decider, address-as-inferior or address-as-superior.

1321

1322 **reply address** the address to which the replying STATUS should be sent.

1323  
1324 **target identifier** The identifier for the business transaction, or part of business  
1325 transaction whose status is sought. If the target-address is an address-as-decider,  
1326 this parameter shall be the “transaction-identifier” on the BEGUN message. If the  
1327 target-address is an address-as-inferior, this parameter shall be the “inferior-  
1328 identifier” on the ENROL message. If the target-address is a an address-as-  
1329 superior, this parameter shall be the “superior-identifier” on the CONTEXT.

1330  
1331 **qualifiers** standardised or other qualifiers.

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

1334  
1335 **General**  
1336 **StatusRefused** – if the receiver is not prepared to report its status to the  
1337 sender of this message  
1338 **UnknownTransaction** – if the target-identifier is unknown

1339  
1340  
1341 **STATUS**

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

1345

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

1346  
1347 **target address** the address to which the STATUS is sent. This will be the reply  
1348 address on the REQUEST\_STATUS message

1349  
1350 ~~responders address~~ the address of the sender of the STATUS message—one of  
1351 address-as-inferior, address-as-decider, address-as-superior(with the responders-  
1352 identifier, this determines who the message is from).. If the sender has different  
1353 addresses as multiple roles (as Decider, Inferior or Superior), this shall be the  
1354 address on which the REQUEST\_STATUS was received.

1355  
1356 **responders-identifier** the identifier of the state, identical to the “target-  
1357 identifier” on the REQUEST\_STATUS, aligned with the responders-address. If  
1358 the sender has multiple roles in the transaction (as Decider, Inferior or Superior),  
1359 this shall be the target-identifier on the REQUEST\_STATUS

1360  
 1361  
 1362  
 1363  
 1364  
 1365

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

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

status value	Meaning from Superior	Meaning from Inferior
<i>Hazard</i>	A hazard has been reported from at least one Inferior	A hazard has been discovered; CONTRADICTION has not been received
<i>Contradicted</i>	Not applicable	CONTRADICTION has been received
<i>Unknown</i>	No state information for the target-identifier exists	No state information for the target-identifier exists
<i>Inaccessible</i>	There may be state information for this target-identifier but it cannot be reached/existence cannot be determined	There may be state information for this target-identifier but it cannot be reached/existence cannot be determined

1366

1367

**qualifiers** standardised or other qualifiers.

1368

1369

Types of FAULT possible

1370

### *General*

1371

1372

## **FAULT**

1373

1374

Sent in reply to various messages to report an error condition

1375

1376

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

1377

1378

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

1379

1380

1381

1382

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

1383

1384

1385

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

1386

1387

1388  
1389  
1390  
1391  
1392  
1393

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

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

1394

<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	Free text explanation
<i>InvalidDecider</i>	The address the message was sent to is not valid (at all or for this Terminator and transaction identifier)	The address
<i>InvalidInferior</i>	The Superior is known but the Inferior identified by the address-as-inferior and identifier are not enrolled in it	The Inferior Identity (address-as-inferior and identifier)
<i>InvalidSuperior</i>	The received identifier is not known or does not identify a known Superior	The identifier
<i>StatusRefused</i>	The receiver will not report the request status (or inferior statuses) to this StatusRequestor	Free text explanation
<i>InvalidTerminator</i>	The address the message was sent to is not valid (at all or for this Decider and transaction identifier)	The address
<i>UnknownParameter</i>	A BTP message has been received with an unrecognised parameter	Free text explanation
<i>UnknownTransaction</i>	The transaction-identifier is unknown	The transaction-identifier
<i>UnsupportedQualifier</i>	A qualifier has been received that is not recognised and on which "must-be-Understood" is "true".	Qualifier group and name
<i>WrongState</i>	The message has arrived when the recipient is in an invalid state.	

1395

1396 *UnknownParameter* A BTP message has been Free text explanation  
 1397 received with an unrecognised  
 1398 **q** parameter  
 1399 **u**  
 1400 **Qualifiers** standardised or other qualifiers.  
 1401

---

1402 Note – If the carrier mechanism used for the transmission of BTP messages  
 1403 is capable of delivering messages in a different order than they were sent in,  
 1404 the “WrongState” FAULT is not sent and should be ignored if received.

---

1405  
 1406 **REQUEST\_INFERIOR\_STATUSES, INFERIOR\_STATUSES**

1407  
 1408 REQUEST\_INFERIOR\_STATUSES may be sent to and INFERIOR\_STATUSES sent from  
 1409 any Decider, Superior or Inferior, asking it to report on the status of its relationships with  
 1410 Inferiors (if any). Since Deciders are required to respond to  
 1411 REQUEST\_INFERIOR\_STATUSES with INFERIOR\_STATUSES but non-Deciders may  
 1412 just issue FAULT(StatusRefused), and INFERIOR\_STATUSES is also used as a reply to  
 1413 other messages from Terminator to Decider, these messages are described below under the  
 1414 messages used in the control relationships.  
 1415

1416 **Messages used in the outcome relationships**

1417  
 1418 **ENROL**

1419  
 1420 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a  
 1421 CONTEXT message in relation to an application request.  
 1422 The actor issuing ENROL plays the role of Enroller.  
 1423

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

1424  
 1425 **target address** the address to which the ENROL is sent. This will be the  
 1426 address-as-superior from the CONTEXT message.  
 1427

1428 **superior identifier.** The superior identifier as on the CONTEXT message  
 1429  
 1430 **reply requested** true if an ENROLLED response is required, false otherwise.  
 1431 Default is false.  
 1432  
 1433 **reply address** the address to which a replying ENROLLED is to be sent, if  
 1434 “reply requested” is true. If this field is absent and “reply requested” is true, the  
 1435 ENROLLED should be sent to the “address-as-inferior” (or one of them, at  
 1436 sender’s option)  
 1437  
 1438 **address-as-inferior** the address to which PREPARE, CONFIRM, CANCEL and  
 1439 SUPERIOR\_STATE messages for this Inferior are to be sent.  
 1440  
 1441 **inferior identifier** an identifier that ~~unambiguously~~ identifies this Inferior. ~~This~~  
 1442 ~~shall be globally unambiguous, within the scope of any of the address-as-inferior~~  
 1443 ~~set of BTP addresses.~~  
 1444  
 1445 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior  
 1446 name” may be present.  
 1447

1448 Types of FAULT possible (sent to Reply address)

1449  
 1450 **General**

1451 **InvalidSuperior** – if superior identifier is unknown

1452 **DuplicateInferior** – if inferior with at least one of the set address-as-  
 1453 inferior the same and the same inferior identifier is already enrolled

1454 **WrongState** – if it is too late to enrol new Inferiors (generally if the  
 1455 Superior has already sent a PREPARED message to its superior or  
 1456 terminator, or if it has already issued CONFIRM to other Inferiors).  
 1457

1458 The form ENROL/rsp-req refers to an ENROL message with “reply requested” having the  
 1459 value “true”; ENROL/no-rsp-req refers to an ENROL message with “reply requested” having  
 1460 the value “false”  
 1461

1462 ENROL/no-rsp-req is typically sent in relation to CONTEXT\_REPLY/related. ENROL/rsp-  
 1463 req is typically when CONTEXT\_REPLY/completed will be used (after the ENROLLED  
 1464 message has been received.)  
 1465

1466 **ENROLLED**

1467  
 1468 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been  
 1469 successfully enrolled (and will therefore be included in the termination exchanges)  
 1470

Parameter	Type
target address	BTP address

Parameter	Type
inferior identifier	Identifier
<del>inferior handle</del>	<del>Handle</del>
Qualifiers	List of qualifiers

1471

1472

1473

1474

1475

1476

1477

1478

1479

1480

1481

1482

1483

1484

1485

1486

1487

1488

1489

1490

1491

1492

1493

1494

1495

1496

1497

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

**inferior identifier** The inferior identifier as on the ENROL message

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

**qualifiers** standardised or other qualifiers.

No FAULT messages are issued on receiving ENROLLED.

## RESIGN

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

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

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

1498

1499

1500

1501

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

1502 **superior-identifier** The superior identifier as on the ENROL message  
 1503  
 1504 ~~address-as-inferior~~ The address as inferior as on the earlier ENROL message  
 1505 ~~(with the inferior identifier, this determines who the message is from)~~  
 1506  
 1507 **inferior-identifier** The inferior identifier as on the earlier ENROL message  
 1508  
 1509 **response-requested** is set to “true” if a RESIGNED response is required.  
 1510  
 1511 **qualifiers** standardised or other qualifiers.  
 1512

1513 Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued  
 1514 early.  
 1515

1516 Types of FAULT possible (sent to address-as-inferior)  
 1517

1518 *General*

1519 *InvalidSuperior* – if superior identifier is unknown

1520 *InvalidInferior* – if no ENROL had been received for this address-as-  
 1521 inferior and identifier (Inferior Identity)

1522 *WrongState* – if a PREPARED or CANCELLED has already been  
 1523 received by the Superior from this Inferior  
 1524

1525 The form RESIGN/rsp-req refers to an RESIGN message with “reply requested” having the  
 1526 value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “reply requested”  
 1527 having the value “false”  
 1528

1529  
 1530 **RESIGNED**

1531 Sent in reply to a RESIGN/rsp-req message.  
 1532  
 1533

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

1534  
 1535 **target address** the address to which the RESIGNED is sent. This will be the  
 1536 address-as-inferior from the ENROL message.  
 1537  
 1538 **inferior identifier** The inferior identifier as on the earlier ENROL message for  
 1539 this Inferior.  
 1540  
 1541 **qualifiers** standardised or other qualifiers.

1542  
 1543 After receiving this message the Inferior will not receive any more messages with this  
 1544 address-as-inferior and identifier.  
 1545  
 1546 No FAULT messages are issued on receiving RESIGNED.  
 1547  
 1548 **PREPARE**  
 1549  
 1550 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor  
 1551 RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after  
 1552 receiving a PREPARED message.  
 1553  
 1554

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

1555  
 1556 **target address** the address to which the PREPARE message is sent. When sent  
 1557 from Superior to Inferior, this will be the address-as-inferior from the ENROL  
 1558 message.  
 1559

1560 **inferior identifier** When sent from Superior to Inferior, the inferior identifier as  
 1561 on the earlier ENROL message.  
 1562  
 1563

1564 **qualifiers** standardised or other qualifiers. The standard qualifier “Minimal  
 1565 inferior timeout” is carried by PREPARE.  
 1566  
 1567

1568 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or  
 1569 RESIGN.  
 1570

1571 Types of FAULT possible (sent to Superior address)  
 1572

**General**

1573 **InvalidInferior** – if inferior identifier is unknown, or an inferior-handle  
 1574 on the inferiors-list is unknown  
 1575

1576 **WrongState** – if a CONFIRM or CANCEL has already been received by  
 1577 this Inferior.  
 1578  
 1579

1580 **PREPARED**  
 1581

1582 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when  
 1583 the Inferior has determined the operations associated with the Inferior can be confirmed and  
 1584 can be cancelled, as may be instructed by the Superior. The level of isolation is a local matter  
 1585 (i.e. it is the Inferiors choice, as constrained by the shared understanding of the application  
 1586 exchanges) – other access may be blocked, may see applied results of operations or may see  
 1587 the original state.  
 1588

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

1589  
 1590 **target address** the address to which the PREPARED is sent. This will be the  
 1591 Superior address as on the ENROL message.  
 1592

1593 ~~**superior identifier** When the message is sent from an Inferior to the Superior,~~  
 1594 ~~the superior identifier as on the ENROL message~~

1595  
 1596 ~~**address as inferior** When the message is sent from an Inferior to the Superior,~~  
 1597 ~~the address as inferior as on the earlier ENROL message (with the inferior~~  
 1598 ~~identifier, this determines who the message is from)~~

1599  
 1600 **inferior identifier** The inferior identifier as on the ENROL message  
 1601

1602 **default is cancel** if “true”, the Inferior states that if the outcome at the Superior  
 1603 is to cancel the operations associated with this Inferior, no further messages need  
 1604 be sent to the Inferior. If the Inferior does not receive a CONFIRM message, it  
 1605 will cancel the associated operations. The value “true” will invariably be used  
 1606 with a qualifier indicating under what circumstances (usually a timeout) an  
 1607 autonomous decision to cancel will be made. If “false”, the Inferior will expect  
 1608 a CONFIRM or CANCEL message as appropriate, even if qualifiers indicate that  
 1609 an autonomous decision will be made.  
 1610

1611 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior  
 1612 timeout” may be carried by PREPARED.  
 1613

1614 On sending a PREPARED, the Inferior undertakes to maintain its ability to confirm or cancel  
 1615 the effects of the associated operations until it receives a CONFIRM or CANCEL message.  
 1616 Qualifiers may define a time limit or other constraints on this promise. The “default is

1617 cancel” parameter affects only the subsequent message exchanges and does not of itself state  
1618 that cancellation will occur.

1619  
1620 Types of FAULT possible (sent to address-as-inferior)

1621  
1622 *General*  
1623 *InvalidSuperior* – if Superior identifier is unknown  
1624 *InvalidInferior* – if no ENROL has been received for this address-as-  
1625 inferior and identifier, or if RESIGN has been received from this Inferior

1626  
1627 The form PREPARED/cancel refers to a PREPARED message with “default is cancel” =  
1628 “true”. The unqualified form PREPARED refers to a PREPARED message with “default is  
1629 cancel” = “false”.

1630  
1631

## 1632 CONFIRM

1633  
1634 Sent by the Superior to an Inferior from whom PREPARED has been received.  
1635

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

1636  
1637 **target address** the address to which the CONFIRM message is sent. This will  
1638 be the address-as-inferior from the ENROL message.

1639  
1640 **inferior identifier** The inferior identifier as on the earlier ENROL message for  
1641 this Inferior.

1642  
1643 **qualifiers** standardised or other qualifiers.

1644  
1645 On receiving CONFIRM, the Inferior is released from its promise to be able to undo the  
1646 operations of associated with the Inferior. The effects of the operations can be made available  
1647 to everyone (if they weren’t already).

1648  
1649 Types of FAULT possible (sent to Superior address)

1650  
1651 *General*  
1652 *InvalidInferior* – if inferior identifier is unknown  
1653 *WrongState* – if no PREPARED has been sent by, or if CANCEL has  
1654 been received by this Inferior.

1655  
1656

1657 **CONFIRMED**

1658  
1659  
1660  
1661  
1662  
1663

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

Parameter	Type
target address	BTP address
superior identifier	Identifier
<del>address-as-inferior</del>	<del>Set of BTP addresses</del>
inferior identifier	Identifier
confirm received	Boolean
qualifiers	List of qualifiers

1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692

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

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

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

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

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

**qualifiers** standardised or other qualifiers.

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

**General**

**InvalidSuperior** – if Superior identifier is unknown

1693 *InvalidInferior* – if no ENROL has been received for this address-as-  
1694 inferior and identifier, or if RESIGN has been received from this Inferior.  
1695

---

1696 Note – A CONFIRMED message arriving before a CONFIRM message is  
1697 sent, or after a CANCEL has been sent will occur when the Inferior has  
1698 taken an autonomous decision and is not regarded as occurring in the wrong  
1699 state. (The latter will cause a CONTRADICTION message to be sent.)

---

1700 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm  
1701 received” = “false”; CONFIRMED/response refers to a CONFIRMED message  
1702 with “confirm received” = ”true”.  
1703  
1704

1705

## CANCEL

1706 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.  
1707  
1708  
1709

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

1710  
1711 **target address** the address to which the CANCEL message is sent. ~~When sent~~  
1712 ~~from Superior to Inferior,~~ ~~t~~ This will be the address-as-inferior from the ENROL  
1713 message.

