

1 Organization for the Advancement of Structured Information Systems

# 2 Business Transaction Protocol

3  
4  
5 An OASIS Committee Specification

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

7  
8 Version 1.0 [0.9.0.2]

9 DD Mmm 2001 4 December 2001]

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 0.3b (not circulated)</i>	17 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 0.5 (uncirculated)</i>	8 August 2001
<i>Working draft 0.6 (State tables)</i>	31 August 2001
<i>Working draft 0.7 (revised abs msgs) – (not circulated)</i>	28 September 2001
<i>Working draft 0.72 (completed abs msg revsn 0.7)</i>	3 October 2001
<i>Working draft 0.80 – full scope, PRF handback</i>	18 October 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 November 2001

12  
13 **Change marks are relative to 0.9**

## 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

## 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  
102 infinitive form) in the Glossary are capitalized whenever the term used with that exact  
103 meaning, thus:

104  
105 Cancel  
106 Participant  
107 Application Message  
108

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

### 110 **Coordinator**

111  
112 Such words may be given in bold in other contexts (for example, in section headings or  
113 captions) to emphasize their status as formally defined terms.  
114  
115

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

117  
118 BEGIN  
119 CONTEXT  
120 RESIGN  
121

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

123  
124 BEGIN/atom  
125

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

127  
128 BEGIN/atom & CONTEXT  
129

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

131  
132 ENROL + VOTE  
133

134 XML schemata and instances are given in Courier:

135  
136 <ctp:begin> ... </ctp:begin>  
137

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

139  
140 **int main (String[] args)**  
141 **{**  
142 **}**  
143

144 Terms such as **MUST**, **MAY** and so on, which are defined in RFC [TBD number], “[TBD  
145 title]” are used with the meanings given in that document but are given in lowercase bold,  
146 rather than in upper-case:  
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>9</b>
158	Introduction .....	9
159	Development and Maintenance of the Specification .....	10
160	Overview of the Business Transaction Protocol.....	11
161	<b>Part 2. Normative Specification of BTP .....</b>	<b>14</b>
162	Actors, Roles and Relationships.....	14
163	Relationships .....	14
164	Roles involved in the Superior:Inferior relationship .....	16
165	Superior .....	16
166	Inferior .....	17
167	Enroller.....	18
168	Participant .....	19
169	Sub-coordinator.....	19
170	Sub-composer .....	20
171	Roles involved in the Terminator:Decider relationship .....	20
172	Decider.....	20
173	Coordinator .....	21
174	Composer.....	21
175	Terminator .....	21
176	Initiator .....	22
177	Factory .....	22
178	Other roles .....	23
179	Redirector.....	23
180	Status Requestor.....	23
181	Abstract Messages and Associated Contracts.....	24
182	Addresses.....	24
183	Request/response pairs.....	26
184	Compounding messages.....	26
185	Extensibility .....	27
186	Inferior handle .....	28
187	Messages .....	28
188	Qualifiers.....	28
189	CONTEXT.....	29
190	CONTEXT_REPLY .....	30
191	BEGIN .....	31
192	BEGUN.....	32
193	ENROL .....	33
194	ENROLLED .....	34
195	RESIGN .....	35
196	RESIGNED .....	36
197	PREP ARE.....	37

198	PREPARED.....	38
199	CONFIRM.....	40
200	CONFIRMED.....	40
201	CANCEL.....	42
202	CANCELLED.....	44
203	CONFIRM_ONE_PHASE.....	45
204	HAZARD.....	46
205	CONTRADICTION.....	47
206	SUPERIOR_STATE.....	48
207	INFERIOR_STATE.....	49
208	REQUEST_CONFIRM.....	51
209	REQUEST_STATUSES.....	53
210	INFERIOR_STATUSES.....	54
211	REQUEST_STATUS.....	55
212	STATUS.....	56
213	REDIRECT.....	58
214	FAULT.....	59
215	Standard qualifiers.....	61
216	Transaction timelimit.....	61
217	Inferior timeout.....	61
218	Minimum inferior timeout.....	62
219	Inferior name.....	63
220	State Tables.....	65
221	Explanation of the state tables.....	65
222	Status queries.....	65
223	Decision events.....	65
224	Disruptions – failure events.....	66
225	Invalid cells and assumptions of the communication mechanism.....	66
226	Meaning of state table events.....	67
227	Persistent information.....	70
228	Failure Recovery.....	84
229	Types of failure.....	84
230	Persistent information.....	85
231	Redirection.....	86
232	Terminator:Decider failures.....	87
233	XML representation of Message Set.....	87
234	Addresses.....	87
235	Qualifiers.....	88
236	Identifiers.....	88
237	Message References.....	89
238	Messages.....	89
239	CONTEXT.....	89
240	CONTEXT -REPLY.....	89
241	BEGIN.....	89
242	BEGUN.....	90
243	ENROL.....	90
244	ENROLLED.....	90
245	RESIGN.....	91

246	RESIGNED .....	91
247	PREPARE.....	91
248	PREPARED.....	92
249	CONFIRM.....	92
250	CONFIRMED.....	92
251	CANCEL.....	93
252	CANCELLED.....	93
253	HAZARD .....	94
254	CONTRADICTION.....	94
255	SUPERIOR_STATE.....	94
256	INFERIOR_STATE.....	95
257	CONFIRM_ONE_PHASE.....	95
258	REQUEST_CONFIRM.....	95
259	REQUEST_STATUSES.....	96
260	INFERIOR_STATUSES.....	96
261	REQUEST_STATUS .....	97
262	STATUS.....	97
263	REDIRECT.....	98
264	FAULT.....	98
265	Standard qualifiers.....	99
266	Transaction timelimit .....	99
267	Inferior timeout .....	99
268	Minimum inferior timeout.....	99
269	Compounding of Messages.....	100
270	Carrier Protocol Bindings .....	101
271	Carrier Protocol Binding Proforma .....	101
272	SOAP Binding.....	102
273	Example scenario using SOAP binding .....	104
274	SOAP + Attachments Binding .....	106
275	XML Schema for SOAP Bindings .....	107
276	Conformance.....	120
277	<b>Part 3. Appendices .....</b>	<b>122</b>
278	A. Glossary .....	122
279		
280		



# 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.

## 323 **Development and Maintenance of the Specification**

324  
325 For more information on the genesis and development of BTP, please consult the OASIS BT  
326 Technical Committee's website, at

327  
328 <http://www.oasis-open.org/committees/business-transactions/>  
329

330  
331 As of the date of adoption of this specification the OASIS BT Technical Committee is still in  
332 existence, with the charter of

- 333
- 334  maintaining the specification in the light of implementation experiences
- 335
- 336  coordinating publicity for BTP
- 337
- 338  liaising with other standards bodies whose work affects or may be affected by
- 339 BTP
- 340
- 341  reviewing the appropriate time, in the light of implementation experience and
- 342 user support, to put BTP forward for adoption as a full OASIS standard
- 343
- 344

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

347  
348 [bt-spec@lists.oasis-open.org](mailto:bt-spec@lists.oasis-open.org)  
349

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

352  
353 The main list of the committee is:

354  
355 [business-transaction@lists.oasis-open.org](mailto:business-transaction@lists.oasis-open.org)  
356  
357  
358  
359  
360  
361

## 361 Overview of the Business Transaction Protocol

362  
363 A Business Transaction is a consistent change in the state of a business relationship between  
364 two or more parties. BTP provides means to allow the consistent and coordinated changes in  
365 the relationship as viewed from each party.

366  
367 BTP assumes that for a given business transaction state changes occur, or are desired, in some  
368 set of parties, and that these changes are related in some business-defined manner.

369  
370 Typically business-defined messages (“application messages”) are exchanged between the  
371 parties to the transaction, which result in the performance of some set of operations. These  
372 operations create provisional or tentative state changes (the transaction’s effect). The  
373 provisional changes of each party must either be confirmed (given final effect), or must be  
374 cancelled (counter-effected). Those parties which are confirmed create an atomic unit, within  
375 which the business transaction ~~has~~ should have a consistent final effect.

376  
377 The meaning of “effect”, “final effect” and “counter-effect” is specific to each business  
378 transaction and to each party’s role within it. A party may log intended changes (as its effect)  
379 and only process them as visible state changes on confirmation (its final effect). Or it may  
380 make visible state changes and store the information needed to cancel (its effect), and then  
381 simply delete the information needed for cancellation (its final effect). A counter-effect may  
382 be a precise inversion or removal of provisional changes, or it may be the processing of  
383 operations that in some way compensate for, make good, alleviate or supplement their effect.  
384

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

- 387
- 388  determine whether it is able both to cancel (counter-effect) and to confirm (give final  
389 effect to) its effect
- 390
- 391  report its ability or inability to cancel-or-confirm (its preparedness) to a central  
392 coordinating entity
- 393

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

396  
397 Such a two-phase exchange (ask, instruct) mediated by a central coordinator is required to  
398 achieve a consistent outcome for a set of operations. BTP defines the means for software  
399 agents executing on network nodes to interoperate using a two-phase coordination protocol,  
400 leading either to the abandonment of the entire attempted transaction, or to the selection of an  
401 internally consistent set of confirmed operations.

402  
403 BTP centres on the bilateral relationship between the computer systems of the coordinating  
404 entity and those of one of the parties in the overall business transaction. In that relationship a  
405 software agent within the coordinating entity’s systems plays the BTP role of Superior for a  
406 given transaction and one or more software agents within the systems of the party play the  
407 BTP role of Inferior. Each Inferior has one Superior, therefore, while a single Superior may

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

410  
411 An Inferior is associated with some set of operation invocations that creates effect  
412 (provisional or tentative changes) within the party, for a given business transaction. The  
413 Inferior is responsible for reporting to its related Superior whether its associated operations'  
414 effect can be confirmed/cancelled. A Superior is responsible for gathering the reports of all of  
415 its Inferiors, in order to ascertain which should be cancelled or confirmed. For example, if a  
416 Superior is acting as an atomic Coordinator it will treat any Inferior which cannot prepare to  
417 cancel/confirm as having veto power over the whole business transaction, causing the  
418 Superior to instruct all its Inferiors to cancel. A Superior may, under the dictates of a  
419 controlling application, increase or reduce the set of Inferiors to which a common confirm or  
420 cancel outcome may be delivered. Thus, the set of prepared Inferiors may be larger than the  
421 set of confirmed Inferiors.

422  
423 An Inferior:Superior relationship is typically established in relation to one or more  
424 application messages sent from one part of the application (linked to the Superior) to some  
425 other part of the application to request the performance of operations that are to be subject to  
426 the confirm or cancel decision of the Superior. If an application is divided between a client  
427 and a service, which use RPCs to communicate application requests and responses, then the  
428 client would typically be associated with the Superior and the service would typically host the  
429 Inferior(s). (BTP does not mandate such an application topology nor does it require the use of  
430 RPC or any other application communication paradigm.)

431  
432 BTP defines a CONTEXT message that can be sent "in relation to" such application  
433 messages. On receipt of a CONTEXT, one or more Inferiors **are may be** created and  
434 "enrolled" with the Superior, establishing the Superior:Inferior relationships. The particular  
435 mechanisms by which a CONTEXT is "related" to application **messages** is an issue for the  
436 application protocol and its binding to carrier mechanisms. BTP does not require that the  
437 enrolment is requested by any particular entity – in a particular implementation this may be  
438 done by the Inferior itself, by parts of the application or by other entities involved in the  
439 transmission of the CONTEXT and the application messages. BTP defines a  
440 CONTEXT\_REPLY message that can be sent on the return path of the CONTEXT to indicate  
441 whether the enrolment was successful. Without CONTEXT\_REPLY it would be possible for  
442 a Superior to have an incorrect view of which Inferiors it was supposed to involve in its  
443 confirm decision.