1714  
1715 **inferior identifier** ~~When sent from Superior to Inferior,~~ the inferior identifier as  
1716 on the earlier ENROL message.  
1717

1718 **qualifiers** standardised or other qualifiers.  
1719

1720 When ~~sent to an~~ ~~received by an~~ Inferior, the effects of any operations associated with the  
1721 Inferior should be undone. If the Inferior had sent PREPARED, the Inferior is released from  
1722 its promise to be able to confirm the operations.  
1723

1724 Types of FAULT possible (sent to Superior address)  
1725

1726 *General*  
1727 *InvalidInferior* – if inferior identifier is unknown, or an inferior-handle  
1728 on the inferiors-list is unknown

1729 *WrongState* – if a CONFIRM has been received by this Inferior.  
1730  
1731

1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751

**CANCELLED**

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

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

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

**Parameter**

target address	BTP address
superior identifier	Identifier
<del>address-as-inferior</del>	<del>Set of BTP address</del>
inferior identifier	Identifier
qualifiers	List of qualifiers

1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769

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

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

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

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

**qualifiers** standardised or other qualifiers.

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

1770  
1771  
1772  
1773  
1774  
1775  
1776

*General*

*InvalidSuperior* – if Superior identifier is unknown

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

*WrongState* – if CONFIRM has been sent

1777  
1778  
1779  
1780

---

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

---

1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788

**CONFIRM\_ONE\_PHASE**

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

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

1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804

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

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

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

**qualifiers** standardised or other qualifiers.

1805 CONFIRM\_ONE\_PHASE can be issued by a Superior to an Inferior from whom  
1806 PREPARED has been received (subject to the requirement that there is only one enrolled  
1807 Inferior).

1808  
1809 Types of FAULT possible (sent to Superior address)

1810  
1811 *General*  
1812 *InvalidInferior* – if inferior identifier is unknown  
1813 *WrongState* – if a PREPARE has already been received from this  
1814 Inferior

1815  
1816 **HAZARD**

1817  
1818 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly  
1819 and consistently cancel or confirm the operations in accord with the decision (either the  
1820 received decision of the superior or its own autonomous decision), or when the Inferior is  
1821 unable to determine that a “mixed” condition has not occurred.

1822  
1823 HAZARD is also used to reply to a CONFIRM\_ONE\_PHASE if the Inferior determines there  
1824 is a mixed condition within its associated operations or is unable to determine that there is not  
1825 a mixed condition.  
1826

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

1827  
1828 **target address** the address to which the HAZARD is sent. This will be the  
1829 superior address from the ENROL message.

1830  
1831 **superior identifier** The superior identifier as ~~used~~ on the ENROL message

1832  
1833 ~~address as inferior~~ ~~The address as inferior as on the earlier ENROL message~~  
1834 ~~(with the inferior identifier, this determines who the message is from)~~

1835  
1836 **inferior identifier** The inferior identifier as on the earlier ENROL message

1837  
1838 **level** indicates, with value “mixed” that a mixed condition has definitely  
1839 occurred; or, with value “possible” that it is unable to determine whether a mixed  
1840 condition has occurred or not.

1841  
1842 **qualifiers** standardised or other qualifiers.

1843  
1844 Types of FAULT possible (sent to address-as-inferior)

1845  
1846 **General**  
1847 **InvalidSuperior** – if Superior identifier is unknown  
1848 **InvalidInferior** – if no ENROL has been received for this address-as-  
1849 inferior and identifier, or if RESIGN has been received from this Inferior

1850  
1851  
1852 The form HAZARD/mixed refers to a HAZARD message with “level” = “mixed”, the form  
1853 HAZARD/possible refers to a HAZARD message with “level” = “possible”.

1854  
1855 **CONTRADICTION**

1856  
1857 Sent by the Superior to an Inferior that has taken an autonomous decision contrary to the  
1858 decision for the atom. This is detected by the Superior when the ‘wrong’ one of  
1859 CONFIRMED or CANCELLED is received. CONTRADICTION is also sent in response to a  
1860 HAZARD message.  
1861

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

1862  
1863 **target address** the address to which the CONTRADICTION message is sent.  
1864 This will be the address-as-inferior from the ENROL message.

1865  
1866 **inferior identifier** The inferior identifier as on the earlier ENROL message for  
1867 this Inferior.

1868  
1869 **qualifiers** standardised or other qualifiers.

1870  
1871 Types of FAULT possible (sent to Superior address)

1872  
1873 **General**  
1874 **InvalidInferior** – if inferior identifier is unknown  
1875 **WrongState** – if neither CONFIRMED or CANCELLED has been sent  
1876 by this Inferior

1877  
1878 **SUPERIOR\_STATE**

1879  
1880 Sent by a Superior as a query to an Inferior when  
1881

1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889

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

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

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

1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899

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

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

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

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

1900  
1901  
1902  
1903  
1904

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

1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938

**qualifiers** standardised or other qualifiers.

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

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

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

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

## INFERIOR\_STATE

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

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

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

1939 **target address** the address to which the INFERIOR\_STATE is sent. This will  
1940 be the target address as used the original ENROL message.

1941  
1942 **superior identifier** The superior identifier as used on the ENROL message

1943  
1944 ~~address as inferior~~ ~~The address as inferior as on the ENROL message (with the~~  
1945 ~~inferior identifier, this determines who the message is from)~~

1946  
1947 **inferior identifier** The inferior identifier as on the ENROL message

1948  
1949 **status** states the current state of the Inferior for the atomic business transaction,  
1950 which corresponds to the last message sent to the Superior by (or in the case of  
1951 ENROL for) the Inferior

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

1953  
1954 **reply requested** “true” if INFERIOR\_STATE is sent as a query at the  
1955 Superior’s initiative; “false” if INFERIOR\_STATE is sent in reply to a received  
1956 SUPERIOR\_STATE or other message. Can only be “true” if “status” is “active”  
1957 or “prepared-received”. Can only be “true” if “status” is “active”.

1958  
1959 **qualifiers** standardised or other qualifiers.

1960  
1961 The Superior, on receiving INFERIOR\_STATE with “reply requested” = “true”, should reply  
1962 in a timely manner by (depending on its state) repeating the previous message it sent or by  
1963 sending SUPERIOR\_STATE with the appropriate status value.

1964  
1965 A status of “unknown” shall only be sent if it has been determined for certain that the Inferior  
1966 has no knowledge of a relationship with the Superior. If there could be persistent information  
1967 corresponding to the Superior, but it is not accessible from the entity receiving an  
1968 SUPERIOR\_STATE/\*y (or other) message targetted on the Inferior or the entity cannot  
1969 determine whether any such persistent information exists, the response shall be  
1970 “inaccessible”.

1971  
1972 INFERIOR\_STATE/unknown is also used as a response to messages, other than  
1973 SUPERIOR\_STATE/\*y that are received when the Inferior is not known (and it is known  
1974 there is no state information for it).

1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997

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

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

## REDIRECT

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

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

1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011

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

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

**inferior identifier** The inferior identifier as on the ENROL message

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

2012 **new address** The (set of alternatives) new addresses to be used for messages  
 2013 sent to this entity.  
 2014  
 2015 **qualifiers** standardised or other qualifiers.  
 2016  
 2017 If the actor whose address is changed is an Inferior, the new address value  
 2018 replaces the address-as-inferior as present in the ENROL.  
 2019  
 2020 If the actor whose address is changed is a Superior, the new address value  
 2021 replaces the Superior address as present in the CONTEXT message (or as present  
 2022 in any other mechanism used to establish the Superior:Inferior relationship).  
 2023  
 2024  
 2025

## 2026 Messages used in control relationships

### 2027 BEGIN

2028 A request to a Factory to create a new Business Transaction. This may either be a new top-  
 2029 level transaction, in which case the Composer or Coordinator will be the Decider, or the new  
 2030 Business Transaction may be immediately made the Inferior within an existing Business  
 2031 Transaction (thus creating a sub-Composer or sub-Coordinator).  
 2032  
 2033  
 2034

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

2035  
 2036 **target address** the address of the entity to which the BEGIN is sent. How this  
 2037 address is acquired and the nature of the entity are outside the scope of this  
 2038 specification.  
 2039  
 2040 **reply address** the address to which the replying BEGUN and related  
 2041 CONTEXT message should be sent.  
 2042  
 2043 **transaction type** identifies whether a new Cohesion or new Atom is to be  
 2044 created; this value will be the “superior type” in the new CONTEXT  
 2045  
 2046 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction  
 2047 timelimit” may be present on BEGIN, to set the timelimit for the new business  
 2048 transaction and will be copied to the new CONTEXT. The standard qualifier  
 2049 “Inferior name” may be present if there is a CONTEXT related to the BEGIN.  
 2050

2051 A new top-level Business Transaction is created if there is no CONTEXT related to the  
2052 BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is  
2053 created if the CONTEXT message for the existing Business Transaction is related to the  
2054 BEGIN. In this case, the Factory is responsible for enrolling the new Composer or  
2055 Coordinator as an Inferior of the Superior identified in that CONTEXT.  
2056

---

2057 Note – This specification does not provide a standardised means to  
2058 determine which of the Inferiors of a sub-Composer are in its confirm set.  
2059 This is considered part of the application:inferior relationship.

---

2060 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction type” having  
2061 the corresponding value.  
2062

2063 Types of FAULT possible (sent to Reply address)  
2064

### 2065 General

### 2066 BEGUN

2067 BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT  
2068 for the new business transaction.  
2069  
2070  
2071  
2072

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

2073  
2074 **target address** the address to which the BEGUN is sent. This will be the reply  
2075 address from the BEGIN.  
2076

2077 **address-as-decider** for a top-level transaction (no CONTEXT related to the  
2078 BEGIN), this is the address to which PREPARE\_INFERIORS,  
2079 CONFIRM\_TRANSACTION, CANCEL\_TRANSACTION,  
2080 CANCEL\_INFERIORS and REQUEST\_INFERIOR\_STATUSES messages are  
2081 to be sent; if a CONTEXT was related to the BEGIN this parameter is absent  
2082

2083 **transaction-identifier** identifies the new Decider (Composer or Coordinator)  
2084 within the scope of the address-as-decider. If this is not a top-level transaction,  
2085 the transaction-identifier is optional, but if present shall be the inferior-identifier

2086 used in the enrolment with the Superior identified by the CONTEXT related to  
2087 the BEGIN.

2088  
2089 **inferior handle** Shall be absent if this is a top-level transaction and may or may  
2090 not be present otherwise. (Presence or absence will be determined by the nature  
2091 of the Superior identified in the CONTEXT related to the BEGIN). If present, the  
2092 inferior handle will identify this new business transaction as in the inferiors-list  
2093 parameters in messages between the Superior identified in the CONTEXT related  
2094 to the BEGIN (acting as a Decider) and its Terminator. The value shall be  
2095 different for each enrolled Inferior of that Superior.

2096  
2097 **address-as-inferior** This parameter shall be absent if this is a top-level  
2098 transaction and may be present, at implementation option otherwise. If present, it  
2099 shall be the address-as-inferior used in the enrolment with the Superior identified  
2100 by the CONTEXT related to the BEGIN. If this is a top-level transaction

2101  
2102 **qualifiers** standardised or other qualifiers.

2103  
2104 At implementation option, the “address-as-decider” and/or “address-as-inferior” and the  
2105 “address-as-superior” in the related CONTEXT may be the same or may be different. There  
2106 is no general requirement that they even use the same bindings. Any may also be the same as  
2107 the target address of the BEGIN message (the inferior identifier on messages will ensure they  
2108 are applied to the appropriate Composer or Coordinator).

2109  
2110 No FAULT messages are issued on receiving BEGUN.

2111

## 2112 PREPARE\_INFERIORS

2113

2114 Sent from a Terminator to a Decider, but only if it is a Cohesion Composer, to tell it to  
2115 prepare all or some of its inferiors, by sending PREPARE to any that have not already sent  
2116 PREPARED, RESIGN or CANCELLED to the Decider (Composer) on its relationships as  
2117 Superior. If the inferiors-list parameter is absent, the request applies to all the inferiors; if the  
2118 parameter is present, it applies only to the identified inferiors of the Decider (Composer).

2119

Parameter	Type
target address	BTP address
reply address	BTP address
transaction-identifier	Identifier
inferiors-list	List of <a href="#">Identifiers</a> <a href="#">inferior handles</a>
qualifiers	List of qualifiers

2120

2121 **target address** the address to which the PREPARE\_INFERIORS message is  
2122 sent. This will be the decider-address from the BEGUN message.

2123

2124 **reply address** the address of the Terminator sending the  
 2125 PREPARE\_INFERIORS message.  
 2126  
 2127 **transaction identifier** identifies the Decider and will be the transaction-identifier  
 2128 from the BEGUN message.  
 2129  
 2130 **inferiors-list** defines which of the Inferiors of this Decider preparation is  
 2131 requested for, using the “inferior-identifiers” as on the ENROL received by the  
 2132 Decider (in its role as Superior). If this parameter is absent, the PREPARE  
 2133 applies to all Inferiors.  
 2134  
 2135 **qualifiers** standardised or other qualifiers.  
 2136

2137  
 2138 For all Inferiors identified in the inferiors-list parameter (all Inferiors if the parameter is  
 2139 absent), from which none of PREPARED, CANCELLED or RESIGNED has been received,  
 2140 the Decider shall issue PREPARE. It will reply to the Terminator, using the reply address on  
 2141 the PREPARE\_INFERIORS message, sending an INFERIOR\_STATUSES message giving  
 2142 the status of the Inferiors identified on the inferiors-list parameter (all of them if the  
 2143 parameter was absent).  
 2144

2145 Types of FAULT possible (sent to Superior address)

2146  
 2147 *General*  
 2148 *InvalidDecider* – if Decider address is unknown  
 2149 *UnknownTransaction* – if the transaction-identifier is unknown  
 2150 *InvalidInferior* – if an inferior-handle on the inferiors-list is unknown  
 2151 *WrongState* – if a CONFIRM\_TRANSACTION or  
 2152 CANCEL\_TRANSACTION has already been received by this  
 2153 Composer.  
 2154

2155 The form PREPARE\_INFERIORS/all refers to a PREPARE\_INFERIORS message where  
 2156 the “inferiors-list” parameter is absent. The form PREPARE\_INFERIORS/specific refers to a  
 2157 PREPARE\_INFERIORS message where the “inferiors-list” parameter is present.  
 2158

2159  
 2160  
 2161 **CONFIRM\_TRANSACTION**

2162  
 2163 Sent from a Terminator to a Decider to request confirmation of the business transaction. If the  
 2164 business transaction is a Cohesion, the confirm-set is specified by the “inferiors-list”  
 2165 parameter.  
 2166

Parameter	Type
target address	BTP address

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

2167

2168

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

2169

2170

2171

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

2172

2173

2174

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

2175

2176

2177

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

2178

2179

2180

2181

2182

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

2183

2184

2185

2186

2187

2188

2189

**qualifiers** standardised or other qualifiers.

2190

2191

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

2192

2193

2194

2195

2196

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

2197

2198

2199

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

2200

2201

2202

---

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

2203

2204

2205 send PREPARE if there are indications that the earlier PREPARE was not  
2206 delivered.

---

2207

2208

2209 A confirm decision may be made only if PREPARED has been received from all Inferiors in  
2210 the “confirm-set”. The making of the decision shall be persistent (and if it is not possible to  
2211 persist the decision, it is not made). If there is only one remaining Inferior in the “confirm  
2212 set” and PREPARE has not been sent to it, CONFIRM\_ONE\_PHASE may be sent to it.

2213

2214 All remaining Inferiors that are not in the confirm set shall be cancelled.

2215

2216 If a confirm decision is made and “report-hazard” was “false”, a CONFIRM\_COMPLETE  
2217 message shall be sent to the “reply-address”.

2218

2219 If a cancel decision is made and “report-hazard” was “false”, a CANCEL\_COMPLETE  
2220 message shall be sent to the “reply-address”.

2221

2222 If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.  
2223 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in  
2224 the confirm-set), an INFERIOR\_STATUSES reporting the status for all Inferiors shall be sent  
2225 to the “reply-address”.

2226

2227 Types of FAULT possible (sent to reply address)

2228

2229

**General**

2230

**InvalidDecider** – if Decider address is unknown

2231

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

2232

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

2233

**WrongState** – if a CANCEL\_TRANSACTION has already been

2234

received .