444  
445 It should be noted that this BTP specification recognises that:

- 446     ❑ an Inferior may itself be a Superior to other BTP Inferiors; this occurs when some of  
447     the operations associated with the Inferior involve other application elements whose  
448     operations are to be subject to the confirm/cancel instruction sent to the Inferior. The  
449     specification treats any lower Inferiors as part of the associated operations;
- 450     ❑ the requirement on an Inferior to be able to confirm or cancel does not include any  
451     specific mechanism to determine the isolation of the effects of operations; the  
452     requirement is only that the Inferior is able to confirm or cancel the operations, as  
453     their effects are known to the Superior and the application directly in contact with the  
454     Superior. Thus the confirm-or-cancel requirement may be achieved by performing all  
455     the operations and remembering a compensating counter operation (that will be

456 triggered by a cancel order); or by remembering the operations (having checked they  
457 are valid) and performing them only if a confirm order is received; or by forbidding  
458 any other access to data changed by the operations and releasing them in their  
459 unchanged state (if cancelled) or their changed state (if confirmed); or by various  
460 combinations of these. In addition, a cancellation may not return data to their original  
461 state, but only to a state accepted by the application as appropriate to a cancelled  
462 operation.  
463  
464  
465  
466  
467  
468  
469

## 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);

512

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

515

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

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

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

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

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

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

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

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

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

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

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

538

539           There are other relationships that are secondary to Terminator:Decider, Superior:Inferior,  
540           mostly involved in the establishment of the primary relationships.

541

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

545

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

549

550           2. The Superior:Inferior relationship is recoverable – depending on the progress of the  
551           relationship, the two sides will re-establish their shared state after failure; the  
552           Terminator:Decider relationship is not recoverable

553

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

558  
559 In the following sections, the responsibility of each role is defined, and the messages that are  
560 sent or received by that role are listed. Note that some roles exist only to have a name for an  
561 actor that issues a message and receives a reply to that message. Some of these roles may be  
562 played by several actors in the course of a single business transaction.  
563

## 564 **Roles involved in the Superior:Inferior relationship**

### 565 **Superior**

566  
567  
568 Accepts enrolments from Inferiors, establishing a Superior:Inferior relationship with each. In  
569 cooperation with other actors and constrained by the messages exchanged with the Inferior,  
570 the Superior determines the **Outcome** applicable to the Inferior and informs the Inferior by  
571 sending CONFIRM or CANCEL. This outcome can be confirm only if a PREPARED  
572 message is received from the Inferior, and if a record, identifying the Inferior can be  
573 persisted. (Whether this record is also a record of a confirm decision depends on the  
574 Superior's position in the business transaction as a whole.). The Superior must retain this  
575 persistent record until it receives a CONFIRMED (or, in exceptional cases, CANCELLED or  
576 HAZARD) from the Inferior.

577  
578 A Superior may delegate the taking of the confirm or cancel decision to an Inferior, if there is  
579 only one Inferior, by sending CONFIRM\_ONE\_PHASE.

580  
581 A Superior may be *Atomic* or ~~Cohesive~~. ~~An~~ *Cohesive*; an Atomic Superior will apply the same  
582 decision to all of its Inferiors; a Cohesive Superior ~~can~~ *may* apply confirm *to* some Inferiors  
583 and cancel *to* others, or may confirm some after others have reported cancellation. The set of  
584 Inferiors that the Superior confirms (or attempts to confirm) is called the "confirm-set".

585  
586 If RESIGN is received from an Inferior, the Superior:Inferior relationship is ended; the  
587 Inferior has no further effect on the behaviour of the Superior as a whole.

588  
589 A Superior receives

590  
591 ENROL

592  
593 to enrol a new Inferior, establishing a new Superior:Inferior relationship.

594  
595 A Superior sends

596  
597 ENROLLED

598  
599 in reply to ENROL, if the appropriate parameter on the ENROL asked for the reply.

600  
601 A Superior sends



602  
603           PREPARE  
604           CONFIRM  
605           CANCEL  
606           RESIGNED  
607           CONFIRM\_ONE\_PHASE  
608           SUPERIOR\_STATE  
609

610 to an enrolled Inferior.

611  
612 A Superior receives

613  
614           PREPARED  
615           CANCELLED  
616           CONFIRMED  
617           HAZARD  
618           RESIGN  
619           INFERIOR\_STATE  
620

621 from an enrolled Inferior.

## 622 623 Inferior

624  
625 Responsible for applying the Outcome to some set of associated operations – the application  
626 determines which operations are the responsibility of a particular Inferior.

627  
628 An Inferior is **Enrolled** with a single Superior (hereafter referred to as “its Superior”),  
629 establishing a Superior:Inferior relationship. If the Inferior is able to ensure that either a  
630 confirm or cancel decision can be applied to the associated operations, and can persist  
631 information to retain that condition, it sends a PREPARED message to the Superior. When  
632 the Outcome is received from the Superior, the Inferior applies it, deletes the persistent  
633 information, and replies with CANCELLED or CONFIRMED as appropriate.

634  
635 If an Inferior is unable to come to a prepared state, it cancels the associated operations and  
636 informs the Superior with a CANCELLED message. If it is unable to either come to a  
637 prepared state, or to cancel the associated operations, it informs the Superior with a  
638 HAZARD message.

639  
640 An Inferior that has become prepared may, exceptionally, make an autonomous decision, to  
641 be applied to the associated operations, without waiting for the Outcome from the Superior. It  
642 is required to persist this autonomous decision and report it to the Superior with  
643 CONFIRMED or CANCELLED as appropriate. If, when CONFIRM or CANCEL is  
644 received, the autonomous decision and the decision received from the Superior are  
645 contradictory, the Inferior must retain the record of the autonomous decision until receiving a  
646 CONTRADICTION message.

647  
648 An Inferior receives  
649

650                   PREPARE  
651                   CONFIRM  
652                   CANCEL  
653                   RESIGNED  
654                   CONFIRM\_ONE\_PHASE  
655                   SUPERIOR\_STATE

656  
657                   from its Superior.

658  
659                   An Inferior sends

660                     
661                   PREPARED  
662                   CANCELLED  
663                   CONFIRMED  
664                   HAZARD  
665                   RESIGN  
666                   INFERIOR\_STATE

667  
668                   to its Superior.

669  
670                   An Inferior receives REQUEST\_STATUS and replies with STATUS. If it is also a Superior,  
671                   the STATUS concerns the Inferior as a whole.

## 672                   Enroller

673  
674  
675                   Causes the enrolment of an Inferior with a Superior. This role is distinguished because in  
676                   some implementations the enrolment request will be performed by the application, in some  
677                   the application will ask the actor that will play the role of Inferior to enrol itself, and a  
678                   Factory may enrol a new Inferior (which will also be Superior) as a result of receiving  
679                   BEGIN&CONTEXT.

680  
681                   An Enroller sends

682                     
683                   ENROL

684  
685                   to a Superior.

686  
687                   An Enroller receives

688                     
689                   ENROLLED

690  
691                   in reply to ENROL if the Enroller asked for a response when the ENROL was sent.

692  
693                   An ENROL message sent from an Enroller that did not require an ENROLLED response may  
694                   be modified *en route* to the Superior by an intermediate actor to ask for an ENROLLED  
695                   response to be sent to the intermediate. (This may occur in the “one-shot” scenario, where an  
696                   ENROL/no-rsp-req is received in relation to a CONTEXT\_REPLY/related; the receiver of  
697                   the CONTEXT\_REPLY will need to ensure the enrolment is successful).

698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
  
715  
  
716  
717  
718  
719  
  
720  
721  
722  
  
723  
724  
725  
  
726  
727  
  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740

## Participant

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

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

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

---

Note – Possible approaches are:

---

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

## Sub-coordinator

An Inferior which is also an Atomic Superior.

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

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

741 A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and  
742 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is  
743 propagated to all Inferiors.  
744

### 745 **Sub-composer**

746 An Inferior which is also a Cohesive Superior.  
747

748 Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from  
749 the perspective of its Superior.  
750

751 A sub-composer is similar to a sub-coordinator, except that the constraints linking the  
752 different Inferiors concern only those Inferiors in the confirm-set. How the confirm-set is  
753 controlled, and when<sub>u</sub> is not defined in this specification. |

754  
755  
756 If the sub-composer is instructed to cancel, by receiving a CANCEL message from its  
757 Superior, the cancellation is propagated to all its Inferiors.  
758  
759

## 760 **Roles involved in the Terminator:Decider relationship**

### 761 **Decider**

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

768  
769  
770 A Decider is instructed to cancel by receiving CANCEL/whole.  
771

772  
773 A Decider that is an Atomic Superior (all Inferiors will have the same outcome) is a  
774 Coordinator. A Decider that is a Cohesive Superior (some Inferiors may cancel, some  
775 confirm) is a Cohesion.  
776

777 All Deciders receive  
778 REQUEST\_CONFIRM  
779 CANCEL/whole  
780 REQUEST\_STATUSES  
781

782 All Deciders send  
783 CONFIRMED  
784 CANCELLED  
785 INFERIOR\_STATUSES  
786

787 A<sub>n</sub> Decider also receives REQUEST\_STATUS and replies with STATUS, reporting its state |  
788 as a whole.

789

790

## Coordinator

791

792

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

793

794

795

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

796

797

A Coordinator must make a cancel decision if

798

it is instructed to cancel by the Terminator

799

if CANCELLED is received from any Inferior

800

if it is unable to persist a confirm decision

801

802

## Composer

803

804

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

805

806

807

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

808

809

810

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

811

it is instructed to cancel by the Terminator

812

if CANCELLED is received from any Inferior in the confirm-set

813

if it is unable to persist a confirm decision

814

815

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

816

817

818

819

820

A Composer may be asked to cancel some of its Inferiors, but not itself, by receiving CANCEL/inferiors.

821

822

823

In addition to the messages received by the Composer as a Decider, it receives

824

PREPARE

825

CANCEL/inferiors

826

827

## Terminator

828

829

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

830

831

832

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

833

834

835

A request to confirm is made by sending REQUEST\_CONFIRM to the target Decider. If the Decider is a Cohesion Composer, the Terminator may select which of the Composer's

836

837 Inferiors are to be included in the confirm-set. If the Decider is an Atom Coordinator, all  
838 Inferiors are included. After applying the decision, the Decider replies with CONFIRMED,  
839 CANCELLED or (in the case of problems) INFERIOR\_STATUSES.  
840

841 A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its  
842 Inferiors with PREPARE/inferiors. The Composer replies with INFERIOR\_STATUSES.  
843

844 A Terminator may send CANCEL to instruct the Decider to cancel the whole business  
845 transaction, or, if it is a Cohesion Composer, some of its Inferiors. The Decider replies with  
846 CANCELLED, or for a selective cancel or in the case of problems, INFERIOR\_STATUSES.  
847

848 A Terminator may check the status of the Inferiors of the Decider by sending  
849 REQUEST\_STATUSES. The Decider replies with INFERIOR\_STATUSES.  
850

851 A Terminator sends  
852     REQUEST\_CONFIRM  
853     CANCEL  
854     PREPARE/inferiors  
855     REQUEST\_STATUSES  
856

857 A Terminator receives  
858     CONFIRMED  
859     CANCELLED  
860     INFERIOR\_STATUSES  
861

862 **Initiator**  
863

864 Requests a **Factory** to create a Superior – this will either be a Decider (representing a new  
865 top-level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an  
866 existing business transaction.  
867

868 An Initiator sends  
869  
870     BEGIN  
871     BEGIN & CONTEXT  
872

873 to a Factory, and receives in reply  
874  
875     BEGUN & CONTEXT  
876

877 **Factory**  
878

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

881  
882     Decider, which **mayis** either  
883         Composer or  
884         Coordinator

885 Sub-composer  
886 Sub-coordinator

887  
888 A Factory receives

889 BEGIN  
890 BEGIN & CONTEXT  
891

892  
893 and replies with

894  
895 BEGUN & CONTEXT  
896

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

900  
901 If the BEGIN has a related CONTEXT, the new Superior is also enrolled as an Inferior of the  
902 Superior identified by the CONTEXT. The new Superior is thus a sub-composer or sub-  
903 coordinator, as determined by the “superior type” parameter on the BEGIN.

904  
905  
906

## 907 **Other roles**

908

### 909 **Redirector**

910

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

914

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

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

922

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

924

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

927

### 928 **Status Requestor**

929

930 Requests and receives the current status of an Inferior or a Decider. The role of Status  
931 Requestor has no responsibilities – it is just a name for where the REQUEST\_STATUS  
932 comes from.

933  
934 A Status Requestor sends  
935  
936 REQUEST\_STATUS  
937  
938 and receives  
939  
940 STATUS  
941  
942 in response.

943  
944 The information returned will always relate to the actor concerned in its role as an Inferior,  
945 even if it [is](#) also a Superior.  
946

## 947 **Abstract Messages and Associated Contracts**

948  
949 BT Protocol Messages are defined in this section in terms of the abstract information that has  
950 to be communicated. These abstract messages will be mapped to concrete messages  
951 communicated by a particular carrier protocol (there can be several such mappings defined).

952  
953 The abstract message set and the associated state table assume the carrier protocol will

- 954  
955  deliver messages completely and correctly, or not at all (corrupted messages will  
956 not be delivered);
- 957  
958  report some communication failures, but will not necessarily report all (i.e. not all  
959 message deliveries are positively acknowledged within the carrier);
- 960  
961  sometimes deliver successive messages in a different order than they were sent;

962  
963 and

- 964  
965  does not have built-in mechanisms to link a request and a response  
966

967  
968 Note that these assumptions would be met by a mapping to SMTP and more than met by  
969 mappings to SOAP/[HTTP](#).

970  
971 However, when the abstract message set is mapped to a carrier protocol that provides a richer  
972 service (e.g. reports all delivery failures, guarantees ordered delivery or offers a  
973 request/response mechanism), the mapping can take advantage of these features. Typically in  
974 such cases, some of the parameters of an abstract message will be implicit in the carrier  
975 mechanisms, while the values of other parameters will be directly represented in transmitted  
976 elements.  
977

## 978 **Addresses**

979



980 All of the messages except CONTEXT and CONTEXT\_REPLY have a “target address”  
981 parameter and many also have other address parameters. These latter identify the desired  
982 target of other messages in the set. In all cases, the exact value will invariably have been  
983 originally determined by the implementation that is the target or desired future target.  
984

985 The detailed format of the address will depend on the particular carrier protocol, but at this  
986 abstract level is considered to have three parts. The first part, the “binding name”, identifies  
987 the binding to a particular carrier protocol – some bindings are specified in this document,  
988 others can be specified elsewhere. The second part of the address, the “binding address”, is  
989 meaningful to the carrier protocol itself, which will use it for the communication (i.e. it will  
990 permit a message to be delivered to a receiver). The third part, “additional information”, is  
991 not used or understood by the carrier protocol. The “additional information” may be a  
992 structured value.  
993

994 When a message is actually transmitted, the “binding name” of the target address will identify  
995 which carrier protocol is in use and the “binding address” will identify the destination, as  
996 known to the carrier protocol. The entire binding address is considered to be “consumed” by  
997 the carrier protocol implementation. All of it may ~~may~~ be used by the sending  
998 implementation, or some of it may be transmitted in headers, or as part of a URL in the  
999 carrier protocol, but then used or consumed by the receiving implementation of the carrier  
1000 protocol to direct the BTP message to a BTP-aware entity (BTP-aware in that it is capable of  
1001 interpreting the BTP messages). The “additional information” of the target address will be  
1002 part of the BTP message itself and used in some way by the receiving BTP-aware entity (it  
1003 could be used to route the message on to some other BTP entity). Thus, for the target address,  
1004 only the “additional information” field is transmitted in the BTP message and the “additional  
1005 information” is opaque to parties other than the recipient.  
1006

1007 For other addresses in BTP messages, all three components will be within the message.  
1008

1009 All messages that concern a particular Superior:Inferior relationship have an identifier  
1010 parameter for the target side as well as the compound target address. This allows full  
1011 flexibility for implementation choices – an implementation can:  
1012

- 1013 a) Use the same binding address and additional information for multiple business  
1014 transactions, using the identifier parameter to locate the relevant state  
1015 information;
- 1016 b) Use the same binding address for multiple business transactions and use the  
1017 additional information to locate the information; or
- 1018 c) Use a different binding address for each business transaction.  
1019

1020 Which of these choices is used is opaque to the entity sending the message – both parts of the  
1021 address and the identifier originated at the recipient of this message (and were transmitted as  
1022 parameters of earlier messages in the opposite direction). In cases b) and c), the identifier is to  
1023 some extent redundant, although interoperation requires that it always be present.  
1024

1025 BTP recovery requires that the state information for a Superior or Inferior is accessible after  
1026 failure and that the peer can distinguish between temporary inaccessibility and the permanent  
1027 non-existence of the state information. As is explained in “Redirection” below, BTP provides

1028 mechanisms – having a set of BTP addresses for some parameters, and the REDIRECT  
1029 message – that make this possible, even if the recovered state information is on a different  
1030 address to the original one (as may be the case if case c) above is used).

1031  
1032

### 1033 **Request/response pairs**

1034

1035 Many of the messages combine in pairs as a request and its response. However, in some cases  
1036 the response message is sent without a triggering request, or as a possible response to more  
1037 than one type of request. To allow for this, the abstract message set treats each message as  
1038 standalone; but where a request does expect a reply, a “reply-address” parameter will be  
1039 present. For any message with a reply address parameter, in the case of certain errors, a  
1040 FAULT message will be sent to the reply address instead of the expected reply.

1041

1042 For messages which are specified as sent between Superior and Inferior, a FAULT message is  
1043 sent to the peer.

1044

### 1045 **Compounding messages**

1046

1047 BTP messages may be sent in combination with each other, or with other (application)  
1048 messages. There are two cases:

1049

1050 a) Sending the messages together has semantic significance. One message is said  
1051 to be “related to” the other.

1052

1053 b) Sending of the messages has no semantic significance, but is merely a  
1054 convenience or optimisation. This is termed “bundling”.

1054

1055 The form A&B is used to refer to a combination where message B is sent in relation to A  
1056 (“relation” is asymmetric). The form A+B is used to refer to A and B bundled together- the  
1057 transmission of the bundle "A+B" is semantically identical to the transmission of A followed  
1058 by the transmission of B.

1059

1060 In both cases the messages will have the same binding address, but may have different  
1061 “additional information” values. Unless constrained by the binding, any messages that are to  
1062 be sent to the same binding address may be bundled – the fact that the binding addresses are  
1063 the same is a necessary and sufficient condition for the sender to determine that the messages  
1064 can be bundled.

1065

1066 A particular and important case of related messages is where a BTP CONTEXT message is  
1067 sent related to an application message. In this case, the target of the application message  
1068 defines the destination of the CONTEXT message. The receiving implementation may in fact  
1069 remove the CONTEXT before delivering the application message to the application (Service)  
1070 proper, but from the perspective of the sender, the two are sent to the same place.

1071

1072 The compounding mechanisms, and the multi-part address structures, support the “one-wire”  
1073 and “one-shot” communication patterns.

1073

1074 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,  
1075 including the associated application messages, pass via the same “endpoints”. These

1076 “endpoints” may in fact be relays, routing messages on to particular actors within their  
1077 domain. The onward routing will require some further addressing, but this has to be opaque to  
1078 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors  
1079 in its domain have the relays address as their binding address, and any routing information it  
1080 will need in its own domain is placed in the additional information. (This may involve the  
1081 relay changing addresses in messages as they pass through it on the way out). On receiving a  
1082 message, it determines the within-domain destination from the received additional  
1083 information (which is thus rewritten) and forwards the message appropriately. The sender is  
1084 unaware of this, and merely sees addresses with the same binding address, which it is  
1085 permitted to bundle. The content of the “additional information” is a matter only for the relay  
1086 – it could put an entire BTP address in there, or other implementation-defined information.  
1087 Note that a quite different one-wire implementation can be constructed where there is no  
1088 relaying, but the receiving entity effectively performs all roles, using the received identifiers  
1089 to locate the appropriate state.

1091 “One-shot” communication concerns the bundling of application messages, especially where  
1092 the application uses a request/response paradigm. The application request is sent with a  
1093 related CONTEXT message. The application response is sent with a related  
1094 CONTEXT\_REPLY/related, with an ENROL/no-rsp-req message and a bundled  
1095 PREPARED message (assuming the operations succeeded and the Inferior has decided to be  
1096 prepared). The target address of the ENROL and PREPARED (the Superior address) must  
1097 have a binding address that is the same as the target address of the application response (i.e.  
1098 the reply address for the client, as perceived by the Service) – otherwise the Service cannot  
1099 determine that ~~is~~it should bundle the messages together. One-shot is thus a  
1100 ~~specialisation~~specialization of one-wire.

1101  
1102 With “one-shot”, if there are multiple Inferiors created as a result of a single application  
1103 message, there is an ENROL and PREPARED message for each sent with the application  
1104 response and the CONTEXT\_REPLY. If an operation fails, a CANCELLED message can be  
1105 sent with the response instead of a PREPARED. If subsequent messages to the same Service,  
1106 with the same related CONTEXT, have their associated operations put under the control of  
1107 the same Inferior, only a CONTEXT\_REPLY/completed is sent back with the response (if the  
1108 new operations fail, it will be necessary to send back CONTEXT\_REPLY/repudiated, or send  
1109 CANCELLED).

1111 Where does that last bit on one-shot, one-wire belong. It needs to be in somewhere.  
1112 prf

## 1114 Extensibility

1115  
1116 To simplify interoperation between implementations of this edition of BTP with  
1117 implementations of future editions, the “must-be-understood” sub-parameter as specified for  
1118 Qualifiers may be defined for use with any parameter added to an existing message in a future  
1119 revision of this specification. The default for “must-be-understood” shall be “true”, so an  
1120 implementation receiving an unrecognised parameter without a “false” value for “must-be-  
1121 understood” shall not accept it (the FAULT value “UnrecognisedParameter” is available, but  
1122 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If

1123 “must-be-understood” with the value “false” is present as a sub-parameter of a parameter in  
1124 any message, a receiving implementation **should** ignore the parameter.  
1125  
1126 How the sub-parameter is associated with the new parameter is determined by the particular  
1127 binding.  
1128  
1129 No special mechanism is provided to allow for the introduction of completely new messages.  
1130

### 1131 **Inferior handle**

1132  
1133 Some of the messages exchanged between a Terminator and a Decider are concerned with the  
1134 individual Inferiors enrolled with the Decider, and not with the business transaction as a  
1135 whole. These messages distinguish the Inferiors of Decider using an “inferior handle”. This is  
1136 created by the Decider and is unambiguous within the scope of the Decider .  
1137

1138 The “inferior handle” is distinct from the “inferior identifier” passed on an ENROL message  
1139 (among other places). The latter is created by the Inferior (or its enroller) and is required to be  
1140 unambiguous within the scope of the address-as-inferior on the ENROL (and unambiguous  
1141 within **any** of the individual addresses in that set of BTP addresses - the identifier must  
1142 identify the Inferior across all the places it might migrate to or that have recovery  
1143 responsibility for it).  
1144

1145 The “inferior handle” is only used by the Terminator to refer to the inferiors of the Decider.  
1146 In messages between the Decider and its Inferiors, the address-as-inferior and inferior  
1147 identifier are used.  
1148

### 1149 **Messages**

1150

#### 1151 **Qualifiers**

1152

1153 All messages have a Qualifiers parameter which contains zero or more Qualifier values. A  
1154 Qualifier has sub-parameters:  
1155