2235

2236 The form CONFIRM\_TRANSACTION/all refers to a CONFIRM\_TRANSACTION message  
2237 where the “inferiors-list” parameter is absent. The form

2238

CONFIRM\_TRANSACTION/specific refers to a CONFIRM\_TRANSACTION message

2239

where the “inferiors-list” parameter is present.

2240

2241

**TRANSACTION\_CONFIRMED**

2242

2243 A Decider sends TRANSACTION\_CONFIRMED to a Terminator in reply to  
2244 CONFIRM\_TRANSACTION if all of the confirm-set confirms (and, for a Cohesion, all other  
2245 Inferiors cancel) without reporting hazards, or if the Decider made a confirm decision and the  
2246 CONFIRM\_TRANSACTION had a “report-hazards” value of “false”.

2247

**Parameter**

**Type**

target address

BTP address

Parameter	Type
<del>address-as-decider</del>	<del>BTP address</del>
transaction-identifier	identifier
qualifiers	List of qualifiers

2248

2249

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

2250

2251

2252

2253

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

2254

2255

2256

2257

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

2258

2259

2260

**qualifiers** standardised or other qualifiers.

2261

2262

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

2263

2264

#### *General*

2265

*InvalidTerminator* – if Terminator address is unknown

2266

*UnknownTransaction* – if the transaction-identifier is unknown

2267

2268

### CANCEL\_TRANSACTION

2269

2270

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

2271

2272

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

2273

2274

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

2275

2276

2277

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

2278

2279

2280 **transaction identifier** identifies the Decider and will be the transaction-identifier  
2281 from the BEGUN message.

2282  
2283 **report hazard** Defines whether the Terminator wishes to be informed of hazard  
2284 events and contradictory decisions within the business transaction. If “report  
2285 hazard” is “true”, the receiver will wait until responses (CONFIRMED,  
2286 CANCELLED or HAZARD) have been received from all of its inferiors,  
2287 ensuring that any hazard events are reported. If “report hazard” is “false”, the  
2288 Decider will reply with TRANSACTION\_CANCELLED immediately.

2289  
2290 **qualifiers** standardised or other qualifiers.

2291  
2292 The business transaction is cancelled – this is propagated to any remaining Inferiors by  
2293 issuing CANCEL to them. No more Inferiors will be permitted to enrol.

2294  
2295 Types of FAULT possible (sent to Superior address)

2296  
2297 *General*  
2298 *InvalidDecider* – if Decider address is unknown  
2299 *UnknownTransaction* – if the transaction-identifier is unknown  
2300 *WrongState* – if a CONFIRM\_TRANSACTION has been received by  
2301 this Composer.

2302  
2303

## 2304 CANCEL\_INFERIORS

2305  
2306 Sent by a Terminator to a Decider, but only if is a Cohesion Composer, at any time before  
2307 CONFIRM\_TRANSACTION or CANCEL\_TRANSACTION has been sent.

2308

Parameter	Type
target address	BTP address
reply address	BTP address
transaction identifier	Identifier
inferiors-list	List of inferior handles <a href="#">Identifiers</a>
qualifiers	List of qualifiers

2309  
2310 **target address** the address to which the CANCEL\_TRANSACTION message is  
2311 sent. This will be the decider-address from the BEGUN message.

2312  
2313 **reply address** the address of the Terminator sending the  
2314 CANCEL\_TRANSACTION message.

2315  
2316 **transaction identifier** identifies the Decider and will be the transaction-identifier  
2317 from the BEGUN message.

2318  
2319  
2320  
2321  
2322  
2323  
2324  
2325  
2326  
2327  
2328

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

**qualifiers** standardised or other qualifiers.

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

2329  
2330  
2331

---

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

---

2332  
2333  
2334  
2335  
2336  
2337  
2338  
2339  
2340  
2341  
2342  
2343

Types of FAULT possible (sent to Superior address)

**General**

**InvalidDecider** – if Decider address is unknown

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

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

**WrongState** – if a CONFIRM\_TRANSACTION or

CANCEL\_TRANSACTION has been received by this Composer.

2344  
2345  
2346  
2347  
2348  
2349  
2350  
2351

**TRANSACTION\_CANCELLED**

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

**Parameter**

target address	BTP address
<del>address-as-decider</del>	<del>BTP-address</del>
transaction-identifier	identifier
qualifiers	List of qualifiers

2352

2353 **target address** the address to which the TRANSACTION\_CANCELLED is  
 2354 sent. This will be the reply address from the CANCEL\_TRANSACTION or  
 2355 CONFIRM\_TRANSACTION message.  
 2356  
 2357 ~~address-as-decider the address as decider of the Decider as on the BEGUN~~  
 2358 ~~message (with the transaction identifier, this determines who the message is~~  
 2359 ~~from).~~  
 2360  
 2361 **transaction identifier** the transaction identifier as on the BEGUN message (i.e.  
 2362 the identifier of the Decider as a whole).  
 2363  
 2364 **qualifiers** standardised or other qualifiers.

2365 Types of FAULT possible (sent to address-as-decider)

2366  
 2367  
 2368 *General*

2369 *InvalidTerminator* – if Terminator address is unknown

2370 *UnknownTransaction* – if the transaction-identifier is unknown

2371  
 2372  
 2373  
 2374 **REQUEST\_INFERIOR\_STATUSES**

2375  
 2376 Sent to a Decider to ask it to report the status of its Inferiors with an INFERIOR\_STATUSES  
 2377 message. It can also be sent to any actor with an address-as-superior or address-as-inferior,  
 2378 asking it about the status of that transaction tree nodes Inferiors, if there are any. In this latter  
 2379 case, the receiver may reject the request with a FAULT(StatusRefused). If it is prepared to  
 2380 reply, but has no Inferiors, it replies with an INFERIOR\_STATUSES with an empty “status-  
 2381 list” parameter.  
 2382

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

2383  
 2384 **target address** the address to which the REQUEST\_STATUS message is sent.  
 2385 When used to a Decider, this will be the address-as-decider from the BEGUN  
 2386 message. Otherwise it may be an address-as-superior from a CONTEXT or  
 2387 address-as-inferior from an ENROL message.  
 2388  
 2389 **reply address** the address to which the replying INFERIOR\_STATUSES is to  
 2390 be sent



2428 **target address** the address to which the INFERIOR\_STATUSES is sent. This  
 2429 will be the reply address on the received message  
 2430  
 2431 ~~**responders-address** If the sender is a Decider, the address as decider as on the~~  
 2432 ~~BEGUN message. Otherwise the address of the sender of this message— one of~~  
 2433 ~~address as inferior, address as superior. With the responders identifier, this~~  
 2434 ~~determines who the message is from.~~  
 2435  
 2436 **responders-identifier** ~~If the sender is a Decider, the transaction identifier as on~~  
 2437 ~~the BEGUN message. Otherwise,~~ the target-identifier used on the  
 2438 REQUEST\_INFERIOR\_STATUSES.  
 2439  
 2440 **status-list** contains a number of Status-items, each reporting the status of one of  
 2441 the inferiors of the Decider. The fields of a Status-item are  
 2442

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

2443  
 2444 The status value reports the current status of the particular inferior, as known to  
 2445 the Decider (Composer or Coordinator). Values are:  
 2446

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

status value	Meaning
<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)

2447

2448

2449

2450

2451

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

2452

2453

2454

2455

2456

2457

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

2458

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

2459

2460

**General**

2461

**InvalidTerminator** – if Terminator address is unknown

2462

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

2463

2464

2465

2466

2467

**Groups – combinations of related messages**

2468

2469

2470

2471

2472

2473

2474

2475

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.

2476  
2477  
2478  
2479  
2480  
2481  
2482  
2483  
2484  
2485  
2486  
2487  
2488  
2489  
2490  
2491  
2492  
2493  
2494  
2495  
2496  
2497  
2498  
2499  
2500  
2501  
2502

## CONTEXT & application message

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

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

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

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

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

---

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

---

2508  
2509  
2510  
2511  
2512  
2513  
2514  
2515  
2516  
2517  
2518  
2519  
2520

## CONTEXT\_REPLY & ENROL

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

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

The target address of the ENROL message is omitted.

2521  
2522 The actor receiving the related group will use the retained Superior address from the  
2523 CONTEXT sent earlier to forward the ENROL. When doing so, it changes the ENROL to  
2524 ask for a response (if it was an ENROL/no-rsp-req) and supplies its own address as the  
2525 “reply-address”, remembering the original “reply-address” if there was one.  
2526

2527 If ENROLLED is received and the original received ENROL was ENROL/rsp-req, the  
2528 ENROLLED is forwarded back to the original “reply-address”.  
2529

2530 If this attempt fails (i.e. ENROLLED is not received), and the “completion-status” of the  
2531 CONTEXT\_REPLY was “related”, the actor is required to ensure that the Superior does  
2532 not proceed to confirmation. How this is achieved is an implementation option, but must  
2533 take account of the possibility that direct communication with the Superior may fail. (One  
2534 method is to prevent CONFIRM\_TRANSACTION being sent to the Superior (in its role  
2535 as Decider); another is to enrol as another Inferior before sending the original CONTEXT  
2536 out with an application message). If the Superior is a sub-coordinator or sub-composer,  
2537 an enrolment failure must ensure the sub-coordinator does not send PREPARED to its  
2538 own Superior.  
2539

2540 If the actor receiving the related group is also the Superior (i.e. it has the same binding  
2541 address), the explicit forwarding of the ENROL is not required, but the resultant effect –  
2542 that if enrolment fails the Superior does not confirm or issue PREPARED – shall be the  
2543 same.  
2544

2545 A CONTEXT\_REPLY & ENROL group may contain multiple ENROL messages, for  
2546 several Inferiors. Each ENROL shall be forwarded and an ENROLLED reply received  
2547 before the Superior is allowed to confirm if the “completion-status” in the  
2548 CONTEXT\_REPLY was “related”.  
2549

2550 When the group is constructed, if the CONTEXT had “superior-type” value of “atom”,  
2551 the “completion-status” of the CONTEXT\_REPLY shall be “related”. If the “superior-  
2552 type” was “cohesive”, the “completion-status” shall be “completed” or “related” (as  
2553 required by the application). If the value is “completed”, the actor receiving the group  
2554 shall forward the ENROLs, but is not required to (though it may) prevent confirmation.  
2555

## 2556 **CONTEXT\_REPLY (& ENROL) & PREPARED / & CANCELLED**

2557  
2558 This combination is characterised by a related CONTEXT\_REPLY and either or both of  
2559 PREPARED and CANCELLED, with or without ENROL.  
2560

2561 **Meaning:** If ENROL is present, the meaning and required processing is the same as for  
2562 CONTEXT\_REPLY & ENROL. The PREPARED or CANCELLED message(s) are  
2563 forwarded to the Superior identified in the CONTEXT message this CONTEXT\_REPLY  
2564 is replying to.  
2565

2566  
2567

---

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

---

2568  
2569  
2570  
2571  
2572

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

2573  
2574  
2575

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

2576  
2577  
2578  
2579

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

2580  
2581  
2582  
2583  
2584

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

2585  
2586  
2587  
2588  
2589  
2590  
2591  
2592

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

2593  
2594

### CONTEXT\_REPLY & ENROL & application message (& PREPARED)

2595  
2596  
2597  
2598

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

2599  
2600  
2601  
2602

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

2603  
2604  
2605  
2606  
2607

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

2608  
2609  
2610

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

2611  
2612 The processing of ENROL and PREPARED messages is the same as for the previous  
2613 groups.  
2614  
2615 This group can be used when participation in business transaction (normally a cohesion),  
2616 is initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with  
2617 some associated application semantic, performs some work for the transaction and sends  
2618 an application message with a related ENROL. The CONTEXT\_REPLY allows the  
2619 addressing of the application (and the CONTEXT\_REPLY) to be distinct from that of the  
2620 Superior.

2621  
2622 The actor receiving the group may associate the “inferior-~~handle~~identifier” received on  
2623 the ENROL-~~LED~~ with the application message in a manner that is visible to the  
2624 application receiving the message (e.g. for subsequent use in Terminator:Decider  
2625 exchanges).

## 2626 **BEGUN & CONTEXT**

2627  
2628  
2629 **Meaning:** the CONTEXT is that for the new business transaction, containing the  
2630 Superior address.

2631  
2632 **Target address:** the target address is that of the BEGUN message – this will be the reply  
2633 address of the earlier BEGIN message.

## 2634 **BEGIN & CONTEXT**

2635  
2636  
2637 **Meaning:** the new business transaction is to be an Inferior (sub-coordinator or sub-  
2638 composer) of the Superior identified by the CONTEXT. The Factory (receiver of the  
2639 BEGIN) will perform the enrolment.

2640  
2641 **Target address:** the target address is that of the BEGIN – this will be the address of the  
2642 Factory.

## 2643 **Standard qualifiers**

2644  
2645  
2646 The following qualifiers are expected to be of general use to many applications and  
2647 environments. The URI “urn:oasis:names:tc:BTP:qualifiers” is used in the  
2648 Qualifier group value for the qualifiers defined here.

## 2649 **Transaction timelimit**

2650  
2651  
2652  
2653 The transaction timelimit allows the Superior (or an application element initiating the  
2654 business transaction) to indicate the expected length of the active phase, and thus give an  
2655 indication to the Inferior of when it would be appropriate to initiate cancellation if the active  
2656 phase appears to continue too long. The time limit ends (the clock stops) when the Inferior  
2657 decides to be prepared and issues PREPARED to the Superior.

2658  
2659 It should be noted that the expiry of the time limit does not change the permissible actions of  
2660 the Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is  
2661 **permitted** to initiate cancellation for internal reasons. The `timelimit` gives an indication to the  
2662 entity of when it will be useful to exercise this right.

2663  
2664 The qualifier is propagated on a `CONTEXT` message.

2665  
2666 The “Qualifier name” shall be “`transaction-timelimit`”.

2667  
2668 The “Content” shall contain the following field:

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

2670  
2671 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the  
2672 time of transmission of the containing `CONTEXT`, of the active phase of the business  
2673 transaction.

#### 2674 2675 **Inferior timeout**

2676  
2677 This qualifier allows an Inferior to limit the duration of its “promise”, when sending  
2678 `PREPARED`, that it will maintain the ability to confirm or cancel the effects of all associated  
2679 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or  
2680 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and  
2681 can apply the decision indicated in the qualifier.

2682  
2683 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply  
2684 a confirm or cancel decision before the `CONFIRM` or `CANCEL` is received and before this  
2685 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,  
2686 and (as with other transaction mechanisms), is considered to be an exceptional event. As with  
2687 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the  
2688 expiry of this timeout, is liable to cause contradictory decisions across the business  
2689 transaction. BTP ensures that at least the occurrence of such a contradiction will be  
2690 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic  
2691 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this  
2692 timeout does not cause a qualitative (state table) change in what can happen, but rather a step  
2693 change in the probability that it will.

2694  
2695 The expiry of the timeout does not strictly require that the Inferior immediately invokes the  
2696 intended decision, only that is at liberty to do so. An implementation may choose to only  
2697 apply the decision if there is contention for the underlying resource, for example.  
2698 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for  
2699 the business transaction are made before these timeouts expire (and allow a margin of error  
2700 for network latency etc.).

2701

2702 The qualifier may be present on a PREPARED message. If the PREPARED message has the  
2703 “default is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall  
2704 have the value “cancel”.

2705  
2706 The “Qualifier name” shall be “inferior-timeout”.

2707  
2708 The “Content” shall contain the following fields:  
2709

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

2710  
2711 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the  
2712 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the  
2713 effects of the associated operations, as ordered by the receiving Superior.

2714  
2715 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an  
2716 autonomous decision is made.

2717  
2718 **Minimum inferior timeout**

2719  
2720 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the  
2721 Inferior. If a Superior knows that the decision for the business transaction will not be  
2722 determined for some period, it can require that Inferiors do not send PREPARED messages  
2723 with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to  
2724 send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with  
2725 CANCELLED.

2726  
2727 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If  
2728 present on more than one, and with different values of the MinimumTimeout field, the value  
2729 on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall  
2730 prevail over either of the others.