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

1156

1157 **Qualifier group** ensures the Qualifier name is unambiguous. Qualifiers in the  
1158 same group need not have any functional relationship. The qualifier group will  
1159 typically be used to identify the specification that defines the qualifier’s meaning  
1160 and use. Qualifiers may be defined in this or other standard specifications, in

1161 specifications of a particular community of users or of implementations or by  
 1162 bilateral agreement.  
 1163  
 1164 **Qualifier name** this identifies the meaning and use of the Qualifier, using a name  
 1165 that is unambiguous within the scope of the Qualifier group.  
 1166  
 1167 **Must-be-understood** if this has the value “true” and the receiving entity does  
 1168 not recognise the Qualifier type (or does not implement the necessary  
 1169 functionality), a FAULT “UnsupportedQualifier” shall be returned and the  
 1170 message shall not be processed. Default is “true”.  
 1171  
 1172 **To-be-propagated** if this has the value “true” and the receiving entity passes the  
 1173 BTP message (which may be a CONTEXT, but can be other messages) onwards  
 1174 to other entities, the same Qualifier value shall be included. If the value is  
 1175 “false”, the Qualifier shall not be automatically included if the BTP message is  
 1176 passed onwards. (If the receiving entity does support the qualifier type, it is  
 1177 possible a propagated message may contain another instance of the same type,  
 1178 even with the same Content – this is not considered propagation of the original  
 1179 qualifier.). Default is “false”.  
 1180  
 1181 **Content** the type (which may be structured) and meaning of the content is  
 1182 defined by the specification of the Qualifier.  
 1183  
 1184

## 1185 CONTEXT

1186  
 1187 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more  
 1188 application messages. (The means by which this relationship is represented is determined by  
 1189 the binding and the binding mechanisms of the application protocol.) The “superior type”  
 1190 parameter identifies whether the Superior will apply the same decision to all Inferiors  
 1191 enrolled **with** using the same superior identifier (“superior type” is “atom”) or **whether it** may  
 1192 apply different decisions (“superior type” is “cohesion”).  
 1193

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

1194  
 1195  
 1196 **address-as-superior** the address to which ENROL and other messages from an  
 1197 enrolled Inferior are to be sent. This can be a set of alternative addresses.  
 1198  
 1199 **superior identifier** identifies the Superior within the scope of the address-as-  
 1200 superior

1201  
 1202           **superior type** identifies whether the CONTEXT refers to a Cohesion or an  
 1203           Atom. Default is atom.  
 1204  
 1205  
 1206           **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction  
 1207           timelimit” is carried by CONTEXT.  
 1208  
 1209           There is no target address parameter for CONTEXT as it is only transmitted in relation to the  
 1210           application messages.  
 1211  
 1212           The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the  
 1213           superior type with the appropriate value.  
 1214  
 1215

1216           **CONTEXT\_REPLY**

1217  
 1218           CONTEXT\_REPLY is sent after receipt of CONTEXT (related to application message(s)) to  
 1219           indicate whether all necessary enrolments have already completed (ENROLLED has been  
 1220           received) or will be completed by ENROL messages sent in relation to the  
 1221           CONTEXT\_REPLY or if an enrolment attempt has failed. CONTEXT\_REPLY may be sent  
 1222           related to an application message (typically the response to the application message related to  
 1223           the CONTEXT). In some bindings the CONTEXT\_REPLY may be implicit in the application  
 1224           message.  
 1225

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

1226  
 1227           **superior-address** one of the addresses from the address-as-superior from the  
 1228           CONTEXT. (The parameter is present in CONTEXT\_REPLY to disambiguate  
 1229           the superior identifier.)  
 1230

1231           **superior identifier** the superior identifier from the CONTEXT  
 1232

1233           **completion\_status:** reports whether all enrol operations made necessary by the  
 1234           receipt of the earlier CONTEXT message have completed. Values are  
 1235

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.

*repudiated*

At least one enrolment has failed. The implications of receiving the CONTEXT have **not** been honoured.

1236

1237

**qualifiers** standardised or other qualifiers.

1238

1239

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

1240

1241

1242

1243

1244

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

1245

1246

1247

1248

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

1249

1250

1251

## BEGIN

1252

1253

1254

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

1255

1256

1257

1258

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

1259

1260

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

1261

1262

1263

1264

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

1265

1266

1267

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

1268

1269

1270

**qualifiers** standardised or other qualifiers. The standard qualifier “Transaction timelimit” may be present on BEGIN, to set the timelimit for the new business

1271

1272 transaction and will be copied to the new CONTEXT. The standard qualifier  
1273 “Inferior name” may be present if there is a CONTEXT related to the BEGIN.  
1274

1275 A new top-level Business Transaction is created if there is no CONTEXT related to the  
1276 BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is  
1277 created if the CONTEXT message for the existing Business Transaction is related to the  
1278 BEGIN. In this case, the Factory is responsible for enrolling the new Composer or  
1279 Coordinator as an Inferior of the Superior identified in that CONTEXT.  
1280

---

1281 Note – This specification does not provide a standardised means to  
1282 determine which of the Inferiors of a sub-Composer are in its confirm set.  
1283 This is considered part of the application:inferior relationship.

---

1284 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction type” having  
1285 the corresponding value.  
1286

1287 Types of FAULT possible (sent to Reply address)  
1288

1289 **General**

1290 **BEGUN**

1291  
1292 BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT  
1293 for the new business transaction.  
1294  
1295  
1296

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

1297  
1298 **target address** the address to which the BEGUN is sent. This will be the reply  
1299 address from the BEGIN.  
1300

1301 **address-as-decider** for a top-level transaction (no CONTEXT related to the  
1302 BEGIN), this is the address to which PREPARE, REQUEST\_CONFIRM,  
1303 CANCEL and REQUEST\_STATUS messages are to be sent; if a CONTEXT  
1304 was related to the BEGIN this parameter is absent  
1305

1306 **transaction-identifier** identifies the new Composer or Coordinator within the  
1307 scope of the address-as-decider. If this is not a top-level transaction, the



1308 transaction-identifier is optional, but if present shall be the inferior-identifier used  
1309 in the enrolment with the Superior identified by the CONTEXT related to the  
1310 BEGIN.

1311  
1312 **inferior handle** Shall be absent if this is a top-level transaction and may or may  
1313 not be present otherwise. (Presence or absence will be determined by the nature  
1314 of the Superior identified in the CONTEXT related to the BEGIN). If present, the  
1315 inferior handle will identify this new business transaction as in the inferiors-list  
1316 parameters in messages between the Superior identified in the CONTEXT related  
1317 to the BEGIN (acting as a Decider) and its Terminator. The value shall be  
1318 different for each enrolled Inferior of that Superior.

1319  
1320 **address-as-inferior** This parameter shall be absent if this is a top-level  
1321 transaction and may be present, at implementation option otherwise. If present, it  
1322 shall be the address-as-inferior used in the enrolment with the Superior identified  
1323 by the CONTEXT related to the BEGIN. If this is a top-level transaction

1324  
1325 **qualifiers** standardised or other qualifiers.

1326  
1327 At implementation option, the “address-as-decider” and/or “address-as-inferior” and the  
1328 “address-as-superior” in the related CONTEXT may be the same or may be different. There  
1329 is no general requirement that they even use the same bindings. Any may also be the same as  
1330 the target address of the BEGIN message (the inferior identifier on messages will ensure they  
1331 are applied to the appropriate Composer or Coordinator).

1332  
1333 No FAULT messages are issued on receiving BEGUN.

## 1334 ENROL

1335  
1336 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a  
1337 CONTEXT message in relation to an application request.

1338 The actor issuing ENROL plays the role of Enroller.

1339  
1340

Parameter	type
target address	BTP address
superior identifier	Identifier
reply requested	Boolean
reply address	BTP address
address-as-inferior	Set of BTP address <sup>es</sup>
inferior identifier	Identifier
Qualifiers	List of qualifiers

1341  
1342 **target address** the address to which the ENROL is sent. This will be the  
1343 address-as-superior from the CONTEXT message.

1344  
 1345 **superior identifier**. The superior identifier as on the CONTEXT message  
 1346  
 1347 **reply requested** true if an ENROLLED response is required, false otherwise.  
 1348 Default is false.  
 1349  
 1350 **reply address** the address to which a replying ENROLLED is to be sent, if  
 1351 “reply requested” is true. If this field is absent and “reply requested” is true, the  
 1352 ENROLLED should be sent to the “address-as-inferior” (or one of them, at  
 1353 sender’s option)  
 1354  
 1355 **address-as-inferior** the address to which PREPARE, CONFIRM, CANCEL and  
 1356 SUPERIOR\_STATE messages for this Inferior are to be sent.  
 1357  
 1358 **inferior identifier** an identifier that unambiguously identifies this Inferior within  
 1359 the scope of any of the address-as-inferior set of BTP-addresses.  
 1360  
 1361 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior  
 1362 name” may be present.  
 1363

Types of FAULT possible (sent to Reply address)

**General**

**InvalidSuperior** – if superior identifier is unknown

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

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

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

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

**ENROLLED**

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

Parameter	Type
target address	BTP address
inferior identifier	Identifier

Parameter	Type
inferior-handle	Handle
Qualifiers	List of qualifiers

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

**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. ~~RESIGN may be sent in response to a PREPARE message (instead of a PREPARED), or at any point prior to the sending of a PREPARED or CANCELLED message.~~

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

1416

1417

1418

1419

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

1420 **superior-identifier** The superior identifier as on the ENROL message  
 1421  
 1422 **address-as-inferior** The address-as-inferior as on the earlier ENROL message  
 1423 (with the inferior identifier, this determines who the message is from)  
 1424  
 1425 **inferior-identifier** The inferior identifier as on the earlier ENROL message  
 1426  
 1427 **response-requested** is set to “true” if a RESIGNED response is required.  
 1428  
 1429 **qualifiers** standardised or other qualifiers.  
 1430

1431 Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued  
 1432 early.

1433  
 1434 Types of FAULT possible (sent to address-as-inferior)  
 1435

1436 *General*

1437 *InvalidSuperior* – if superior identifier is unknown

1438 *InvalidInferior* – if no ENROL had been received for this address-as-  
 1439 inferior and identifier (Inferior Identity)

1440 *WrongState* – if a PREPARED or CANCELLED has already been  
 1441 received by the Superior from this Inferior  
 1442

1443 The form RESIGN/rsp-req refers to an RESIGN message with “reply requested” having the  
 1444 value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “reply requested”  
 1445 having the value “false”  
 1446  
 1447

1448 **RESIGNED**

1449 Sent in reply to a RESIGN/rsp-req message.  
 1450  
 1451

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

1452  
 1453 **target address** the address to which the RESIGNED is sent. This will be the  
 1454 address-as-inferior from the ENROL message.  
 1455  
 1456 **inferior identifier** The inferior identifier as on the earlier ENROL message for  
 1457 this Inferior.  
 1458  
 1459 **qualifiers** standardised or other qualifiers.  
 1460

1461 After receiving this message the Inferior will not receive any more messages with this  
1462 address-as-inferior and identifier.

1463  
1464 No FAULT messages are issued on receiving RESIGNED.  
1465

## 1466 PREPARE

1467  
1468 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor  
1469 RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after  
1470 receiving a PREPARED message.

1471  
1472 Sent from a Terminator to a Composer to tell it to prepare all or some of its inferiors, by  
1473 sending PREPARE to any that have not already sent PREPARED, RESIGN or  
1474 CANCELLED to the Composer. If the inferiors-list parameter is absent, the request applies to  
1475 all the inferiors; if the parameter is present, it applies only to the identified inferiors of the  
1476 Composer.  
1477

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

1478  
1479 **target address** the address to which the PREPARE message is sent. When sent  
1480 from Superior to Inferior, this will be the address-as-inferior from the ENROL  
1481 message,. When sent from Terminator to Composer, this will be the decider-  
1482 address from the BEGUN message .

1483  
1484 **inferior identifier** When sent from Superior to Inferior, the inferior identifier as  
1485 on the earlier ENROL message. This parameter shall be absent when sent from  
1486 Terminator to Composer.  
1487

1488 **reply address** When sent from Terminator to Composer, the address of the  
1489 Terminator sending the PREPARE message. This parameter shall be absent when  
1490 sent from Superior to Inferior.

1491  
1492 **transaction identifier** When sent from Terminator to Composer, identifies the  
1493 Composer and will be the transaction-identifier from the BEGUN message.. This  
1494 parameter shall be absent when sent from Superior to Inferior.  
1495

1496  
1497 **inferiors-list** When sent from Terminator to Composer, defines which of the  
1498 Inferiors of this Composer preparation is requested for. If this parameter is absent

1499 when sent to a Composer, the PREPARE applies to all Inferiors. This parameter  
1500 shall be absent when sent from Superior to Inferior.

1501  
1502  
1503 **qualifiers** standardised or other qualifiers. The standard qualifier “Minimal  
1504 inferior timeout” is carried by PREPARE.

1505  
1506  
1507 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or  
1508 RESIGN.

1509  
1510 When sent to a Composer, for all Inferiors identified in the inferiors-list parameter (all  
1511 Inferiors if the parameter is absent), from which none of PREPARED, CANCELLED or  
1512 RESIGNED has been received, the Composer shall issue PREPARE. It will reply to the  
1513 Terminator, using the reply address on the PREPARE message, sending an  
1514 INFERIOR\_STATUSES message giving the status of the Inferiors identified on the inferiors-  
1515 list parameter (all of them if the parameter was absent).

1516  
1517 Types of FAULT possible (sent to Superior address)

1518  
1519 **General**  
1520 **UnknownTransaction** – if the transaction-identifier is unknown  
1521 **InvalidInferior** – if inferior identifier is unknown, or an inferior-handle  
1522 on the inferiors-list is unknown  
1523 **WrongState** – if a CONFIRM or CANCEL has already been received by  
1524 this Inferior; if a REQUEST\_CONFIRM or CANCEL/whole has already  
1525 been received by this Composer.

1526  
1527 The form PREPARE/whole refers to a PREPARE message sent to a Composer where the  
1528 “inferiors-list” parameter is absent. The form PREPARE/inferiors refers to a PREPARE  
1529 message sent to a Composer where the “inferiors-list” parameter is present. The unqualified  
1530 form PREPARE is used for a PREPARE message sent to an Inferior.

1531  
1532 **PREPARED**

1533  
1534 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when  
1535 the Inferior has determined the operations associated with the Inferior can be confirmed and  
1536 can be cancelled, as may be instructed by the Superior. The level of isolation is a local matter  
1537 (i.e. it is the Inferiors choice, as constrained by the shared understanding of the application  
1538 exchanges) – other access may be blocked, may see applied results of operations or may see  
1539 the original state.

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP address <u>es</u>

inferior identifier	Identifier
default is cancel	Boolean
qualifiers	List of qualifiers

1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583

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

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

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

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

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

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

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

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

**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

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

1584 CONFIRM

1585

1586 Sent by the Superior to an Inferior from whom PREPARED has been received.

1587

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

1588

1589

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

1590

1591

1592

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

1593

1594

1595

**qualifiers** standardised or other qualifiers.

1596

1597

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

1598

1599

1600

Types of FAULT possible (sent to Superior address)

1601

1602

**General**

1603

**InvalidInferior** – if inferior identifier is unknown

1604

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

1605

1606

1607

1608

1609 CONFIRMED

1610

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.

1611

1612

1613

1614

CONFIRMED is also sent by Decider to a Terminator in reply to REQUEST\_CONFIRM if all of the confirm-set confirms (and, for a Cohesion, all other Inferiors cancel) without reporting hazards.

1615

1616

1617

1618

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP addresses
inferior identifier	Identifier



Parameter	Type
address-as-decider	BTP address
transaction-identifier	identifier
confirm received	Boolean
qualifiers	List of qualifiers

1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658

**target address** the address to which the CONFIRMED is sent. When sent by an Inferior to a Superior, this will be the Superior address as on the CONTEXT message. When sent from a Decider to a Terminator it will be the reply address from the REQUEST\_CONFIRM 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. This parameter shall be absent when CONFIRMED is sent from Decider to Terminator.

**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). This parameter shall be absent when CONFIRMED is sent from Decider to Terminator.

**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. This parameter shall be absent when CONFIRMED is sent from Decider to Terminator.

**address-as-decider** When the message is sent from a Decider to the Terminator, this shall be the address-as-decider of the Decider as on the BEGUN message (with the transaction identifier, this determines who the message is from). This parameter shall be absent when CONFIRMED is sent from an Inferior to Superior.

**transaction identifier** When the message is sent from a Decider to the Terminator, this shall be the transaction identifier as on the BEGUN message (i.e. the identifier of the Decider as a whole). This parameter shall be absent when CONFIRMED is sent from an Inferior to Superior

**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). This parameter shall be absent when CONFIRMED is sent from Decider to Terminator.

**qualifiers** standardised or other qualifiers.

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

1659  
1660  
1661  
1662  
1663  
1664

**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.

1665  
1666  
1667  
1668

---

Note – A CONFIRMED message arriving before a CONFIRM message is sent, or after a CANCEL 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.)

---

1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676

The form CONFIRMED/auto refers to a CONFIRMED message with “confirm received” = “false”; CONFIRMED/response refers to a CONFIRMED message with “confirm received” = ”true”. The unqualified form CONFIRMED refers to the message without an confirm received parameter, as used between Decider and Terminator.

**CANCEL**

1677  
1678  
1679  
1680  
1681  
1682

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

Sent by a Terminator to a Decider at any time before REQUEST\_CONFIRM has been sent.

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

1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692

**target address** the address to which the CANCEL message is sent. When sent from Superior to Inferior, this will be the address-as-inferior from the ENROL message,. When sent from Terminator to Composer, this will be the decider-address from the BEGUN message .

**inferior identifier** When sent from Superior to Inferior, the inferior identifier as on the earlier ENROL message. This parameter shall be absent when sent from Terminator to Decider.

1693 **reply address** When sent from Terminator to Decider, the address of the  
1694 Terminator sending the CANCEL message. This parameter shall be absent when  
1695 sent from Superior to Inferior.

1696  
1697 **transaction identifier** When sent from Terminator to Decider, identifies the  
1698 Decider and will be the transaction-identifier from the BEGUN message.. This  
1699 parameter shall be absent when sent from Superior to Inferior.  
1700

1701 **inferiors-list** When sent from Terminator to Composer, defines which of the  
1702 Inferiors of this Composer are to be cancelled. This parameter shall be absent  
1703 when sent from a Superior to an Inferior and when sent from a Terminator to a  
1704 Coordinator.  
1705

1706 **qualifiers** standardised or other qualifiers.  
1707

1708 When sent to an Inferior, the effects of any operations associated with the Inferior should be  
1709 undone. If the Inferior had sent PREPARED, the Inferior is released from its promise to be  
1710 able to confirm the operations.  
1711

1712 When sent to a Decider with the inferiors-list parameter is absent, the business transaction is  
1713 cancelled – this is propagated to any remaining Inferiors by issuing CANCEL to them. No  
1714 more Inferiors will be permitted to enrol.  
1715

1716 When sent to a Composer, with the inferiors-list parameter present, only the Inferiors  
1717 identified in the inferiors-list are to be cancelled. Any other inferiors are unaffected by a  
1718 CANCEL/inferiors. Further Inferiors may be enrolled.  
1719

---

1720 Note – A CANCEL/inferiors issued to a Cohesion Composer identifying all  
1721 of its currently enrolled Inferiors will leave the Cohesion ‘empty’, but  
1722 permitted to continue with new Inferiors, if any enrol.

---

1723  
1724 Types of FAULT possible (sent to Superior address)  
1725

1726 **General**

1727 **UnknownTransaction** – if the transaction-identifier is unknown

1728 **InvalidInferior** – if inferior identifier is unknown, or an inferior-handle  
1729 on the inferiors-list is unknown

1730 **WrongState** – if a CONFIRM has been received by this Inferior; if a  
1731 REQUEST\_CONFIRM has been received by this Composer.  
1732

1733 The form CANCEL/whole refers to a CANCEL message sent to a Decider where the  
1734 “inferiors-list” parameter is absent. The form CANCEL/inferiors refers to a CANCEL  
1735 message sent to a Composer where the “inferiors-list” parameter is present. The unqualified  
1736 form CANCEL is used to refer to a CANCEL message sent from a Superior to an Inferior.  
1737  
1738

1739 **CANCELLED**

1740

1741 Sent when the Inferior has applied (or is applying) cancellation of the operations associated  
1742 with the Inferior. CANCELLED is sent from Inferior to Superior in the following cases:

1743

1744

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;

1745

1746

1747

2. in reply to CANCEL, regardless of whether PREPARED has been sent;

1748

1749

3. after sending PREPARED and then making and applying an autonomous decision to cancel.

1750

1751

1752

4. in reply to CONFIRM\_ONE\_PHASE if the ~~Inferior~~Inferior decides to cancel the associated operations

1753

1754

1755

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.

1756

1757

1758

CANCELLED is also sent by Decider to a Terminator in reply to REQUEST\_CONFIRM if all Inferiors cancel without reporting hazards.

1759

1760

**Parameter**

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

1761

1762

**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. When sent from a Decider to a Terminator it will be the reply address from the REQUEST\_CONFIRM message.

1763

1764

1765

1766

1767

**superior identifier** When the message is sent from an Inferior to the Superior, this shall be the superior identifier as on the CONTEXT message. This parameter shall be absent when CANCELLED is sent from Decider to Terminator.

1768

1769

1770

1771

**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). This parameter shall be absent when CANCELLED is sent from Decider to Terminator.

1772

1773

1774

1775  
1776 **inferior identifier** When the message is sent from an Inferior to the Superior, this  
1777 shall be the inferior identifier as on the earlier ENROL message. This parameter  
1778 shall be absent when CANCELLED is sent from Decider to Terminator.  
1779

1780 **address-as-decider** When the message is sent from a Decider to the  
1781 Terminator, this shall be the address-as-decider of the Decider as on the BEGUN  
1782 message (with the transaction identifier, this determines who the message is  
1783 from). This parameter shall be absent when CANCELLED is sent from an  
1784 Inferior to Superior.  
1785

1786 **transaction identifier** When the message is sent from a Decider to the  
1787 Terminator, this shall be the transaction identifier as on the BEGUN message (i.e.  
1788 the identifier of the Decider as a whole). This parameter shall be absent when  
1789 CANCELLED is sent from an Inferior to Superior  
1790

1791 **qualifiers** standardised or other qualifiers.  
1792

1793 Types of FAULT possible (sent to address-as-inferior)  
1794

1795 **General**

1796 **InvalidSuperior** – if Superior identifier is unknown

1797 **InvalidInferior** – if no ENROL has been received for this address-as-  
1798 inferior and identifier, or if RESIGN has been received from this Inferior

1799 **WrongState** – if CONFIRM has been sent  
1800

---

1801 Note – A CANCELLED message arriving before a CANCEL message is  
1802 sent, or after a CONFIRM has been sent will occur when the Inferior has  
1803 taken an autonomous decision and is not regarded as occurring in the wrong  
1804 state. (The latter will cause a CONTRADICTION message to be sent.)