2731  
2732 The “Qualifier name” shall be “minimum-inferior-timeout”.

2733  
2734 The “Content” shall contain the following field:  
2735

Content field	Type
MinimumTimeout	Integer

2736  
2737 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be  
2738 acceptable in the Inferior timeout qualifier on an answering PREPARED message.

2739  
2740 **Inferior name**

2741

2742 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on  
2743 INFERIOR\_STATUSES and thus allow the Terminator to determine which Inferior (of the  
2744 Composer or Coordinator) is related to which application work. This is in addition to the  
2745 “inferior-~~identifier-handle~~” field. The name can be human-readable and can also be used in  
2746 fault tracing, debugging and auditing.

2747  
2748 The name is never used by the BTP actors themselves to identify each other or to direct  
2749 messages. (The BTP actors use the addresses and the identifiers in the message parameters  
2750 for those purposes.)

2751  
2752 This specification makes no requirement that the names are unambiguous within any scope  
2753 (unlike the ~~globally unambiguous~~ “inferior-~~handle~~identifier” on ENROLLED and BEGUN,  
2754 ~~which is required to be unambiguous within the scope of the Decider~~). Other specifications,  
2755 including those defining use of BTP with a particular application may place requirements on  
2756 the use and form of the names. (This may include reference to information passed in  
2757 application messages or in other, non-standardised, qualifiers.)

2758  
2759 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item  
2760 in INFERIOR\_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if  
2761 present, the same qualifier value **should** be included in the consequent ENROL. If  
2762 INFERIOR\_STATUSES includes a Status-item for an Inferior whose ENROL had an  
2763 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.

2764  
2765 The “Qualifier -name” shall be “inferior-name”

2766  
2767 The “Content” shall contain the following fields:

Content field	Type
inferior-name	String

2769  
2770 **Inferior name** the name assigned to the enrolling Inferior.  
2771

2772  
2773  
2774  
2775  
2776  
2777  
2778  
2779  
2780  
2781  
2782  
2783  
2784  
2785  
2786  
2787  
2788  
2789  
2790  
2791  
2792  
2793  
2794  
2795  
2796  
2797  
2798  
2799  
2800  
2801  
2802  
2803  
2804  
2805  
2806  
2807  
2808  
2809  
2810  
2811  
2812  
2813  
2814  
2815  
2816  
2817

## State Tables

### Explanation of the state tables

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

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

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

### Status queries

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

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

### Decision events

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

2818 business transaction and on features of the implementation (e.g. making of a persistent record  
2819 of the decision means that the information will survive at least some failures that otherwise  
2820 lose state information, but the level of survival depends on the purpose of the  
2821 implementation). [Table 2](#) and [Table 3](#) define the decision events.

2822  
2823 In some cases, an implementation may not need to make an active change to have a persistent  
2824 record of a decision, provided that the implementation will restore itself to the appropriate  
2825 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,  
2826 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via  
2827 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a  
2828 record of “decide to cancel autonomously”, provided it always updated such a record as part  
2829 of the “apply ordered confirmation” decision event.

2830  
2831 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of  
2832 PREPARE indicates that the application exchange (to associate operations with the Inferior)  
2833 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier  
2834 state corresponding to an incomplete application exchange. However, implementations are  
2835 not required to make the sending of PREPARE persistent in terms of recovery – a Superior  
2836 that experiences failure after sending PREPARE may, on recovery, have no information  
2837 about the transaction, in which case it is considered to be in the completed state (Z), which  
2838 will imply the cancellation of the Inferior and its associated operations.

2839  
2840 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its  
2841 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a  
2842 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local  
2843 persistent information (which would combine both superior and inferior information, pointing  
2844 both up and down the tree).

2845  
2846

### 2847 **Disruptions – failure events**

2848  
2849 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or  
2850 may) cause a change of state. The disruption events in the state tables model different extents  
2851 of loss of state information. An implementation is not required to exhibit all the possible  
2852 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a  
2853 possible disruption.

2854  
2855 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,  
2856 which involves possible interruption of service and loss of messages in transit, but no change  
2857 of state (either because no state information was lost, or because recovery from persistent  
2858 information restores the implementation to the same state). The “disruption 0” event would  
2859 typically be an appropriate abstraction for a communication failure.

2860  
2861

### 2861 **Invalid cells and assumptions of the communication mechanism**

2862  
2863 The empty cells in state table represent events that cannot happen. For events corresponding  
2864 to sending a message or any of the decision events, this prohibition is absolute – e.g. a

2865 conformant implementation in the Superior active state “B1” will not send CONFIRM. For  
2866 events corresponding to receiving a message, the interpretation depends on the properties of  
2867 the underlying communications mechanism.

2868

2869 For all communication mechanisms, it is assumed that

- 2870 a) the two directions of the Superior:Inferior communication are not synchronised –
- 2871 that is messages travelling in opposite directions can cross each other to any
- 2872 degree; any number of messages may be in transit in either direction; and
- 2873 b) messages may be lost arbitrarily

2874

2875 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered  
2876 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a  
2877 state where the corresponding cell is empty indicates that the far-side has sent a message out  
2878 of order – a FAULT message with the Fault Type “WrongState” can be returned.

2879

2880 If the communication mechanisms cannot guarantee ordered delivery, then messages received  
2881 where the corresponding cell is empty should be ignored. Assuming the far-side is  
2882 conformant, these messages can assumed to be “stale” and have been overtaken by messages  
2883 sent later but already delivered. (If the far-side is non-conformant, there is a problem  
2884 anyway).

2885

## 2886 **Meaning of state table events**

2887

2888 The tables in this section define the events (rows) in the state tables. [Table 1](#) defines  
2889 the events corresponding to sending or receiving BTP messages and the disruption events.  
2890 [Table 2](#) describes the decision events for an Inferior, [Table 3](#) those for a  
2891 Superior.

2892

2893 The decision events for a Superior, defined in [Table 3](#) cannot be specified without  
2894 reference to other Inferiors to which it is Superior and to its relation with the application or  
2895 other entity that (acting ultimately on behalf of the application) drives it.

2896

2897 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and  
2898 which are to be treated as an atomic decision unit with (and thus including) the Inferior on  
2899 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior  
2900 type” of “atom”, this will be all Inferiors established with same Superior address and Superior  
2901 identifier except those from which RESIGN has been received. If the CONTEXT had  
2902 “superior type” of “cohesion”, the “remaining Inferiors” excludes any that it has been  
2903 determined will be cancelled, as well as any that have resigned – in other words it includes  
2904 only those for which a confirm decision is still possible or has been made. The determination  
2905 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some  
2906 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this  
2907 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.

2908

2909 In order to ensure that the confirmation decision is delivered to all remaining Inferiors,  
2910 despite failures, the Superior must persistently record which these Inferiors are (i.e. their  
2911 addresses and identifiers). It must also either record that the decision is confirm, or ensure

2912 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it  
 2913 will be told about it. This latter would apply if the Superior were also BTP Inferior to another  
 2914 entity which persisted a confirm decision (or recursively deferred it still higher). However,  
 2915 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the  
 2916 behaviour of asking another entity to make (and persist) the confirm decision is termed  
 2917 "offering confirmation" - the Superior offers the possible confirmation of itself, and its  
 2918 remaining Inferiors to some other entity. If that entity (or something higher up) then does  
 2919 make and persist a confirm decision, the Superior is "instructed to confirm" (which is  
 2920 equivalent BTP CONFIRM).

2921  
 2922 The application, or an entity acting indirectly on behalf of the application, may request a  
 2923 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no  
 2924 more operations associated with the Inferior. Following a request to prepare all remaining  
 2925 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the  
 2926 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the  
 2927 application.)

2928  
 2929 The application, or an entity acting indirectly on behalf of the application, may also request  
 2930 confirmation. This means the Superior is to attempt to make and persist a confirm decision  
 2931 itself, rather than offer confirmation.

2932  
 2933

2934

**Table 1 : send, receive and disruption events**

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

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

2935

2936

**Table 2 : 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>
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>

Event name	Meaning
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>

2937

2938

**Table 3: 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> <li>○ Superior has been requested to confirm; AND</li> <li>○ persistent information records the confirm decision and identifies all remaining Inferiors;</li> </ul> </li> <li>• Or <ul style="list-style-type: none"> <li>○ persistent information records an offer of confirmation and has been instructed to confirm</li> </ul> </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> </ul>

Event name	Meaning
	<ul style="list-style-type: none"> <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>

2939  
2940  
2941  
2942  
2943  
2944  
2945  
2946  
2947  
2948  
2949  
2950  
2951  
2952  
2953  
2954  
2955  
2956  
2957  
2958  
2959  
2960  
2961  
2962  
2963  
2964  
2965  
2966  
2967

### Persistent information

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

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

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

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

**Table 4 : Superior states**

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

**Table 5 : Inferior states**

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

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

2971  
2972

**Table 6: Superior state table – normal forward progression**

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

**Table 7: Superior state table – cancellation and contradiction**

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

**Table 8: Superior state table – hazard and request confirm**

	P1	P2	P3	P4	Q1	R1	R2	S1
recei ve ENROL/rsp-req								
recei ve ENROL/no-rsp-req								
recei ve RESI GN/rsp-req								C1
recei ve RESI GN/no-rsp-req								Z
recei ve PREPARED								S1
recei ve PREPARED/cancel								S1
recei ve CONFIR MED/auto					Q1	R1	R1	S1
recei ve CONFIR MED/response					Z	R2		Z
recei ve CANCELLED						R1	R1	Z
recei ve HAZARD	P1	P2	P3	P4		R1	R1	Z
recei ve INF_STATE/acti ve/y								S1
recei ve INF_STATE/acti ve								S1
recei ve INF_STATE/unknown	P1	P2		P4		R2	R2	Z
send ENROLLED								
send RESI GNED								
send PREPARE								
send CONFIR M_ONE_PHASE								S1
send CONFIR M								
send CANCEL								
send CONTRADI CTI ON						R2		
send SUP_STATE/acti ve/y								
send SUP_STATE/acti ve								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
deci de to confi rm one-phase								
deci de to prepare								
deci de to confi rm								
deci de to cancel								
remove persi stent i nformati on							Z	
record contradi cti on	R1	R1	R1	R1	R1			
di srupti on I	Z	Z	Z	Z	Z		R1	Z
di srupti on II	D1		F1	G2				
di srupti on III	B1							
di srupti on IV								

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

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

2976

2977

2977

2978

**Table 10: Inferior state table – normal forward progression**

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

2979

2980

**Table 11: Inferior state table – cancellation and contradiction**

	g1	g2	h1	h2	j1	j2	k1	k2	l1	l2
send ENROL/rsp-req send ENROL/no-rsp-req send 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			h1		j1		k1		l1	
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	g1	g2	h1 s3 h2 h2 l1 l2		j1 s4 k1 j2 j2 k2		k1 k2 k2		l1 l2 l2	
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	x2	h1 h1 h1 h1 l1		j1 j1 j1 j1 j2 j2		k2 k2		l1	
deci de to resi gn deci de to be prepared deci de to be prepared/cancel deci de to confi rm autonomously deci de to cancel autonomously apply ordered confi rmati on remove persi stent i nformati on detect probl em detect and record probl em	n1 p2	n1 p2	m1		z		z		z	
di srupti on I di srupti on II di srupti on III	e1	e2	h1		j1		j1 k1 j1		h1 l1 h1	

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

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

**Table 13: Inferior state table – request confirm states**

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

2985

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

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

2986

2987

2988

2988  
2989  
2990  
2991  
2992  
2993  
2994  
2995  
2996  
2997  
2998  
2999  
3000  
3001  
3002  
3003  
3004  
3005  
3006  
3007  
3008  
3009  
3010  
3011  
3012  
3013  
3014  
3015  
3016  
3017  
3018  
3019  
3020  
3021  
3022  
3023  
3024  
3025  
3026  
3027  
3028  
3029  
3030  
3031  
3032  
3033

## Failure Recovery

### Types of failure

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

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

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

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

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

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

3034 After recovery from node failure, the implementation behaves much as if a communication  
3035 failure had occurred.

3036

### 3037 Persistent information

3038

3039 BTP requires that some decision events are persisted – that information recording an  
3040 Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s  
3041 autonomous decision survive failure. Making the first two decisions persistent ensures that a  
3042 consistent decision can be reached for the business transaction and that it is delivered to all  
3043 involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to  
3044 ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the  
3045 contradiction will be reported to the Superior, despite failures.

3046

3047 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the  
3048 active state (unlike many transaction protocols, where a communication or endpoint failure in  
3049 active state would invariably cause rollback of the transaction). Recovery in the active state  
3050 may require that the application exchange is resynchronised as well – BTP does not directly  
3051 support this, but does allow continuation of the business transaction as such. In the state  
3052 tables, from some states, there are several levels of disruption, distinguished by which state  
3053 the implementation transits to – this represents the survival of different extents of state  
3054 information over failure and recovery. The different levels of disruption describe legitimate  
3055 states for the endpoint to be in after it has recovered – **they do not require that all  
3056 implementations are able to exhibit the appropriate partial loss of state information.**

3057

3058 The absence of a destination state for the disruption events means that such a transition is not  
3059 legitimate – thus, for example, an Inferior that has decided to be prepared will always recover  
3060 to the same state, by virtue of the information persisted in the “decide to be prepared” event.

3060

3061 Apart from the (optional) recovery in active state, BTP follows the well-known presume-  
3062 abort model – it is only required that information be persisted when decisions are made (and  
3063 not, e.g. on enrolment). This means that on recovery, one side may have persistent  
3064 information but the other does not. This occurs when an Inferior has decided to be prepared  
3065 but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the  
3066 Superior did confirm, and the Inferior applied the confirm, removed its persistent information  
3067 but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it  
3068 still had the persistent information when the failure occurred).

3069

3070 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to  
3071 re-establish communication with the Superior, to apply a confirm decision and to apply a  
3072 cancel decision. It will thus need to include

3073

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

3074

Superior address (as on CONTEXT)

3075

Superior identifier (as on CONTEXT)

3076

default-is-cancel value (as on PREPARED)

3077

3078 The information needed to apply confirm/cancel decisions will depend on the application and  
3079 the associated operations. It may also normally be necessary to persist any qualifiers that

3080 were sent with the PREPARED message or application messages sent with the PREPARED,  
3081 since the PREPARED message will be repeated if a failure occurs.

3082  
3083 A Superior must record corresponding information to allow it to re-establish communication  
3084 with the Inferior:

3085 Inferior address (as on ENROL)

3086 Inferior identifier (as on ENROL)

3087

3088 A Superior that is the Decider for the business transaction need only persist this information  
3089 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A  
3090 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as  
3091 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as  
3092 Superior (to this Inferior) as part of the persistent information of its decision to be prepared  
3093 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and  
3094 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If  
3095 CONFIRM is received, the persistent information may be changed to show the confirm  
3096 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.  
3097 If the persistent information is left unchanged and there is a node failure, on recovery the  
3098 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision  
3099 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

3100

3101 After failure, an implementation may not be able to restore an endpoint to the appropriate  
3102 state immediately – in particular, the necessary persistent information may be inaccessible,  
3103 although the implementation can respond to received BTP messages. In such a case, a  
3104 Superior may reply to any BTP message except INFERIOR\_STATE/\* (i.e. with a “reply-  
3105 requested” value “false”) with SUPERIOR\_STATE/inaccessible and an Inferior to any BTP  
3106 message except SUPERIOR\_STATE/\* with “INFERIOR\_STATE/inaccessible. Receipt of  
3107 the \*\_STATE/inaccessible messages has no effect on the endpoint state.

3108

## 3109 Redirection

3110

3111 As described above, BTP uses the presume-abort model for recovery. A corollary of this is  
3112 that there are cases where one side will attempt to re-establish communication when there is  
3113 no persistent information for the relationship at the far-end. In such cases, it is important the  
3114 side that is attempting recovery can distinguish between unsuccessful attempts to connect to  
3115 the holder of the persistent information and when the information no longer exists. If the peer  
3116 information does not exist, this side can draw conclusions and complete appropriately; if they  
3117 merely fail to get through they are stuck in attempting recovery.