---

1805  
1806

1807 **CONFIRM\_ONE\_PHASE**  
1808

1809 Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In  
1810 this case the two-phase exchange is not performed between the Superior and Inferior and the  
1811 outcome decision for the operations associated with the Inferior is determined by the Inferior.  
1812

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

1813  
 1814 **target address** the address to which the CONFIRM\_ONE\_PHASE message is  
 1815 sent This will be the address-as-inferior on the ENROL message.  
 1816  
 1817 **inferior identifier** The inferior identifier as on the earlier ENROL message for  
 1818 this Inferior.  
 1819  
 1820 **report hazard** Defines whether the superior wishes to be informed if a mixed  
 1821 condition occurs for the operations associated with the Inferior. If “report hazard”  
 1822 is “true”, the Inferior will reply with HAZARD if a mixed condition occurs, or if  
 1823 the Inferior cannot determine that a mixed condition has not occurred. If “report  
 1824 hazard” is false, the Inferior will report only its own decision, regardless of  
 1825 whether that decision was correctly and consistently applied. Default is false.  
 1826  
 1827 **qualifiers** standardised or other qualifiers.

1828  
 1829 CONFIRM\_ONE\_PHASE can be issued by a Superior to an Inferior from whom  
 1830 PREPARED has been received (subject to the requirement that there is only one enrolled  
 1831 Inferior).

1832  
 1833 Types of FAULT possible (sent to Superior address)

1834  
 1835 *General*

1836 *InvalidInferior* – if inferior identifier is unknown

1837 *WrongState* – if a PREPARE has already been received from this  
 1838 Inferior

1839  
 1840 **HAZARD**

1841  
 1842 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly  
 1843 and consistently cancel or confirm the operations in accord with the decision (either the  
 1844 received decision of the superior or its own autonomous decision), or when the Inferior is  
 1845 unable to determine that a “mixed” condition has not occurred.  
 1846

1847 HAZARD is also used to reply to a CONFIRM\_ONE\_PHASE if the Inferior determines there  
 1848 is a mixed condition within its associated operations or is unable to determine that there is not  
 1849 a mixed condition.  
 1850

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

1851  
 1852 **target address** the address to which the MIXED is sent. This will be the  
 1853 superior address from the ENROL message.  
 1854  
 1855 **superior identifier** The superior identifier as used on the ENROL message  
 1856  
 1857 **address-as-inferior** The address-as-inferior as on the earlier ENROL message  
 1858 (with the inferior identifier, this determines who the message is from)  
 1859  
 1860 **inferior identifier** The inferior identifier as on the earlier ENROL message  
 1861  
 1862 **level** indicates, with value “mixed” that a mixed condition has definitely  
 1863 occurred; or, with value “possible” that it is unable to determine whether a mixed  
 1864 condition has occurred or not.  
 1865  
 1866 **qualifiers** standardised or other qualifiers.

1867  
 1868 Types of FAULT possible (sent to address-as-inferior)  
 1869

1870 *General*

1871 *InvalidSuperior* – if Superior identifier is unknown

1872 *InvalidInferior* – if no ENROL has been received for this address-as-  
 1873 inferior and identifier, or if RESIGN has been received from this Inferior  
 1874

1875  
 1876 The form HAZARD/mixed refers to a HAZARD message with “level” = “mixed”, the form  
 1877 HAZARD/possible refers to a HAZARD message with “level” = “possible”.  
 1878

1879 **CONTRADICTION**  
 1880

1881 Sent by the Superior to an Inferior that has taken an autonomous decision contrary to the  
 1882 decision for the atom. This is detected by the Superior when the ‘wrong’ one of  
 1883 CONFIRMED or CANCELLED is received. CONTRADICTION is also sent in response to a  
 1884 HAZARD message.  
 1885

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

1886  
 1887 **target address** the address to which the CONTRADICTION message is sent.  
 1888 This will be the address-as-inferior from the ENROL message.  
 1889  
 1890 **inferior identifier** The inferior identifier as on the earlier ENROL message for  
 1891 this Inferior.  
 1892

1893 **qualifiers** standardised or other qualifiers.

1894

1895 Types of FAULT possible (sent to Superior address)

1896

1897 **General**

1898 **InvalidInferior** – if inferior identifier is unknown

1899 **WrongState** – if neither CONFIRMED or CANCELLED has been sent  
1900 by this Inferior

1901

1902 **SUPERIOR\_STATE**

1903

1904 Sent by a Superior as a query to an Inferior when

1905

1906 1. in the active state

1907

1908 2. there is uncertainty what state the Inferior has reached (due to recovery from  
1909 previous failure or other reason).

1910

1911 Also sent by the Superior to the Inferior in response to a received INFERIOR\_STATE, in  
1912 particular states.

1913

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

1914

1915 **target address** the address to which the SUPERIOR\_STATE message is sent.  
1916 This will be the address-as-inferior from the ENROL message.

1917

1918 **inferior identifier** The inferior identifier as on the earlier ENROL message for  
1919 this Inferior.

1920

1921 **status** states the current state of the Superior, in terms of its relation to this  
1922 Inferior only.

1923

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

1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961

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

**qualifiers** standardised or other qualifiers.

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

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

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

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

## INFERIOR\_STATE

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

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

Parameter	Type
target address	BTP address

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

1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976

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

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

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

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

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

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

1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988

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

**qualifiers** standardised or other qualifiers.

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

1989 A status of “unknown” shall only be sent if it has been determined for certain that the Inferior  
 1990 has no knowledge of a relationship with the Superior. If there could be persistent information  
 1991 corresponding to the Superior, but it is not accessible from the entity receiving an  
 1992 SUPERIOR\_STATE/\*/\*y (or other) message targeted on the Inferior or the entity cannot  
 1993 determine whether any such persistent information exists, the response shall be  
 1994 “inaccessible”.

1995  
 1996 INFERIOR\_STATE/unknown is also used as a response to messages, other than  
 1997 SUPERIOR\_STATE/\*/\*y that are received when the Inferior is not known (and it is known  
 1998 there is no state information for it).

1999  
 2000 A SUPERIOR\_STATE/INFERIOR\_STATE exchange that determines that one or both sides  
 2001 are in the active state does not require that the Inferior be cancelled (unlike some other two-  
 2002 phase commit protocols). The relationship between Superior and Inferior, and related  
 2003 application elements may be continued, with new application messages carrying the same  
 2004 CONTEXT. Similarly, if the Inferior is prepared but the Superior is active, there is no  
 2005 required impact on the progression of the relationship between them.

2006  
 2007 The form INFERIOR\_STATE/abcd refers to a INFERIOR\_STATE message status having a  
 2008 value equivalent to “abcd” (for active, unknown and inaccessible) and with “reply requested”  
 2009 = “false”. INFERIOR\_STATE/abcd/y refers to a similar message, but with “reply requested”  
 2010 = “true”. The form INFERIOR\_STATE/\*/\*y refers to a INFERIOR\_STATE message with  
 2011 “reply requested” = “true” and any value for status.

2012  
 2013 **REQUEST\_CONFIRM**

2014  
 2015 Sent from a Terminator to a Decider to request confirmation of the business transaction. If the  
 2016 business transaction is a Cohesion, the confirm-set is specified by the “inferiors-list”  
 2017 parameter.

2018

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

2019  
 2020 **target address** the address to which the REQUEST\_CONFIRM message is  
 2021 sent. This will be the address-as-decider on the BEGUN message.

2022  
 2023 **reply address** the address of the Terminator sending the REQUEST\_CONFIRM  
 2024 message.

2025

2026 **transaction identifier** identifies the Decider. This will be the transaction-  
2027 identifier from the BEGUN message.  
2028

2029 **inferiors-list** defines which Inferiors enrolled with the Decider, if it is a  
2030 Cohesion Composer, are to be confirmed. Shall be absent if the Decider is an  
2031 Atom Coordinator.  
2032

2033 **report hazard** Defines whether the Terminator wishes to be informed of hazard  
2034 events and contradictory decisions within the business transaction. If “report  
2035 hazard” is “true”, the receiver will wait until responses (CONFIRMED,  
2036 CANCELLED or HAZARD) have been received from all of its inferiors,  
2037 ensuring that any hazard events are reported. If “report hazard” is “false”, the  
2038 Decider will reply with CONFIRMED or CANCELLED as soon as the decision  
2039 for the transaction is known.  
2040

2041 **qualifiers** standardised or other qualifiers.  
2042

2043 If the “inferiors-list” parameter is present, the Inferiors identified shall be the “confirm-set” of  
2044 the Cohesion. If the parameter is absent and the business transaction is a Cohesion, the  
2045 “confirm-set” shall be all remaining Inferiors. If the business transaction is an Atom, the  
2046 “confirm-set” is automatically all the Inferiors.  
2047

2048 Any Inferiors from which RESIGN is received are not counted in the confirm-set.  
2049

2050 If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED  
2051 has not been received, PREPARE shall be issued to that Inferior.  
2052

---

2053 NOTE -- If PREPARE has been sent but PREPARED not yet received from  
2054 an Inferior in the confirm-set, it is an implementation option whether and  
2055 when to re-send PREPARE. The Superior implementation may choose to re-  
2056 send PREPARE if there are indications that the earlier PREPARE was not  
2057 delivered.

---

2058 ~~If PREPARED has not been received from any Inferiors in the confirm-set, PREPARE shall~~  
2059 ~~be issued to them.~~  
2060

2061 A confirm decision may be made only if PREPARED has been received from all Inferiors in  
2062 the “confirm-set”. The making of the decision shall be persistent (and if it is not possible to  
2063 persist the decision, it is not made). If there is only one remaining Inferior in the “confirm  
2064 set” and PREPARE has not been sent to it, CONFIRM\_ONE\_PHASE may be sent to it.  
2065

2066 All remaining Inferiors that are not in the confirm set shall be cancelled.  
2067

2068 If a confirm decision is made and “report-hazard” was “false”, a CONFIRMED message shall  
2069 be sent to the “reply-address”.  
2070

2071 If a cancel decision is made and “report-hazard” was “false”, a CANCELLED message shall  
2072 be sent to the “reply-address”.

2073  
2074 If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.  
2075 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in  
2076 the confirm-set), an INFERIOR\_STATUSES reporting the status for all Inferiors shall be sent  
2077 to the “reply-address”.

2078  
2079 Types of FAULT possible (sent to reply address)

2080  
2081 *General*  
2082 *UnknownTransaction* – if the transaction-identifier is unknown  
2083 *InvalidInferior* – if an inferior handle in the inferiors-list is unknown  
2084 *WrongState* – if a CANCEL/whole has already been received .

2085  
2086 The form REQUEST\_CONFIRM/whole refers to a REQUEST\_CONFIRM message where  
2087 the “inferiors-list” parameter is absent. The form REQUEST\_CONFIRM /inferiors refers to  
2088 a REQUEST\_CONFIRM message where the “inferiors-list” parameter is present.

## 2089 2090 REQUEST\_STATUSES

2091  
2092 Sent to a Decider to ask it to report the status of its Inferiors with an  
2093 INFERIOR\_STATUSES message.  
2094

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

2095  
2096 **target address** the address to which the REQUEST\_STATUS message is  
2097 sent..This will be the address-as-decider from the BEGUN message.

2098  
2099 **reply address** the address to which the replying INFERIOR\_STATUSES is to  
2100 be sent

2101  
2102 **transaction identifier** identifies the Decider. This will be the transaction-  
2103 identifier from the BEGUN message.

2104  
2105 **inferiors-list** defines which inferiors enrolled with the Composer or Coordinator  
2106 are to be included in the INFERIOR\_STATUSES. If the list is absent, the status  
2107 of all enrolled inferiors will be reported.

2108  
2109 **qualifiers** standardised or other qualifiers.