3118

3119 Two mechanisms are provided to make it possible that even when one side of a  
3120 Superior:Inferior relationship has completed, that a message can eventually get through to  
3121 something that can definitively report the status, distinguishing this case from a temporary  
3122 inability to access the state of a continuing transaction element. The mechanisms are:

- 3123 o Address fields which provide a “callback address” can be a set of addresses,  
3124 which are alternatives one of which is chosen as the target address for the  
3125 future message. If the sender of that message finds the address does not work,  
3126 it can try a different alternative.

- 3127                   o    The REDIRECT message can be used to inform the peer that an address  
3128                            previously given is no longer valid and to supply a replacement address (or  
3129                            set of addresses). REDIRECT can be issued either as a response to receipt of  
3130                            a message or spontaneously.

3131  
3132                   The two mechanisms can be used in combination, with one or more of the original set of  
3133                   addresses just being a redirector, which does not itself ever have direct access to the state  
3134                   information for the transaction, but will respond to any message with an appropriate  
3135                   REDIRECT.

3136  
3137                   An alternative implementation approach is to have a single addressable entity that uses the  
3138                   same address for all transactions, distinguishing them by identifier, and which always  
3139                   recovers to use the same address. Such an implementation would not need to supply  
3140                   “backup” addresses (and would only use REDIRECT if it was being permanently migrated).

### 3141 3142                   **Terminator:Decider failures**

3143  
3144                   BTP does not provide facilities or impose requirements on the recovery of  
3145                   Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator  
3146                   may survive failures (by retaining knowledge of the Decider’s address and identifier), but this  
3147                   is an implementation option. Although a Decider (if it decides to confirm) will persist  
3148                   information about the confirm decision, it is not required, after failure, to remain accessible  
3149                   using the inferior address it offered to the Terminator. Any such recovery is an  
3150                   implementation option.

3151  
3152                   A Decider’s address (as returned on BEGUN) may be a set of addresses, allowing a failed  
3153                   Decider to be recovered at a different address.

3154  
3155                   A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and  
3156                   thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for  
3157                   a CONFIRM\_TRANSACTION that will never arrive, the standard qualifier “Transaction  
3158                   timelimit” can be used (by the Initiator) to inform the Decider when it can assume the  
3159                   Terminator will not issue CONFIRM\_TRANSACTION and so it (the Decider) should initiate  
3160                   cancellation.

### 3161 3162                   **XML representation of Message Set**

3163  
3164                   This section describes the syntax for BTP messages in XML. These XML messages represent  
3165                   a midpoint between the abstract messages and what actually gets sent on the wire.

3166  
3167                   All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)  
3168                   [3121](#)

3169  
3170                   The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:xml

3171  
3172                   In addition to an XML schema, this specification uses an informal syntax to describe the  
3173                   structure of the BTP messages. The syntax appears as an XML instance, but the values

3174 contain data types instead of values. The following symbols are appended to some of the  
3175 XML constructs: ? (zero or one), \* (zero or more), + (one or more.) The absence of one of  
3176 these symbols corresponds to "one and only one."

3177

## 3178 **Addresses**

3179

3180 As described in the "Abstract Message and Associated Contracts – Addresses" section, a BTP  
3181 address comprises three parts, and for a target address only the "additional information" field  
3182 is inside the BTP messages. For all BTP messages whose abstract form includes a target  
3183 address parameter, the corresponding XML representation includes a "target-additional-  
3184 information" element. This element may be omitted if it would be empty.

3185

3186 For other addresses, all three fields are represent, as in:

3187

```
3188 <ctp:some-address>  
3189   <ctp:binding-name>...carrier binding URI...</ctp:binding-name>  
3190   <ctp:binding-address>...carrier specific  
3191   address...</ctp:binding-address>  
3192   <ctp:additional-information>...optional additional addressing  
3193   information...</ctp:additional-information> ?  
3194 </ctp:some-address>
```

3195

3196

3197 A "published" address can be a set of <some-address>, which are alternatives which can be  
3198 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to  
3199 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice  
3200 of which address to use (depending on which binding is preferable.) In the case where  
3201 multiple addresses are used for redundancy, a priority attribute can be specified to help the  
3202 receiver choose among the addresses- the address with the highest priority should be used,  
3203 other things being equal. The priority is used as a hint and does not enforce any behaviour in  
3204 the receiver of the message. Default priority is a value of 1.

3205

## 3206 **Qualifiers**

3207 The "Qualifier name" is used as the element name, within the namespace of the "Qualifier  
3208 group".

3209

3210 **Examples:**

3211

```
3212 <ctpq:inferior-timeout  
3213   xmlns:ctpq="urn:oasis:names:tc:BTP:qualifiers"  
3214   xmlns:ctp="urn:oasis:names:tc:BTP:xml "  
3215   ctp:must-be-understood="false"  
3216   ctp:to-be-propagated="false">1800</ctpq:inferior-timeout>
```

3217

```
3218 <auth:username  
3219   xmlns:auth="http://www.example.com/ns/auth"  
3220   xmlns:ctp="urn:oasis:names:tc:BTP:xml "  
3221   ctp:must-be-understood="true"  
3222   ctp:to-be-propagated="true">jtauber</auth:username>
```

3222

3223 Attributes must-be-understood **has default value “true”** and to-be-propagated has default  
3224 value “false”.

3225

## 3226 Identifiers

3227

3228 ~~Identifiers shall be URIs. Unspecified length strings made of up hexadecimal digits (0-9, A-~~  
3229 ~~->F). Note: lower case a->f are not valid.~~

3230

3231 ~~Examples: "01", "FAB224234CCCC2"~~

3232

3233 Note — ~~Identifiers need to be unambiguous over all the systems that might be involved in a~~  
3234 ~~business transaction and over indefinite periods of time. Apart from their generation, Use of~~  
3235 ~~hexadecimal digits avoids problems with character code representations.~~ The only operation  
3236 the BTP implementations have to perform on identifiers is to match them.

3237

## 3238 Message References

3239 Each BTP message has an optional id attribute to give it a unique identifier. An application  
3240 can make use of those identifiers, but no processing is enforced.

3241

## 3242 Messages

3243

### 3244 CONTEXT

3245

```
3246 <btp:context id? superior-type="cohesion|atom">  
3247   <btp:superior-address> +  
3248     ...address...  
3249   </btp:superior-address>  
3250   <btp:superior-identifier>...hexstring...</btp:superior-  
3251 identifier>  
3252   <btp:reply-address> ?  
3253     ...address...  
3254   </btp:reply-address>  
3255   <btp:qualifiers> ?  
3256     ...qualifiers...  
3257   </btp:qualifiers>  
3258 </btp:context>
```

3259

3260

### 3261 CONTEXT\_REPLY

3262

```
3263 <btp:context-reply id? superior-type="cohesion|atom">  
3264   <btp:target-additional-information> ?  
3265     ...additional address information...  
3266   </btp:target-additional-information>  
3267   <btp:superior-address> +  
3268     ...address...  
3269   </btp:superior-address>  
3270   <btp:superior-identifier>...hexstring...</btp:superior-  
3271 identifier>
```

```
3272     <completion-status>completed|related|repudiated</completion-  
3273 status>  
3274     <btp:qualifiers> ?  
3275     ...qualifiers...  
3276     </btp:qualifiers>  
3277 </btp:context>
```

## BEGIN

```
3281  
3282 <btp:begin id? transaction-type="cohesion|atom">  
3283   <btp:target-additional-information>  
3284     ...additional address information...  
3285   </btp:target-additional-information>  
3286   <btp:reply-address>  
3287     ...address...  
3288   </btp:reply-address>  
3289   <btp:qualifiers> ?  
3290   ...qualifiers...  
3291   </btp:qualifiers>  
3292 </btp:begin>
```

## BEGUN

```
3296  
3297 <btp:begun id? transaction-type="cohesion|atom">  
3298   <btp:target-additional-information>  
3299     ...additional address information...  
3300   </btp:target-additional-information>  
3301   <btp:decider-address> ?  
3302     ...address...  
3303   </btp:decider-address>  
3304   <btp:transaction-identifier>...hexstring...</btp:transaction-  
3305 identifier> ?  
3306   <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?  
3307   <btp:inferior-address> ?  
3308     ...address...  
3309   </btp:inferior-address>  
3310   <btp:qualifiers> ?  
3311   ...qualifiers...  
3312   </btp:qualifiers>  
3313 </btp:begun>
```

## ENROL

```
3317  
3318 <btp:enrol reply-requested="true|false" id?>  
3319   <btp:target-additional-information>  
3320     ...additional address information...  
3321   </btp:target-additional-information>
```

```
3322     <btpr:superior-identifier>...hexstring...</btpr:superior-
3323 identifier>
3324     <btpr:reply-address> ?
3325     ...address...
3326     </btpr:reply-address>
3327     <btpr:inferior-address> +
3328     ...address...
3329     </btpr:inferior-address>
3330     <btpr:inferior-identifier>...hexstring...</btpr:inferior-
3331 identifier>
3332     <btpr:qualifiers> ?
3333     ...qualifiers...
3334     </btpr:qualifiers>
3335 </btpr:enrol>
```

3336

3337

## ENROLLED

3339

3340

```
<btpr:enrolled id?>
<btpr:target-additional-information>
...additional address information...
</btpr:target-additional-information>
<btpr:inferior-identifier>...hexstring...</btpr:inferior-
identifier>
<btpr:inferior-handle>...hexstring...</btpr:inferior:handle> ?
<btpr:qualifiers> ?
...qualifiers...
</btpr:qualifiers>
</btpr:enrolled>
```

3351

3352

## RESIGN

3353

3354

```
<btpr:resign response-requested="true|false" id?>
<btpr:target-additional-information>
...additional address information...
</btpr:target-additional-information>
<btpr:superior-identifier>...hexstring...</btpr:superior-
identifier>
<btpr:inferior-address> +
...address...
</btpr:inferior-address>
<btpr:inferior-identifier>...hexstring...</btpr:inferior-
identifier>
<btpr:qualifiers> ?
...qualifiers...
</btpr:qualifiers>
</btpr:resign>
```

3370

3371

3372  
3373  
3374  
3375  
3376  
3377  
3378  
3379  
3380  
3381  
3382  
3383  
3384  
3385  
3386  
3387  
3388  
3389  
3390  
3391  
3392  
3393  
3394  
3395  
3396  
3397  
3398  
3399  
3400  
3401  
3402  
3403  
3404  
3405  
3406  
3407  
3408  
3409  
3410  
3411  
3412  
3413  
3414  
3415  
3416  
3417  
3418  
3419  
3420  
3421  
3422

## RESIGNED

```
<btp:resigned id?>  
  <btp:target-additional-information>  
    ...additional address information...  
  </btp:target-additional-information>  
  <btp:inferior-identifier>...hexstring...</btp:inferior-  
inferior-  
identifier>  
  <btp:qualifiers> ?  
    ...qualifiers...  
  </btp:qualifiers>  
</btp:resigned>
```

## PREPARE

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

## PREPARED

```
<btp:prepared default-is-cancel="false|true" id?>  
  <btp:target-additional-information>  
    ...additional address information...  
  </btp:target-additional-information>  
  <btp:superior-identifier>...hexstring...</btp:superior-  
superior-  
identifier>  
  <btp:inferior-address> +  
    ...address...  
  </btp:inferior-address>  
  <btp:inferior-identifier>...hexstring...</btp:inferior-  
inferior-  
identifier>  
  <btp:qualifiers> ?  
    ...qualifiers...  
  </btp:qualifiers>  
</btp:prepared>
```

## CONFIRM

```
<btp:confirm id?>  
  <btp:target-additional-information>
```

```
3423     ...additional address information...
3424     </btp:target-additional-information>
3425     <btp:inferior-identifier>...hexstring...</btp:inferior-
3426     identifier>
3427     <btp:qualifiers> ?
3428     ...qualifiers...
3429     </btp:qualifiers>
3430 </btp:confirm>
```

3431  
3432

### CONFIRMED

```
3433
3434
3435     <btp:confirmed confirmed-received="true|false" id?>
3436     <btp:target-additional-information>
3437     ...additional address information...
3438     </btp:target-additional-information>
3439     <btp:superior-identifier>...hexstring...</btp:superior-
3440     identifier>
3441     <btp:inferior-address> ?
3442     ...address...
3443     </btp:inferior-address>
3444     <btp:inferior-identifier>...hexstring...</btp:inferior-
3445     identifier> ?
3446     <btp:qualifiers> ?
3447     ...qualifiers...
3448     </btp:qualifiers>
3449 </btp:confirmed>
```

3450  
3451

### CANCEL

```
3452
3453
3454     <btp:cancel id?>
3455     <btp:target-additional-information>
3456     ...additional address information...
3457     </btp:target-additional-information>
3458     <btp:inferior-identifier>...hexstring...</btp:inferior-
3459     identifier> ?
3460     <btp:reply-address> ?
3461     ...address...
3462     </btp:reply-address>
3463     <btp:qualifiers> ?
3464     ...qualifiers...
3465     </btp:qualifiers>
3466 </btp:cancel>
```

3467  
3468

### CANCELLED

```
3469
3470
3471     <btp:cancelled id?>
3472     <btp:target-additional-information>
3473     ...additional address information...
```

```
3474     </btp:target-additional-information>
3475     <btp:superior-identifier>...hexstring...</btp:superior-
3476 identifier>
3477     <btp:inferior-address> +
3478     ...address...
3479     </btp:inferior-address> ?
3480     <btp:inferior-identifier>...hexstring...</btp:inferior-
3481 identifier> ?
3482     <btp:qualifiers> ?
3483     ...qualifiers...
3484     </btp:qualifiers>
3485 </btp:cancelled>
```

3486

3487

## CONFIRM\_ONE\_PHASE

3489

```
3490 <btp:confirm-one-phase report-hazard="true|false" id?>
3491   <btp:target-additional-information>
3492     ...additional address information...
3493   </btp:target-additional-information>
3494   <btp:inferior-identifier>...hexstring...</btp:inferior-
3495 identifier>
3496   <btp:qualifiers> ?
3497     ...qualifiers...
3498   </btp:qualifiers>
3499 </btp:confirm-one-phase>
```

3500

3501

## HAZARD

3502

```
3503 <btp:hazard level="mixed|possible" id?>
3504   <btp:target-additional-information>
3505     ...additional address information...
3506   </btp:target-additional-information>
3507   <btp:superior-identifier>...hexstring...</btp:superior-
3508 identifier>
3509   <btp:inferior-address> +
3510     ...address...
3511   </btp:inferior-address>
3512   <btp:inferior-identifier>...hexstring...</btp:inferior-
3513 identifier>
3514   <btp:qualifiers> ?
3515     ...qualifiers...
3516   </btp:qualifiers>
3517 </btp:hazard>
```

3518

3519

3520

## CONTRADICTION

3521

```
3522 <btp:contradiction id?>
3523   <btp:target-additional-information>
3524     ...additional address information...
```

```
3525     </btp:target-additional-information>
3526     <btp:inferior-identifier>...hexstring...</btp:inferior-
3527 identifier>
3528     <btp:qualifiers> ?
3529     ...qualifiers...
3530     </btp:qualifiers>
3531 </btp:contradiction>
```

3532

3533

3534

## SUPERIOR\_STATE

3535

3536

```
<btp:superior-state reply-requested="true|false" id?>
<btp:target-additional-information>
...additional address information...
</btp:target-additional-information>
<btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
<btp:status>active|prepared-
received|inaccessible|unknown</btp:status>
<btp:qualifiers> ?
...qualifiers...
</btp:qualifiers>
</btp:superior-state>
```

3548

3549

3550

## INFERIOR\_STATE

3551

3552

```
<btp:inferior-state reply-requested="true|false" id?>
<btp:target-additional-information>
...additional address information...
</btp:target-additional-information>
<btp:superior-identifier>...hexstring...</btp:superior-
identifier>
<btp:inferior-address> +
...address...
</btp:inferior-address>
<btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
<btp:status> active| inaccessible|unknown</btp:status>
<btp:qualifiers> ?
...qualifiers...
</btp:qualifiers>
</btp:inferior-state>
```