2110  
2111 Types of FAULT possible (sent to reply-address)  
2112

2113 *General*  
2114

2115 The form REQUEST\_STATUSES/whole refers to a REQUEST\_STATUS with the inferiors-  
2116 list absent. The form REQUEST\_STATUS/inferiors refers to a REQUEST\_STATUS with  
2117 the inferiors-list present.  
2118

2119 **INFERIOR\_STATUSES**  
2120

2121 Sent by a Decider to report the status of all or some of its inferiors in response to a  
2122 REQUEST\_STATUSES, PREPARE, CANCEL/inferiors and REQUEST\_CONFIRM with  
2123 “report-hazard” = “true”.  
2124

Parameter	Type
target address	BTP address
address-as-decider	BTP address
transaction-identifier	identifier
status-list	Set of Status items - see below
general-qualifiers	List of qualifiers

2125  
2126 **target address** the address to which the INFERIOR\_STATUSES is sent. This  
2127 will be the reply address on the received message  
2128

2129 **address-as-decider** The address-as-decider of the Decider as on the BEGUN  
2130 message (with the transaction identifier, this determines who the message is  
2131 from)  
2132

2133 **transaction identifier** The transaction identifier as on the BEGUN message (i.e.  
2134 the identifier of the Decider as a whole)  
2135

2136 **status-list** contains a number of Status-items, each reporting the status of one of  
2137 the inferiors of the Decider. The fields of a Status-item are  
2138

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

2139  
2140  
2141  
2142

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

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

2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156

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

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

#### REQUEST\_STATUS

Sent to an Inferior or to a Decider to ask it to reply with STATUS.

2157

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

2158

2159

2160

2161

2162

2163

**target address** the address to which the REQUEST\_STATUS message is sent. If sent. If the target is an Inferior, this will be the address-as-inferior on the ENROL message. If the target is a Decider, this will be the address-as-decider on the BEGUN message.

2164

2165

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

2166

2167

2168

**inferior identifier** If the target is an Inferior, the “inferior-identifier” on the ENROL message. If the target is a Decider, this parameter shall be absent.

2169

2170

**transaction-identifier** If the target is a Decider, the “transaction-identifier” on the BEGUN message. If the target is an Inferior, this parameter shall be absent.

2171

2172

**qualifiers** standardised or other qualifiers.

2173

Types of FAULT possible (sent to reply address)

2174

2175

### *General*

2176

2177

2178

## STATUS

2179

2180

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

2181

2182

2183

Parameter	Type
target address	BTP address
address-as-inferior	BTP address
inferior identifier	Identifier
address-as-decider	BTP address
transaction-identifier	Identifier
status	See below
qualifiers	List of qualifiers



2184  
 2185  
 2186  
 2187  
 2188  
 2189  
 2190  
 2191  
 2192  
 2193  
 2194  
 2195  
 2196  
 2197  
 2198  
 2199  
 2200  
 2201  
 2202  
 2203  
 2204

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

**address-as-inferior** If the sender is an Inferior, the address-as-inferior as on the ENROL message (with the inferior-identifier, this determines who the message is from). If the sender is a Decider, this parameter shall be absent

**inferior-identifier** If the sender is an Inferior, the inferior-identifier as on the ENROL message. If the sender is a Decider, this parameter shall be absent.

**address-as-decider** If the sender is a Decider, the address-as-decider on the BEGUN message (with the “transaction-identifier”, this determines who the message is from). If the sender is an Inferior, this parameter shall be absent.

**transaction-identifier** If the sender is a Decider, the transaction identifier as on the BEGUN message. If the sender is an Inferior, this parameter shall be absent.

**status** states the current status of the transaction tree node represented by the sender.

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

status value	Meaning from Inferior	Meaning from Decider
<i>Cancelling</i>	CANCEL has been received or auto-cancel has been decided	Cancel decision has been made but responses from inferiors are pending
<i>Cancelled</i>	CANCELLED has been sent	CANCELLED has been sent
<i>cancel-contradiction</i>	Autonomous cancel decision was made, CONFIRM received; CONTRADICTION has not been received	Not applicable
<i>confirm-contradiction</i>	Autonomous confirm decision was made, CANCEL received; CONTRADICTION has not been received	Not applicable
<i>Hazard</i>	A hazard has been discovered; CONTRADICTION has not been received	A hazard has been reported from at least one Inferior
<i>Contradicted</i>	CONTRADICTION has been received	Not applicable
<i>Unknown</i>	No state information for the identifier exists; no such Inferior exists	No state information for the transaction identifier exists; no such Decider exists
<i>Inaccessible</i>	There may be state information for this identifier but it cannot be reached/existence cannot be determined	There may be state information for this identifier but it cannot be reached/existence cannot be determined

2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213

**qualifiers** standardised or other qualifiers.

## 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

2214  
 2215 **target address** the address to which the REDIRECT is sent. This may be the  
 2216 reply address from a received message or the address of the opposite side  
 2217 (superior/inferior) as given in a CONTEXT or ENROL message  
 2218  
 2219 **superior identifier** The superior identifier as on the CONTEXT message and  
 2220 used on an ENROL message. (present only if the REDIRECT is sent from the  
 2221 Inferior).  
 2222  
 2223 **inferior identifier** The inferior identifier as on the ENROL message  
 2224  
 2225 **old address** The previous address of the sender of REDIRECT. A match is  
 2226 considered to apply if any of the old addresses match one that is already known.  
 2227  
 2228 **new address** The (set of alternatives) new addresses to be used for messages  
 2229 sent to this entity.  
 2230  
 2231 **qualifiers** standardised or other qualifiers.  
 2232  
 2233 If the actor whose address is changed is an Inferior, the new address value  
 2234 replaces the address-as-inferior as present in the ENROL.  
 2235  
 2236 If the actor whose address is changed is a Superior, the new address value  
 2237 replaces the Superior address as present in the CONTEXT message (or as present  
 2238 in any other mechanism used to establish the Superior:Inferior relationship).  
 2239  
 2240

## 2241 FAULT

2242  
 2243 Sent in reply to various messages to report an error condition  
 2244

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

2245  
 2246 **target address** the address to which the FAULT is sent. This may be the reply  
 2247 address from a received message or the address of the opposite side  
 2248 (superior/inferior) as given in a CONTEXT or ENROL message  
 2249

2250 **superior identifier** the superior identifier as on the CONTEXT message and as  
2251 used on the ENROL message (present only if the FAULT is sent to the superior).

2252  
2253 **inferior identifier** the inferior identifier as on the ENROL message (present only  
2254 if the FAULT is sent to the inferior)

2255  
2256 **fault type** identifies the nature of the error, as specified for each of the main  
2257 messages.

2258  
2259 **fault data** information relevant to the particular error. Each fault type defines the  
2260 content of the fault data:

2261

<b>fault type</b>	<b>meaning</b>	<b>fault data</b>
<i>General</i>	Any otherwise unspecified problem	Free text explanation
<i>UnknownParameter</i>	A BTP message has been received with an unrecognised parameter	Free text explanation
<i>WrongState</i>	The message has arrived when the recipient is in an invalid state.	
<i>CommunicationFailure</i>	Any fault arising from the carrier mechanism and communication infrastructure.	Determined by the carrier mechanism and binding specification
<i>InvalidSuperior</i>	The received identifier is not known or does not identify a known Superior	The identifier
<i>DuplicateInferior</i>	An inferior with the same address and identifier is already enrolled with this Superior	The identifier
<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>UnsupportedQualifier</i>	A qualifier has been received that is not recognised and on which "must-be-Understood" is "true".	Qualifier group and name

2262

2263

2264

**qualifiers** standardised or other qualifiers.

2265  
2266  
2267

---

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

---

2268

2269  
2270

### Standard qualifiers

2271  
2272  
2273  
2274

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

2275

2276  
2277

### Transaction timelimit

2278  
2279  
2280  
2281  
2282  
2283

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

2284  
2285  
2286  
2287  
2288

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

2289  
2290

The qualifier is propagated on a CONTEXT message.

2291  
2292

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

2293  
2294

The “Content” shall contain the following field:

Content field	Type
Timelimit	Integer

2295

2296  
2297  
2298  
2299

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

2300  
2301

### Inferior timeout

2302  
2303  
2304  
2305  
2306  
2307

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

2308 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply  
 2309 a confirm or cancel decision before the CONFIRM or CANCEL is received and before this  
 2310 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,  
 2311 and (as with other transaction mechanisms), is considered to be an exceptional event. As with  
 2312 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the  
 2313 expiry of this timeout, is liable to cause contradictory decisions across the business  
 2314 transaction. BTP ensures that at least the occurrence of such a contradiction will be  
 2315 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic  
 2316 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this  
 2317 timeout does not cause a qualitative (state table) change in what can happen, but rather a step  
 2318 change in the probability that it will.  
 2319

2320 The expiry of the timeout does not strictly require that the Inferior immediately invokes the  
 2321 intended decision, only that is at liberty to do so. An implementation may choose to only  
 2322 apply the decision if there is contention for the underlying resource, for example.  
 2323 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for  
 2324 the business transaction are made before these timeouts expire (and allow a margin of error  
 2325 for network latency etc.).  
 2326

2327 The qualifier may be present on a PREPARED message. If the PREPARED message has the  
 2328 “default is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall  
 2329 have the value “cancel”.  
 2330

2331 The “Qualifier name” shall be “inferior-timeout” .  
 2332

2333 The “Content” shall contain the following fields:  
 2334

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

2335  
 2336 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the  
 2337 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the  
 2338 effects of the associated operations, as ordered by the receiving Superior.  
 2339

2340 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an  
 2341 autonomous decision is made.  
 2342

### 2343 **Minimum inferior timeout**

2344  
 2345 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the  
 2346 Inferior. If a Superior knows that the decision for the business transaction will not be  
 2347 determined for some period, it can require that Inferiors do not send PREPARED messages  
 2348 with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to  
 2349 send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with  
 2350 CANCELLED.  
 2351

2352 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If  
2353 present on more than one, and with different values of the MinimumTimeout field, the value  
2354 on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall  
2355 prevail over either of the others.  
2356

2357 The “Qualifier name” shall be “minimum-inferior-timeout” .

2358  
2359 The “Content” shall contain the following field:  
2360

Content field	Type
MinimumTimeout	Integer

2361  
2362 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be  
2363 acceptable in the Inferior timeout qualifier on an answering PREPARED message. |  
2364

### 2365 Inferior name

2366  
2367 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on  
2368 INFERIOR\_STATUSES and thus allow the Terminator to determine which Inferior (of the  
2369 Composer or Coordinator) is related to which application work. This is in addition to the  
2370 “inferior handle” field. The name can be human-readable and can also be used in fault  
2371 tracing, debugging and auditing.  
2372

2373 The name is never used by the BTP actors themselves to identify each other or to direct  
2374 messages. (The BTP actors use the addresses and the identifiers in the message parameters  
2375 for those purposes.)  
2376

2377 This specification makes no requirement that the names are unambiguous within any scope  
2378 (unlike the “inferior-handle” on ENROLLED and BEGUN, which is required to be  
2379 unambiguous within the scope of the Decider). Other specifications, including those defining  
2380 use of BTP with a particular application may place requirements on the use and form of the  
2381 names. (This may include reference to information passed in application messages or in other,  
2382 non-standardised, qualifiers.)  
2383

2384 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item  
2385 in INFERIOR\_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if  
2386 present, the same qualifier value **should** be included in the consequent ENROL. If  
2387 INFERIOR\_STATUSES includes a Status-item for an Inferior whose ENROL had an  
2388 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.  
2389

2390 The “Qualifier -name” shall be “inferior-name”

2391  
2392 The “Content” shall contain the following fields:  
2393

Content field	Type
inferior-name	String

2394  
2395  
2396

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



2397

## State Tables

2398

### Explanation of the state tables

2399

2400

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).

2401

2402

2403

2404

2405

2406

2407

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.

2408

2409

2410

2411

2412

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.

2413

2414

2415

### Status queries

2416

2417

2418

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.

2419

2420

2421

2422

2423

2424

2425

2426

2427

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.