3568

3569

3570

3571

3572

## REDIRECT

3573

3574

```
<btp:redirect id?>
<btp:target-additional-information>
```

3575

```

3576     ...additional address information...
3577     </btp:target-additional-information>
3578     <btp:superior-identifier>...hexstring...</btp:superior-
3579     identifier> ?
3580     <btp:inferior-identifier>...hexstring...</btp:inferior-
3581     identifier>
3582     <btp:old-address> +
3583     ...address...
3584     </btp:old-address>
3585     <btp:new-address> +
3586     ...address...
3587     </btp:new-address>
3588     <btp:qualifiers> ?
3589     ...qualifiers...
3590     </btp:qualifiers>
3591 </btp:redirect>
3592

```

### PREPARE\_INFERIORS

```

3593
3594
3595     <btp:prepare-inferiors id?>
3596     <btp:target-additional-information>
3597     ...additional address information...
3598     </btp:target-additional-information>
3599     <btp:reply-address> ?
3600     ...address...
3601     </btp:reply-address>
3602     <btp:transaction-identifier>...hexstring...</btp:transaction-
3603     identifier> ?
3604     <btp:inferiors-list> ?
3605     <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3606     +
3607     </btp:inferiors-list>
3608     <btp:qualifiers> ?
3609     ...qualifiers...
3610     </btp:qualifiers>
3611 </btp:prepare-inferiors>
3612
3613

```

### CONFIRM\_TRANSACTION

```

3614
3615
3616     <btp:confirm-transaction report-hazard="true|false" id?>
3617     <btp:target-additional-information>
3618     ...additional address information...
3619     </btp:target-additional-information>
3620     <btp:reply-address>
3621     ...address...
3622     </btp:reply-address>
3623     <btp:transaction-identifier>...hexstring...</btp:transaction-
3624     identifier>
3625     <btp:inferiors-list> ?
3626     <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3627     +

```

```
3628     </btp:inferiors-list>
3629     <btp:qualifiers> ?
3630     ...qualifiers...
3631     </btp:qualifiers>
3632 </btp: confirm_transaction>
```

3633  
3634

## TRANSACTION\_CONFIRMED

```
3636
3637     <btp:transaction-confirmed id?>
3638     <btp:target-additional-information>
3639     ...additional address information...
3640     </btp:target-additional-information>
3641     <btp:decider-address> ?
3642     ...address...
3643     </btp:decider-address>
3644     <btp:transaction-identifier>...hexstring...</btp:transaction-
3645     identifier> ?
3646     <btp:qualifiers> ?
3647     ...qualifiers...
3648     </btp:qualifiers>
3649 </btp:transaction-confirmed>
```

3650  
3651

## CANCEL\_TRANSACTION

```
3653
3654     <btp:cancel_transaction id?>
3655     <btp:target-additional-information>
3656     ...additional address information...
3657     </btp:target-additional-information>
3658     <btp:reply-address> ?
3659     ...address...
3660     </btp:reply-address>
3661     <btp:transaction-identifier>...hexstring...</btp:transaction-
3662     identifier> ?
3663     <btp:qualifiers> ?
3664     ...qualifiers...
3665     </btp:qualifiers>
3666 </btp:cancel_transaction>
```

3667  
3668

## CANCEL\_INFERIORS

```
3669
3670     <btp: -cancel-inferiors id?>
3671     <btp:target-additional-information>
3672     ...additional address information...
3673     </btp:target-additional-information>
3674     <btp:reply-address> ?
3675     ...address...
3676     </btp:reply-address>
3677     <btp:transaction-identifier>...hexstring...</btp:transaction-
3678     identifier> ?
```

```
3679     <btp:inferiors-list><btp:inferior-
3680 handle>...hexstring...</btp:inferior-handle>
3681 </btp:inferiors-list>
3682 <btp:qualifiers> ?
3683     ...qualifiers...
3684 </btp:qualifiers>
3685 </btp:cancel-inferiors>
```

3686

3687

3688

## TRANSACTION\_CANCELLED

3689

3690

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

3691

3692

3693

3694

3695

3696

3697

3698

3699

3700

3701

3702

3703

3704

3705

## REQUEST\_INFERIOR\_STATUSES

3706

3707

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

3708

3709

3710

3711

3712

3713

3714

3715

3716

3717

3718

3719

3720

3721

3722

3723

3724

3725

## INFERIOR\_STATUSES

3726

3727

3728

3729

```
<btp:inferior_statuses id?>
<btp:target-additional-information>
    ...additional address information...
```

```

3730     </btp:target-additional-information>
3731     <btp:responders-address>
3732         ...address...
3733     </btp:responders-address>
3734     <btp:responders-identifier>...hexstring...</btp:responders-
3735 identifier>
3736     <btp:status-list>
3737         <btp:status-item> +
3738             <btp:inferior-handle>...hexstring...</btp:inferior-
3739 handle>
3740             <btp:status>active|resigned|preparing|prepared|
3741                 autonomously-confirmed|autonomously-cancelled|
3742                 confirming|confirmed|cancelling|cancelled|
3743                 cancel-contradiction|confirm-contradiction|
3744                 hazard</btp:status>
3745             <btp:qualifiers> ?
3746                 ...qualifiers...
3747             </btp:qualifiers>
3748         </btp:status-item>
3749     </btp:status-list>
3750     <btp:qualifiers> ?
3751         ...qualifiers...
3752     </btp:qualifiers>
3753 </btp:inferior_statuses>

```

## REQUEST\_STATUS

```

3754
3755
3756
3757
3758     <btp:request_status id?>
3759     <btp:target-additional-information>
3760         ...additional address information...
3761     </btp:target-additional-information>
3762     <btp:reply-address>
3763         ...address...
3764     </btp:reply-address>
3765     <btp:target-identifier>...hexstring...</btp:target-identifier>
3766     <btp:qualifiers> ?
3767         ...qualifiers...
3768     </btp:qualifiers>
3769 </btp:request_status>

```

## STATUS

```

3770
3771
3772
3773     <btp:status id?>
3774     <btp:target-additional-information>
3775         ...additional address information...
3776     </btp:target-additional-information>
3777     <btp:responder-address>
3778         ...address...
3779     </btp:responder-address>
3780     <btp:responder-identifier>...hexstring...</btp:responder-
3781 identifier>

```

```

3782
3783     <btp:status-value> created|enrolling|active|resigning|
3784         resigned|preparing|prepared|
3785         confirming|confirmed|cancelling|cancelled|
3786         cancel-contradiction|confirm-contradiction|
3787         hazard|contradicted|unknown|inaccessible</btp:status-
3788 value>
3789     <btp:qualifiers> ?
3790         ...qualifiers...
3791     </btp:qualifiers>
3792 </btp:status>

```

## FAULT

```

3796 <btp:fault id?>
3797     <btp:target-additional-information>
3798         ...additional address information...
3799     </btp:target-additional-information>
3800     <btp:superior-identifier>...hexstring...</btp:superior-
3801 identifier> ?
3802     <btp:inferior-identifier>...hexstring...</btp:inferior-
3803 identifier> ?
3804     <btp:fault-type>...fault type name...</btp:fault-type>
3805     <btp:fault-data>...fault data...</btp:fault-data> ?
3806     <btp:qualifiers> ?
3807         ...qualifiers...
3808     </btp:qualifiers>
3809 </btp:fault>

```

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

- 3815           o   general
- 3816           o   unknown-parameter
- 3817           o   wrong-state
- 3818           o   communication-failure
- 3819           o   invalid-superior
- 3820           o   duplicate-inferior
- 3821           o   unknown-inferior

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

Fault data can take on various forms:

Free text:

```

3830     <btp:fault-data>...string data...</btp:fault-data>
3831

```

3832  
3833  
3834  
3835  
3836  
3837  
3838  
3839  
3840  
3841  
3842  
3843  
3844  
3845  
3846  
3847  
3848  
3849  
3850  
3851  
3852  
3853  
3854  
3855  
3856  
3857  
3858  
3859  
3860  
3861  
3862  
3863  
3864  
3865  
3866  
3867  
3868  
3869  
3870  
3871  
3872  
3873  
3874  
3875  
3876  
3877  
3878  
3879  
3880

Identifier:

```
<btqp:fault-data>...hexstring...</btqp:fault-data>
```

Inferior Identity:

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

### Standard qualifiers

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

### Transaction timelimit

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

### Inferior timeout

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

### Minimum inferior timeout

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

### Compounding of Messages

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

3881  
3882  
3883  
3884  
3885  
3886  
3887  
3888  
3889  
3890  
3891  
3892  
3893  
3894  
3895  
3896  
3897  
3898  
3899  
3900  
3901  
3902  
3903  
3904  
3905

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

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

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

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

3905

## 3906 **Carrier Protocol Bindings**

3907

3908 The notion of bindings is introduced to act as the glue between the BTP messages and an  
3909 underlying transport. A binding specification must define various particulars of how the BTP  
3910 messages are carried and some aspects of how the related application messages are carried.

3911 This document specifies two bindings: a SOAP binding and a SOAP + Attachments binding.

3912 However, other bindings could be specified by the Oasis BTP technical committee or by a  
3913 third party. For example, in the future a binding might exist to put a BTP message directly on  
3914 top of HTTP without the use of SOAP, or a closed community could define their own

3915 binding. To ensure that such specifications are complete, the Binding Proforma defines the  
3916 information that must be included in a binding specification.

3917

### 3918 **Carrier Protocol Binding Proforma**

3919

3920 A BTP carrier binding specification should provide the following information:

3921

3922 **Binding name:** A name for the binding, as used in the “binding name” field of BTP  
3923 addresses (and available for declaring the capabilities of an implementation). Binding  
3924 specified in this document, and future revisions of this document have binding names that are  
3925 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).  
3926 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in  
3927 this document use numbers to identify the version of the binding, not the version(s) of the  
3928 carrier protocol.

3929

3930 **Binding address format:** This section states the format of the “binding address” field of a  
3931 BTP address for this binding. For many bindings, this will be a URL of some kind; for other  
3932 bindings it may be some other form

3933

3934 **BTP message representation:** This section will define how BTP messages are represented.  
3935 For many bindings, the BTP message syntax will be as specified in the XML schema defined  
3936 in this document, and the normal string encoding of that XML will be used.

3937

3938 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that  
3939 are not related to application messages are sent in either direction between Superior and  
3940 Inferior. (i.e. those messages sent directly between BTP actors). This mapping need not be  
3941 symmetric (i.e. Superior to Inferior may differ to some degree to Inferior to Superior). The  
3942 mapping may define particular rules for particular BTP messages, or messages with particular  
3943 parameter values (e.g. the FAULT message with “fault-type” “CommunicationFailure” will  
3944 typically not be sent as a BTP message). The mapping states any constraints or requirements  
3945 on which BTP may or must be bundled together by compounding.

3946

3947 **Mapping for BTP messages related to application messages:** This section will define how  
3948 BTP messages that are related to application messages are sent. A binding specification may  
3949 defer details of this to a particular application (e.g. a mapping specification could just say  
3950 “the CONTEXT may be carried as a parameter of an application invocation”). Alternatively,

3951 the binding may specify a general method that represents the relationship between application  
3952 and BTP messages.

3953

3954 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly  
3955 but are treated as implicit in application messages or other BTP messages. This may depend  
3956 on particular parameter values of the BTP messages or the application messages.

3957

3958 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier  
3959 protocol and of BTP shall be defined. This may include definition of which carrier protocol  
3960 exceptions are equivalent to a FAULT/communication-failure message.

3961

3962 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.  
3963 If BTP addresses with different bindings are be considered to match (for purposes of  
3964 identifying the peer Superior/Inferior and redirection), this should be specified here.

3965

3966 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are  
3967 imposed by use of this binding should be listed. This would include limitations on which  
3968 messages can be sent, which event sequences are supported and restrictions on parameter  
3969 values. Such limitations may reduce the usefulness of an implementation, but may be  
3970 appropriate in certain environments.

3971

3972 **Other:** Other features of the binding, especially any that will potentially affect interoperability  
3973 should be specified here. This may include restrictions or requirements on the use or support  
3974 of optional carrier parameters or mechanisms.

3975

### 3976 **Bindings for request/response carrier protocols**

3977

3978 BTP does not generally follow request/response pattern. In particular, on the outcome  
3979 relationship either side may initiate a message – this is an essential part of the presume-abort  
3980 recovery paradigm although it is not limited to recovery cases. However, there are some BTP  
3981 messages, especially in the control relationship, that do have a request/response pattern.  
3982 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The  
3983 specification of a binding specification to a request/response carrier protocol needs to state  
3984 what rules apply – which messages can be carried by requests, which by responses. The  
3985 simplest rule is to send all BTP messages on requests, and let the carrier responses travel back  
3986 empty. This would be inefficient in use of network resources, and possibly inconvenient  
3987 when used for the BTP request/response pairs.

3988

3989 This section defines a set of rules that allow more efficient use of the carrier, while allowing  
3990 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the  
3991 carrier response. These rules are specified in this section to enable binding specifications to  
3992 reference them, without requiring each binding specification to repeat similar information.

3993

3994 A binding to a request/response carrier is not required to use these rules. It may define other  
3995 rules.

3996

3997  
3998  
3999  
4000  
4001  
4002  
4003  
4004  
4005  
4006  
4007  
4008  
4009  
4010  
4011  
4012  
4013  
4014  
4015  
4016  
4017  
4018  
4019  
4020  
4021  
4022  
4023  
4024  
4025  
4026  
4027  
4028  
4029  
4030  
4031  
4032  
4033  
4034  
4035  
4036  
4037  
4038  
4039  
4040  
4041  
4042

## Request/response exploitation rules

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

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

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

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

The rules:

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

- 4043 e) A BTP actor that receives a carrier protocol request carrying BTP messages that  
4044 do have a reply address, or which initiate processing that produces BTP messages  
4045 whose target binding address matches the origin of the request, **may** freely  
4046 choose whether to use the carrier protocol response for the replies, or to send  
4047 back an “empty carrier protocol response”, and send the BTP replies in a  
4048 separately initiated carrier protocol request. The characteristics of an “empty  
4049 carrier protocol response” shall be stated in the particular binding specification.  
4050
- 4051 f) A BTP actor that sends BTP messages on a carrier protocol request **must** be able  
4052 to accept returning BTP messages on the corresponding carrier protocol response  
4053 and, if the actor has offered an address on which it will receive carrier requests,  
4054 must be able to accept “replying” BTP messages on a separate carrier protocol  
4055 request.  
4056

## 4057 SOAP Binding

4058 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)  
4059 specification, using the SOAP literal messaging style conventions. If no application message  
4060 is sent at the same time, the BTP messages are contained within the SOAP Body element. If  
4061 application messages are sent, the BTP messages are contained in the SOAP Header element.  
4062

4063 **Binding name:** soap-http-1  
4064

4065 **Binding address format:** shall be a URL, of type HTTP.  
4066

4067 **BTP message representation:** The string representation of the XML, as specified in the  
4068 XML schema defined in this document shall be used. The BTP XML messages are embedded  
4069 in the SOAP message without the use of any specific encoding rules (literal style SOAP  
4070 message); hence the encodingStyle attribute need not be set or can be set to an empty string.  
4071

4072 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be  
4073 used.  
4074

4075 BTP messages sent on an HTTP request or HTTP response which is not carrying an  
4076 application message, the messages are contained in a single btp:messages element which is  
4077 the immediate child element of the SOAP Body element.  
4078

4079 An “empty carrier protocol response” sent after receiving an HTTP request containing a  
4080 btp:messages element in the SOAP Body and the implementation BTP actor chooses just to  
4081 reply at the lower level (and when the request/response exploitation rules allow an empty  
4082 carrier protocol response), shall be any of:  
4083

- 4084 a) an empty HTTP response
- 4085 b) an HTTP response containing an empty SOAP Envelope
- 4086 c) an HTTP response containing a SOAP Envelope containing a single, empty  
4087 btp:messages element.  
4088

4089 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they  
4090 have no effect on the BTP sequence (other than indicating that the earlier sending did not  
4091 cause a communication failure.)  
4092

4093

4094

4095

4096 If an application message is being sent at the same time, the mapping for related messages  
4097 shall be used, as if the BTP messages were related to the application message. (There is no  
4098 ambiguity in whether the BTP messages are related, because only CONTEXT and ENROL  
4099 can be related to an application message.)

4100

4101 **Mapping for BTP messages related to application messages:** All BTP messages sent with  
4102 an application message, whether related to the application message or not, shall be sent in a  
4103 single btp:messages element in the SOAP Header. There shall be precisely one btp:messages  
4104 element in the SOAP Header.

4105

4106 The “request/response exploitation” rules shall apply to the BTP messages carried in the  
4107 SOAP Header, as if they had been carried in a SOAP Body, unrelated to an application  
4108 message, sent to the same binding address.

4108

---

Note – The application protocol itself (which is using the SOAP Body) may  
4109 use the SOAP RPC or document approach – this is determined by the  
4110 application.

---

4111

4112

4113

4114 Only CONTEXT and ENROL messages are related (&) to application messages. If there is  
4115 only one CONTEXT or one ENROL message present in the SOAP Header, it is assumed to  
be related to the whole of the application message in the SOAP Body. If there are multiple  
CONTEXT or ENROL messages, any relation of these BTP messages shall be indicated by  
application specific means.

4116

---

Note 1 – An application protocol could use references to the ID values of the  
4117 BTP messages to indicate relation between BTP CONTEXT or ENROL  
4118 messages and the application message.

4119

4120

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

---

4121

4122

4123

4124

4125

4126

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

4127

4128

4129

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

4130 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding  
4131 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-  
4132 http-1” if the binding address and additional information fields match.

4133  
4134 **Limitations on BTP use:** None

4135  
4136 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor  
4137 attribute. The SOAPAction HTTP header is left to be application specific when there are  
4138 application messages in the SOAP Body, as an already existing web service that is being  
4139 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP  
4140 header shall be omitted when the SOAP message carries only BTP messages in the SOAP  
4141 Body.

4142  
4143 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP  
4144 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to  
4145 determine whether any enrolments are necessary and replies with CONTEXT\_REPLY as  
4146 appropriate. The sender of the CONTEXT (and related application message) can use this to  
4147 ensure that the application work is performed as part of the business transaction, assuming the  
4148 receiver’s SOAP implementation supports the mustUnderstand attribute. If mustUnderstand if  
4149 false, a receiver can ignore the CONTEXT (if BTP is not supported there), and no  
4150 CONTEXT\_REPLY will be returned. It is a local option on the sender (client) side whether  
4151 the absence of a CONTEXT\_REPLY is assumed to be equivalent to aCONTEXT\_REPLY/ok  
4152 (and the business transaction allowed to proceed to confirmation).

4153  
4154 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to  
4155 enforce these requirements.

#### 4156 **Example scenario using SOAP binding**

4157  
4158 The example below shows an application request with CONTEXT message sent from  
4159 client.example.com (which includes the Superior) to services.example.com (Service).

```
4160  
4161  
4162 <soap:Envelope  
4163     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
4164     soapencodingStyle=" " >  
4165  
4166     <soap:Header>  
4167  
4168         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">  
4169             <btp:context superior-type="atom">  
4170                 <btp:superior-address>  
4171                     <btp:binding>soap-http-1</btp:binding>  
4172                     <btp:binding-  
4173 address>http://client.example.com/soaphandler</btp:binding-  
4174 address>  
4175                     <btp:additional-information>btpengine</btp:additional-  
4176 information>  
4177                     </btp:superior-address>  
4178                     <btp:superior-identifier>1001</btp:superior-identifier>
```

```

4179         <btp:qualifiers>
4180             <btpq:transaction-timelimit
4181 xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers">1800</btpq:transact
4182 ion-timelimit>
4183         </btp:qualifiers>
4184     </btp:context>
4185 </btp:messages>
4186
4187 </soap:Header>
4188
4189 <soap:Body>
4190
4191     <ns1:orderGoods
4192 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4193         <custID>ABC8329045</custID>
4194         <itemID>224352</itemID>
4195         <quantity>5</quantity>
4196     </ns1:orderGoods>
4197
4198 </soap:Body>
4199
4200 </soap:Envelope>
4201
4202

```

4203 The example below shows CONTEXT\_REPLY and a related ENROL message sent from
 4204 services.example.com to client.example.com, in reply to the previous message. There is no
 4205 application response, so the BTP messages are in the SOAP Body. The ENROL message
 4206 does not contain the target-additional-information, since the grouping rules for
 4207 CONTEXT\_REPLY & ENROL omit the target address (the receiver of this example
 4208 remembers the superior address from the original CONTEXT)
 4209

```

4210 <soap:Envelope
4211     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
4212     soap:encodingStyle="">
4213
4214     <soap:Header>
4215     </soap:Header>
4216
4217     <soap:Body>
4218
4219         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
4220             <btp:relatedgroup>
4221                 <btp:context-reply>
4222                     <btp:superior-address>
4223                         <btp:binding>soap-http-1</btp:binding>
4224                         <btp:binding-address>
4225                             http://client.example.com/soaphandler
4226                         </btp:binding-address>
4227                         <btp:additional-information>
4228                             btpengine
4229                         </btp:additional-information>
4230                     </btp:superior-address>

```

```

4231     <btp:superior-identifier>1001</btp:superior-identifier>
4232     <completion-status>related</completion-status>
4233     </btp:context-reply>
4234
4235     <btp:enrol reply-requested="false">
4236         <btp:superior-identifier>
4237             1001
4238         </btp:superior-identifier>
4239         <btp:inferior-address>
4240             <btp:binding>soap-http-1</btp:binding>
4241             <btp:binding-address>
4242                 http://services.example.com/soaphandler
4243             </btp:binding-address>
4244         </btp:inferior-address>
4245         <btp:inferior-identifier>
4246             AAAB
4247         </btp:inferior-identifier>
4248     </btp:enrol>
4249
4250     </btp:relatedgroup>
4251
4252     </btp:messages>
4253
4254 </soap:Body>
4255
4256 </soap:Envelope>
4257
4258
4259

```

## 4260 SOAP + Attachments Binding

4261  
4262 This binding describes how BTP messages will be carried using SOAP as in the [SOAP](#)  
4263 [Messages with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-  
4264 http-1. The two bindings only differ when application messages are sent.

4265  
4266 **Binding name:** soap-attachments-http-1

4267  
4268 **Binding address format:** as for soap-http-1

4269  
4270 **BTP message representation:** As for soap-http-1

4271  
4272 **Mapping for BTP messages (unrelated):** As for “soap-http-1”, except the SOAP Envelope  
4273 containing the SOAP Body containing the BTP messages shall be in a MIME body part, as  
4274 specified in [SOAP Messages with Attachments](#) specification. If an application message is  
4275 being sent at the same time, the mapping for related messages for this binding shall be used,  
4276 as if the BTP messages were related to the application message(s).

4277  
4278 **Mapping for BTP messages related to application messages:** MIME packaging shall be  
4279 used. One of the MIME multipart/related parts shall contain a SOAP Envelope, whose SOAP

4280 Headers element shall contain precisely one `btm:messages` element, containing any BTP  
4281 messages. Any BTP CONTEXT in the `btm:messages` is considered to be related to the  
4282 application message(s) in the SOAP Body, and to also any of the MIME parts referenced  
4283 from the SOAP Body (using the “href” attribute).

4284

4285 **Implicit messages:** As for `soap-http-1`.

4286

4287 **Faults:** As for `soap-http-1`.

4288

4289 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding  
4290 string “`soap-http-1`” is considered to match one that has the binding string “`soap-`  
4291 `attachements-http-1`” if the binding address and additional information fields match.

4292

4293 **Limitations on BTP use:** None

4294

4295 **Other:** As for `soap-http-1`

4296

4297 *Example using SOAP + Attachments binding*

4298

```
4299 MIME-Version: 1.0
4300 Content-Type: Multipart/Related; boundary=MIME_boundary;
4301 type=text/xml;
4302         start="someID"
4303
4304 --MIME_boundary
4305 Content-Type: text/xml; charset=UTF-8
4306 Content-ID: someID
4307
4308 <?xml version='1.0' ?>
4309 <soap:Envelope
4310     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
4311     soap-
4312 env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
4313
4314     <soap:Header>
4315
4316         <btm:messages xmlns:btm="urn:oasis:names:tc:BTP:xml">
4317             <btm:context superior-type="atom">
4318                 <btm:superior-address>
4319                     <btm:binding>soap-http-1</btm:binding>
4320                     <btm:binding-address>
4321                         http://client.example.com/soaphandler
4322                     </btm:binding-address>
4323                     </btm:superior-address>
4324                     <btm:superior-identifier>1001</btm:superior-identifier>
4325                 </btm:context>
4326             </btm:messages>
4327
4328         </soap:Header>
4329
```

```

4330     <soap:Body>
4331         <orderGoods href="cid:anotherID" />
4332     </soap:Body>
4333
4334 </soap:Envelope>
4335
4336 --MIME_boundary
4337 Content-Type: text/xml
4338 Content-ID: anotherID
4339
4340     <ns1:orderGoods
4341 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4342     <custID>ABC8329045</custID>
4343     <itemID>224352</itemID>
4344     <quantity>5</quantity>
4345 </ns1:orderGoods>
4346
4347
4348 --MIME_boundary--
4349
4350

```

## XML Schema

```

4351
4352
4353 <?xml version="1.0"?>
4354 <schema targetNamespace="urn:oasis:names:tc:BTP:xml "
4355     xmlns="http://www.w3.org/2001/XMLSchema "
4356     xmlns:tns="urn:oasis:names:tc:BTP:xml ">
4357
4358     <complexType name="qualifier_type">
4359         <simpleContent>
4360             <extension base="string">
4361                 <attribute name="must-be-understood" type="boolean"/>
4362                 <attribute name="to-be-propagated" type="boolean"/>
4363             </extension>
4364         </simpleContent>
4365     </complexType>
4366     <element name="qualifier" type="tns:qualifier_type"/>
4367     <element name="qualifiers">
4368         <complexType>
4369             <sequence>
4370                 <element ref="tns:qualifier" maxOccurs="unbounded"/>
4371             </sequence>
4372         </complexType>
4373     </element>
4374
4375     <complexType name="address">
4376         <sequence>
4377             <element name="binding-name" type="string"/>
4378             <element name="binding-address" type="string"/>
4379             <element name="additional-information" type="string"
4380 minOccurs="0"/>
4381         </sequence>

```

```

4382     </complexType>
4383
4384     <simpleType name="identifier">
4385         <restriction base="string">
4386             <pattern value="([0-9,A-Z])*"/>
4387         </restriction>
4388     </simpleType>
4389
4390     <simpleType name="superior-type">
4391         <restriction base="string">
4392             <enumeration value="cohesion"/>
4393             <enumeration value="atom"/>
4394         </restriction>
4395     </simpleType>
4396
4397     <simpleType name="transaction-type">
4398         <restriction base="string">
4399             <enumeration value="cohesion"/>
4400             <enumeration value="atom"/>
4401         </restriction>
4402     </simpleType>
4403
4404     <element name="context">
4405         <complexType>
4406             <sequence>
4407                 <element name="superior-address" type="tns:address"
4408 maxOccurs="unbounded"/>
4409                 <element name="superior-identifier" type="tns:identifier"/>
4410                 <element ref="tns:qualifiers" minOccurs="0"/>
4411             </sequence>
4412             <attribute name="id" type="ID" use="optional"/>
4413             <attribute name="superior-type" type="tns:superior-type"
4414 use="required"/>
4415         </complexType>
4416     </element>
4417
4418     <element name="context-reply">
4419         <complexType>
4420             <sequence>
4421                 <element name="superior-address" type="tns:address"
4422 maxOccurs="unbounded"/>
4423                 <element name="superior-identifier" type="tns:identifier"/>
4424                 <element name="completion-status">
4425                     <simpleType>
4426                         <restriction base="string">
4427                             <enumeration value="completed"/>
4428                             <enumeration value="related"/>
4429                             <enumeration value="repudiated"/>
4430                         </restriction>
4431                     </simpleType>
4432                 </element>
4433                 <element ref="tns:qualifiers" minOccurs="0"/>
4434             </sequence>

```

```

4435         </sequence>
4436         <attribute name="id" type="ID"/>
4437         <attribute name="superior-type" type="tns:superior-type"
4438 use="required"/>
4439     </complexType>
4440 </element>
4441
4442     <element name="begin">
4443         <complexType>
4444             <sequence>
4445                 <element name="target-additional-information"
4446 type="string"/>
4447                 <element name="reply-address" type="tns:address"/>
4448                 <element ref="tns:qualifiers" minOccurs="0"/>
4449             </sequence>
4450             <attribute name="id" type="ID"/>
4451             <attribute name="transaction-type" type="tns:superior-type"
4452 use="required"/>
4453         </complexType>
4454     </element>
4455
4456     <element name="begun">
4457         <complexType>
4458             <sequence>
4459                 <element name="target-additional-information"
4460 type="string"/>
4461                 <element name="decider-address" type="tns:address"
4462 minOccurs="0"/>
4463                 <element name="transaction-identifier"
4464 type="tns:identifier" minOccurs="0"/>
4465                 <element name="inferior-handle" type="tns:identifier"
4466 minOccurs="0"/>
4467                 <element name="inferior-address" type="tns:address"
4468 minOccurs="0"/>
4469                 <element ref="tns:qualifiers" minOccurs="0"/>
4470             </sequence>
4471             <attribute name="id" type="ID"/>
4472             <attribute name="transaction-type" type="tns:superior-type"
4473 use="required"/>
4474         </complexType>
4475     </element>
4476
4477     <element name="enrol">
4478         <complexType>
4479             <sequence>
4480                 <element name="target-additional-information"
4481 type="string"/>
4482                 <element name="superior-identifier" type="tns:identifier"/>
4483                 <element name="reply-address" type="tns:address"
4484 minOccurs="0"/>
4485                 <element name="inferior-address" type="tns:address"
4486 minOccurs="1" maxOccurs="unbounded"/>
4487                 <element name="inferior-identifier" type="tns:identifier"/>

```

```

4488         <element ref="tns:qualifiers" minOccurs="0"/>
4489     </sequence>
4490     <attribute name="id" type="ID"/>
4491     <attribute name="reply-requested" type="boolean"/>
4492 </complexType>
4493 </element>
4494
4495
4496     <element name="enrolled">
4497         <complexType>
4498             <sequence>
4499                 <element name="target-additional-information"
4500 type="string"/>
4501                 <element name="inferior-identifier" type="tns:identifier"/>
4502                 <element name="inferior-handle" type="tns:identifier"
4503 minOccurs="0"/>
4504                 <element ref="tns:qualifiers" minOccurs="0"/>
4505             </sequence>
4506             <attribute name="id" type="ID"/>
4507         </complexType>
4508     </element>
4509
4510     <element name="resign">
4511         <complexType>
4512             <sequence>
4513                 <element name="target-additional-information"
4514 type="string"/>
4515                 <element name="superior-identifier" type="tns:identifier"/>
4516                 <element name="inferior-address" type="tns:address"
4517 minOccurs="1" maxOccurs="unbounded"/>
4518                 <element name="inferior-identifier" type="tns:identifier"/>
4519                 <element ref="tns:qualifiers" minOccurs="0"/>
4520             </sequence>
4521             <attribute name="id" type="ID"/>
4522             <attribute name="response-requested" type="boolean"/>
4523         </complexType>
4524     </element>
4525
4526     <element name="resigned">
4527         <complexType>
4528             <sequence>
4529                 <element name="target-additional-information"
4530 type="string"/>
4531                 <element name="inferior-identifier" type="tns:identifier"/>
4532                 <element ref="tns:qualifiers" minOccurs="0"/>
4533             </sequence>
4534             <attribute name="id" type="ID"/>
4535         </complexType>
4536     </element>
4537
4538     <element name="prepare">
4539         <complexType>
4540             <sequence>

```

```

4541         <element name="target-additional-information"
4542 type="string"/>
4543         <element name="inferior-identifier" type="tns:identifier"
4544 minOccurs="0"/>
4545         <element name="reply-address" type="tns:address"
4546 minOccurs="0"/>
4547         <element name="transaction-identifier"
4548 type="tns:identifier" minOccurs="0"/>
4549         <element name="inferiors-list" minOccurs="0">
4550             <complexType>
4551                 <sequence>
4552                     <element name="inferior-handle"
4553 type="tns:identifier" maxOccurs="unbounded"/>
4554                 </sequence>
4555             </complexType>
4556         </element>
4557         <element ref="tns:qualifiers" minOccurs="0"/>
4558     </sequence>
4559     <attribute name="id" type="ID"/>
4560 </complexType>
4561 </element>
4562
4563     <element name="prepared">
4564         <complexType>
4565             <sequence>
4566                 <element name="target-additional-information"
4567 type="string"/>
4568                 <element name="superior-identifier" type="tns:identifier"/>
4569                 <element name="inferior-address" type="tns:address"
4570 maxOccurs="unbounded"/>
4571                 <element name="inferior-identifier" type="tns:identifier"/>
4572                 <element ref="tns:qualifiers" minOccurs="0"/>
4573             </sequence>
4574             <attribute name="id" type="ID"/>
4575             <attribute name="default-is-cancel" type="boolean"/>
4576         </complexType>
4577     </element>
4578
4579     <element name="confirm">
4580         <complexType>
4581             <sequence>
4582                 <element name="target-additional-information"
4583 type="string"/>
4584                 <element name="inferior-identifier" type="tns:identifier"/>
4585                 <element ref="tns:qualifiers" minOccurs="0"/>
4586             </sequence>
4587             <attribute name="id" type="ID"/>
4588         </complexType>
4589     </element>
4590
4591     <element name="confirmed">
4592         <complexType>
4593             <sequence>

```

```

4594         <element name="target-additional-information"
4595 type="string"/>
4596         <element name="superior-identifier" type="tns:identifier"/>
4597         <element name="inferior-address" type="tns:address"
4598 minOccurs="0"/>
4599         <element name="inferior-identifier" type="tns:identifier"
4600 minOccurs="0"/>
4601         <element name="decider-address" type="tns:address"
4602 minOccurs="0"/>
4603         <element name="transaction-identifier"
4604 type="tns:identifier" minOccurs="0"/>
4605             <element ref="tns:qualifiers" minOccurs="0"/>
4606         </sequence>
4607         <attribute name="id" type="ID"/>
4608         <attribute name="confirmed-received" type="boolean"/>
4609     </complexType>
4610 </element>
4611
4612     <element name="cancel">
4613         <complexType>
4614             <sequence>
4615                 <element name="target-additional-information"
4616 type="string"/>
4617                 <element name="inferior-identifier" type="tns:identifier"
4618 minOccurs="0"/>
4619                 <element name="reply-address" type="tns:address"
4620 minOccurs="0"/>
4621                 <element name="transaction-identifier"
4622 type="tns:identifier" minOccurs="0"/>
4623                 <element name="decider-address" type="tns:address"
4624 minOccurs="0"/>
4625                 <element name="transaction-identifier"
4626 type="tns:identifier" minOccurs="0"/>
4627                 <element name="inferiors-list" minOccurs="0">
4628                     <complexType>
4629                         <sequence>
4630                             <element name="inferior-handle"
4631 type="tns:identifier" maxOccurs="unbounded"/>
4632                         </sequence>
4633                     </complexType>
4634                 </element>
4635                 <element ref="tns:qualifiers" minOccurs="0"/>
4636             </sequence>
4637             <attribute name="id" type="ID"/>
4638         </complexType>
4639     </element>
4640
4641     <element name="cancelled">
4642         <complexType>
4643             <sequence>
4644                 <element name="target-additional-information"
4645 type="string"/>
4646                 <element name="superior-identifier" type="tns:identifier"/>

```

```

4647         <element name="inferior-address" type="tns:address"
4648 maxOccurs="unbounded"/>
4649         <element name="inferior-identifier" type="tns:identifier"
4650 minOccurs="0"/>
4651         <element name="decider-address" type="tns:address"
4652 minOccurs="0"/>
4653         <element name="transaction-identifier"
4654 type="tns:identifier" minOccurs="0"/>
4655         <element ref="tns:qualifiers" minOccurs="0"/>
4656     </sequence>
4657     <attribute name="id" type="ID"/>
4658 </complexType>
4659 </element>
4660
4661     <element name="hazard">
4662         <complexType>
4663             <sequence>
4664                 <element name="target-additional-information"
4665 type="string"/>
4666                 <element name="superior-identifier" type="tns:identifier"/>
4667                 <element name="inferior-address" type="tns:address"
4668 maxOccurs="unbounded"/>
4669                 <element name="inferior-identifier" type="tns:identifier"/>
4670                 <element ref="tns:qualifiers" minOccurs="0"/>
4671             </sequence>
4672             <attribute name="id" type="ID"/>
4673         </complexType>
4674     </element>
4675
4676     <element name="contradiction">
4677         <complexType>
4678             <sequence>
4679                 <element name="target-additional-information"
4680 type="string"/>
4681                 <element name="inferior-identifier" type="tns:identifier"/>
4682                 <element ref="tns:qualifiers" minOccurs="0"/>
4683             </sequence>
4684             <attribute name="id" type="ID"/>
4685         </complexType>
4686     </element>
4687
4688     <element name="superior-state">
4689         <complexType>
4690             <sequence>
4691                 <element name="target-additional-information"
4692 type="string"/>
4693                 <element name="inferior-identifier" type="tns:identifier"/>
4694                 <element name="status">
4695                     <simpleType>
4696                         <restriction base="string">
4697                             <enumeration value="active"/>
4698                             <enumeration value="prepared-received"/>
4699                             <enumeration value="inaccessible"/>

```

```

4700         <enumeration value="unknown"/>
4701     </restriction>
4702 </simpleType>
4703 </element>
4704     <element ref="tns:qualifiers" minOccurs="0"/>
4705 </sequence>
4706 <attribute name="id" type="ID"/>
4707 <attribute name="reply-requested" type="boolean"/>
4708 </complexType>
4709 </element>
4710
4711 <element name="inferior-state">
4712     <complexType>
4713         <sequence>
4714             <element name="target-additional-information"
4715 type="string"/>
4716             <element name="superior-identifier" type="tns:identifier"/>
4717             <element name="inferior-address" type="tns:address"
4718 maxOccurs="unbounded"/>
4719             <element name="inferior-identifier" type="tns:identifier"/>
4720             <element name="status">
4721                 <simpleType>
4722                     <restriction base="string">
4723                         <enumeration value="active"/>
4724                         <enumeration value="prepared-received"/>
4725                         <enumeration value="inaccessible"/>
4726                         <enumeration value="unknown"/>
4727                     </restriction>
4728                 </simpleType>
4729             </element>
4730             <element ref="tns:qualifiers" minOccurs="0"/>
4731         </sequence>
4732 <attribute name="id" type="ID"/>
4733 <attribute name="reply-requested" type="boolean"/>
4734 </complexType>
4735 </element>
4736
4737 <element name="confirm-one-phase">
4738     <complexType>
4739         <sequence>
4740             <element name="target-additional-information"
4741 type="string"/>
4742             <element name="inferior-identifier" type="tns:identifier"/>
4743             <element ref="tns:qualifiers" minOccurs="0"/>
4744         </sequence>
4745 <attribute name="id" type="ID"/>
4746 <attribute name="report-hazard" type="boolean"/>
4747 </complexType>
4748 </element>
4749
4750 <element name="request-confirm">
4751     <complexType>
4752         <sequence>

```

```

4753         <element name="target-additional-information"
4754 type="string"/>
4755         <element name="reply-address" type="tns:address"/>
4756         <element name="transaction-identifier"
4757 type="tns:identifier"/>
4758         <element name="inferiors-list" minOccurs="0">
4759             <complexType>
4760                 <sequence>
4761                     <element name="inferior-handle"
4762 type="tns:identifier" maxOccurs="unbounded"/>
4763                 </sequence>
4764             </complexType>
4765         </element>
4766         <element ref="tns:qualifiers" minOccurs="0"/>
4767     </sequence>
4768     <attribute name="id" type="ID"/>
4769     <attribute name="report-hazard" type="boolean"/>
4770 </complexType>
4771 </element>
4772
4773     <element name="request-statuses">
4774         <complexType>
4775             <sequence>
4776                 <element name="target-additional-information"
4777 type="string"/>
4778                 <element name="reply-address" type="tns:address"/>
4779                 <element name="transaction-identifier"
4780 type="tns:identifier"/>
4781                 <element name="inferiors-list" minOccurs="0">
4782                     <complexType>
4783                         <sequence>
4784                             <element name="inferior-handle"
4785 type="tns:identifier" maxOccurs="unbounded"/>
4786                         </sequence>
4787                     </complexType>
4788                 </element>
4789                 <element ref="tns:qualifiers" minOccurs="0"/>
4790             </sequence>
4791             <attribute name="id" type="ID"/>
4792         </complexType>
4793     </element>
4794
4795     <element name="inferior-statuses">
4796         <complexType>
4797             <sequence>
4798                 <element name="target-additional-information"
4799 type="string"/>
4800                 <element name="decider-address" type="tns:address"/>
4801                 <element name="transaction-identifier"
4802 type="tns:identifier"/>
4803                 <element name="status-list">
4804                     <complexType>
4805                         <sequence>

```

```

4806         <element name="status-item" maxOccurs="unbounded">
4807     <complexType>
4808         <sequence>
4809             <element name="inferior-handle"
4810 type="tns:identifier"/>
4811             <element name="status">
4812                 <simpleType>
4813                     <restriction base="string">
4814                         <enumeration value="active"/>
4815                         <enumeration value="resigned"/>
4816                         <enumeration value="preparing"/>
4817                         <enumeration value="prepared"/>
4818                         <enumeration value="autonomously-confirmed"/>
4819                         <enumeration value="autonomously-cancelled"/>
4820                         <enumeration value="confirming"/>
4821                         <enumeration value="confirmed"/>
4822                         <enumeration value="cancelling"/>
4823                         <enumeration value="cancelled"/>
4824                         <enumeration value="cancel-contradiction"/>
4825                         <enumeration value="confirm-contradiction"/>
4826                         <enumeration value="hazard"/>
4827                     </restriction>
4828                 </simpleType>
4829             </element>
4830             <element ref="tns:qualifiers" minOccurs="0"/>
4831         </sequence>
4832     </complexType>
4833 </element>
4834 </sequence>
4835 </complexType>
4836 </element>
4837     <element ref="tns:qualifiers" minOccurs="0"/>
4838 </sequence>
4839     <attribute name="id" type="ID"/>
4840 </complexType>
4841 </element>
4842
4843     <element name="request-status">
4844         <complexType>
4845             <sequence>
4846                 <element name="target-additional-information"
4847 type="string"/>
4848                 <element name="reply-address" type="tns:address"/>
4849                 <element name="inferior-identifier" type="tns:identifier"
4850 minOccurs="0"/>
4851                 <element name="transaction-identifier"
4852 type="tns:identifier" minOccurs="0"/>
4853                 <element ref="tns:qualifiers" minOccurs="0"/>
4854             </sequence>
4855             <attribute name="id" type="ID"/>
4856         </complexType>
4857     </element>
4858

```

```

4859     <element name="status">
4860         <complexType>
4861             <sequence>
4862                 <element name="target-additional-information"
4863 type="string"/>
4864                 <element name="inferior-address" type="tns:address"
4865 minOccurs="0"/>
4866                 <element name="inferior-identifier" type="tns:identifier"
4867 minOccurs="0"/>
4868                 <element name="decider-address" type="tns:address"
4869 minOccurs="0"/>
4870                 <element name="transaction-identifier"
4871 type="tns:identifier" minOccurs="0"/>
4872                 <element name="status-value">
4873                     <simpleType>
4874                         <restriction base="string">
4875                             <enumeration value="created"/>
4876                             <enumeration value="enrolling"/>
4877                             <enumeration value="active"/>
4878                             <enumeration value="resigning"/>
4879                             <enumeration value="resigned"/>
4880                             <enumeration value="preparing"/>
4881                             <enumeration value="prepared"/>
4882                             <enumeration value="confirming"/>
4883                             <enumeration value="confirmed"/>
4884                             <enumeration value="cancelling"/>
4885                             <enumeration value="cancelled"/>
4886                             <enumeration value="cancel-contradiction"/>
4887                             <enumeration value="confirm-contradiction"/>
4888                             <enumeration value="hazard"/>
4889                             <enumeration value="contradicted"/>
4890                             <enumeration value="unknown"/>
4891                             <enumeration value="inaccessible"/>
4892                         </restriction>
4893                     </simpleType>
4894                 </element>
4895                 <element ref="tns:qualifiers" minOccurs="0"/>
4896             </sequence>
4897             <attribute name="id" type="ID"/>
4898         </complexType>
4899     </element>
4900
4901     <element name="redirect">
4902         <complexType>
4903             <sequence>
4904                 <element name="target-additional-information"
4905 type="string"/>
4906                 <element name="superior-identifier" type="tns:identifier"
4907 minOccurs="0"/>
4908                 <element name="inferior-identifier" type="tns:identifier"/>
4909                 <element name="old-address" type="tns:address"
4910 maxOccurs="unbounded"/>

```

```

4911         <element name="new-address" type="tns:address"
4912 maxOccurs="unbounded"/>
4913         <element ref="tns:qualifiers" minOccurs="0"/>
4914     </sequence>
4915     <attribute name="id" type="ID"/>
4916 </complexType>
4917 </element>
4918
4919 <element name="fault">
4920     <complexType>
4921         <sequence>
4922             <element name="target-additional-information"
4923 type="string"/>
4924             <element name="superior-identifier" type="tns:identifier"
4925 minOccurs="0"/>
4926             <element name="inferior-identifier" type="tns:identifier"
4927 minOccurs="0"/>
4928             <element name="fault-type" type="string"/>
4929             <element name="fault-data" type="anyType" minOccurs="0"/>
4930             <element ref="tns:qualifiers" minOccurs="0"/>
4931         </sequence>
4932         <attribute name="id" type="ID"/>
4933     </complexType>
4934 </element>
4935
4936 </schema>
4937

```

4937  
4938  
4939  
4940  
4941  
4942  
4943  
4944  
4945  
4946  
4947  
4948  
4949  
4950

## Conformance

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

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

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

Role Group	Role
Initiator/Terminator	Initiator
	Terminator
Cohesive Hub	Factory
	Composer (as Decider and Superior)
	Coordinator (as Decider and Superior)
	Sub-composer
	Sub-coordinator
Atomic Hub	Factory
	Coordinator
	Sub-coordinator
Cohesive Superior	Composer (as Superior only)
	Sub-Composer
	Coordinator (as Superior only)
	Sub-coordinator
Atomic Superior	Coordinator (as Superior only))
	Sub-coordinator
Participant	Inferior

Enroller

4951  
4952  
4953  
4954  
4955

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

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

4956  
4957  
4958  
4959  
4960  
4961  
4962  
4963  
4964  
4965

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

4965 Part 3. Appendices

4966

4967

4968

4969

4970 A. Glossary

4971

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

**Message** A datum which is produced and then consumed.

**Sender** The producer of a message.

**Receiver** The consumer of a message.

**Transmission** The passage of a message from a sender to a receiver.

**Endpoint** A sender or receiver.

**Address** An identifier for an endpoint.

Peer The other party in a two-party relationship, as in Superior to Inferior, or Sender to Receiver

**Carrier Protocol** A protocol which defines how transmissions occur.

**Carrier Protocol Address** The address of an endpoint for a particular carrier protocol.

(CPA)

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

(BTPA)

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

**Actor** An entity which executes procedures, a software agent.

**Application** An actor which uses the Business Transaction Protocol.

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

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

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

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

4972