2428

2429

2430

2431

2432

2433

2434

2435

2436

### Decision events

2437

2438

2439

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 business transaction and on features of the implementation (e.g. making of a persistent record

2440

2441

2442

2443

2444 of the decision means that the information will survive at least some failures that otherwise  
2445 lose state information, but the level of survival depends on the purpose of the  
2446 implementation). Table 2 and Table 3 define the decision events.  
2447

2448 In some cases, an implementation may not need to make an active change to have a persistent  
2449 record of a decision, provided that the implementation will restore itself to the appropriate  
2450 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,  
2451 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via  
2452 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a  
2453 record of “decide to cancel autonomously”, provided it always updated such a record as part  
2454 of the “apply ordered confirmation” decision event.  
2455

2456 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of  
2457 PREPARE indicates that the application exchange (to associate operations with the Inferior)  
2458 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier  
2459 state corresponding to an incomplete application exchange. However, implementations are  
2460 not required to make the sending of PREPARE persistent in terms of recovery – a Superior  
2461 that experiences failure after sending PREPARE may, on recovery, have no information  
2462 about the transaction, in which case it is considered to be in the completed state (Z), which  
2463 will imply the cancellation of the Inferior and its associated operations.  
2464

2465 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its  
2466 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a  
2467 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local  
2468 persistent information (which would combine both superior and inferior information, pointing  
2469 both up and down the tree).  
2470

2471

### 2472 **Disruptions – failure events**

2473

2474 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or  
2475 may) cause a change of state. The disruption events in the state tables model different extents  
2476 of loss of state information. An implementation is not required to exhibit all the possible  
2477 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a  
2478 possible disruption.  
2479

2480 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,  
2481 which involves possible interruption of service and loss of messages in transit, but no change  
2482 of state (either because no state information was lost, or because recovery from persistent  
2483 information restores the implementation to the same state). The “disruption 0” event would  
2484 typically be an appropriate abstraction for a communication failure.  
2485

### 2486 **Invalid cells and assumptions of the communication mechanism**

2487

2488 The empty cells in state table represent events that cannot happen. For events corresponding  
2489 to sending a message or any of the decision events, this prohibition is absolute – e.g. a  
2490 conformant implementation in the Superior active state “B1” will not send CONFIRM. For

2491 events corresponding to receiving a message, the interpretation depends on the properties of  
2492 the underlying communications mechanism.

2493  
2494 For all communication mechanisms, it is assumed that

- 2495 a) the two directions of the Superior:Inferior communication are not synchronised –
- 2496 that is messages travelling in opposite directions can cross each other to any
- 2497 degree; any number of messages may be in transit in either direction; and
- 2498 b) messages may be lost arbitrarily

2499  
2500 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered  
2501 at all, are delivered to the receiver in the order they were sent), then receipt of a message in a  
2502 state where the corresponding cell is empty indicates that the far-side has sent a message out  
2503 of order – a FAULT message with the Fault Type “WrongState” can be returned.

2504  
2505 If the communication mechanisms cannot guarantee ordered delivery, then messages received  
2506 where the corresponding cell is empty should be ignored. Assuming the far-side is  
2507 conformant, these messages can assumed to be “stale” and have been overtaken by messages  
2508 sent later but already delivered. (If the far-side is non-conformant, there is a problem  
2509 anyway).

2510

## 2511 **Meaning of state table events**

2512

2513 The tables in this section define the events (rows) in the state tables. Table 1 defines the  
2514 events corresponding to sending or receiving BTP messages and the disruption events. Table  
2515 2 describes the decision events for an Inferior, Table 3 those for a Superior.

2516

2517 The decision events for a Superior, defined in Table 3 cannot be specified without reference  
2518 to other Inferiors to which it is Superior and to its relation with the application or other entity  
2519 that (acting ultimately on behalf of the application) drives it.

2520

2521 The term “remaining Inferiors” ~~are~~ **refers to** any actors to which this endpoint is Superior and  
2522 which are to be treated as an atomic decision unit with (and thus including) the Inferior on  
2523 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior  
2524 type” of “atom”, this will be all Inferiors established with same Superior address and Superior  
2525 identifier except those from which RESIGN has been received. If the CONTEXT had  
2526 “superior type” of “cohesion”, the “remaining Inferiors” excludes any that it has been  
2527 determined will be cancelled, as well as any that have resigned – in other words it includes  
2528 only those for which a confirm decision is still possible or has been made. The determination  
2529 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some  
2530 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this  
2531 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.

2532

2533 In order to ensure that the confirmation decision **is** delivered to all remaining Inferiors,  
2534 despite failures, the Superior must persistently record which these Inferiors are (i.e. their  
2535 addresses and identifiers). It must also either record that the decision is confirm, or ensure  
2536 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it  
2537 will be told about it. This latter would apply if the Superior were also BTP Inferior to another  
2538 entity which persisted a confirm decision (or recursively deferred it still higher). However,

2539 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the  
 2540 behaviour of asking another entity to make (and persist) the confirm decision is termed  
 2541 "offering confirmation" - the Superior offers the possible confirmation of itself, and its  
 2542 remaining Inferiors to some other entity. If that entity (or something higher up) then does  
 2543 make and persist a confirm decision, the Superior is "instructed to confirm" (which is  
 2544 equivalent BTP CONFIRM).

2545 The application, or an entity acting indirectly on behalf of the application, may request a  
 2546 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no  
 2547 more operations associated with the Inferior. Following a request to prepare all remaining  
 2548 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the  
 2549 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the  
 2550 application.)  
 2551

2552 The application, or an entity acting indirectly on behalf of the application, may also request  
 2553 confirmation. This means the Superior is to attempt to make and persist a confirm decision  
 2554 itself, rather than offer confirmation.  
 2555  
 2556  
 2557

2558 **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
send/receive SUP_STATE/***/y	send/receive SUPERIOR_STATE with status *** and reply-requested = true ("prepared-rcvd" represents "prepared-received")

Event name	Meaning
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

2559

2560

**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>
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>

Event name	Meaning
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>

2561

2562

**Table 3: Decision events for a Superior**

Event name	Meaning
decide to request confirm	<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>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> <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>

2563

2564

2565

**Persistent information**

2566 Persisted information (especially prepared information at an Inferior, confirm information at a  
2567 Superior) may include qualifications of the state carried in Qualifiers of the corresponding  
2568 message (e.g. inferior timeouts in prepared information). It may also include application-  
2569 specific information (especially in Inferiors) to allow the future confirmation or cancellation  
2570 of the associated operations. In some cases it will also include information allowing an  
2571 application message sent with a BTP message (e.g. PREPARED) to be repeated.

2572  
2573 The “effective” removal of persistent information allows for the possibility that the  
2574 information is retained (perhaps for audit and tracing purposes) but some change to the  
2575 persistent information (as a whole) means that if there is a failure after such change, on  
2576 recovery, the persistent information does not cause the endpoint to return the state it would  
2577 have recovered to before the change.

2578  
2579 In all cases, the degree to which information described as “persistent” will survive failure is a  
2580 configuration and implementation option. An implementation **should** describe the level of  
2581 failure that it is capable of surviving. For applications manipulating information that is itself  
2582 volatile (e.g. network configurations), there is no requirement to make the BTP state  
2583 information more persistent than the application information.

2584  
2585 The degree of persistence of the recording of a hazard (problem) at an Inferior and recording  
2586 of a detected contradiction at a Superior may be different from that applying to the persistent  
2587 prepared and confirm information. Implementations and configuration may choose to pass  
2588 hazard and contradiction information via management mechanisms rather than through BTP.  
2589 Such passing of information to a management mechanism could be treated as “record  
2590 problem” or “record contradiction”.

2591

**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	REQUEST 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	REQUEST CONFIRM received after prepared state
s2	REQUEST CONFIRM received
s3	REQUEST CONFIRM received, confirming
s4	REQUEST CONFIRM received, cancelling
s5	REQUEST CONFIRM received, hazard detected
s6	REQUEST CONFIRM received, hazard recorded
x1	completed, presuming abort
x2	completed, presuming abort after prepared/cancel
y1	completed, queried

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

2595  
2596

Table 6: Superior state table – normal forward progression

	I 1	A1	B1	C1	D1	E1	E2	F1	F2
receive ENROL/rsp-req	A1								
receive ENROL/no-rsp-req	B1								
receive RESIGN/rsp-req	Y1		C1	C1	C1				
receive RESIGN/no-rsp-req	Z		Z	Z	Z				
receive PREPARED	Y1		E1		E1	E1		F1	
receive PREPARED/cancel	Y1		E2		E2		E2	F1	
receive CONFIRMED/auto	Q1		H1		H1	H1		F1	
receive CONFIRMED/response								F2	F2
receive CANCELLED	Y1		Z		Z	J1	J1	K1	
receive HAZARD	P1	P1	P1		P1	P1	P1	P3	
receive INF_STATE/active/y	Y1	A1	B1		D1				
receive INF_STATE/active			B1		D1				
receive INF_STATE/unknown			Z	Z	Z				
send ENROLLED		B1							
send RESIGNED				Z					
send PREPARE					D1	E1	E2		
send CONFIRM_ONE_PHASE								F1	
send CONFIRM									
send CANCEL									
send CONTRADICTION									
send SUP_STATE/active/y			B1						
send SUP_STATE/active			B1						
send SUP_STATE/prepared-rcvd/y						E1	E2		
send SUP_STATE/prepared-rcvd						E1	E2		
send SUP_STATE/unknown									
decide to request confirm			S1			S1	S1		
decide to prepare			D1						
decide to confirm						F1	F1		
decide to cancel			G1		G1	G1	Z		
remove persistent information									Z
record contradiction									
disruption I	Z	Z	Z	Z	Z	Z	Z		F1
disruption II						D1	D1		
disruption III						B1	B1		
disruption IV									

Table 7: Superior state table – cancellation and contradiction

	G1	G2	G3	G4	H1	J1	K1	L1
receive ENROL/rsp-req								
receive ENROL/no-rsp-req								
receive RESIGN/rsp-req	G3	Z	G3					
receive RESIGN/no-rsp-req	Z	Z	Z					
receive PREPARED	G1	G2						
receive PREPARED/cancel	G1	G2						
receive CONFIRMED/auto	L1	L1			H1			L1
receive CONFIRMED/response								
receive CANCELLED	G4	Z		G4		J1	K1	
receive HAZARD	P4	P4						
receive INF_STATE/active/y	G1	G2						
receive INF_STATE/active	G1	G2						
receive INF_STATE/unknown	Z	Z	Z	Z				
send ENROLLED								
send RESIGNED								
send PREPARE								
send CONFIRM_ONE_PHASE								
send CONFIRM								
send CANCEL	G2	G2	Z	Z				
send CONTRADICTION								
send SUP_STATE/active/y								
send SUP_STATE/active								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
decide to request confirm								
decide to prepare								
decide to confirm					F1	K1		
decide to cancel					L1	G4		
remove persistent information								
record contradiction							R1	R1
disruption I	Z	Z	Z	Z	Z	Z	F1	Z
disruption II			G2	G2	E1	E1		G2
disruption III					D1	D1		
disruption IV					B1	B1		

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

	P1	P2	P3	P4	Q1	R1	R2	S1
receive ENROL/rsp-req								
receive ENROL/no-rsp-req								
receive RESIGN/rsp-req								C1
receive RESIGN/no-rsp-req								Z
receive PREPARED								S1
receive PREPARED/cancel								S1
receive CONFIRMED/auto					Q1	R1	R1	S1
receive CONFIRMED/response					Z	R2		Z
receive CANCELLED						R1	R1	Z
receive HAZARD	P1	P2	P3	P4		R1	R1	Z
receive INF_STATE/active/y								S1
receive INF_STATE/active								S1
receive INF_STATE/unknown	P1	P2		P4		R2	R2	Z
send ENROLLED								
send RESIGNED								
send PREPARE								
send CONFIRM_ONE_PHASE								S1
send CONFIRM								
send CANCEL								
send CONTRADICTION						R2		
send SUP_STATE/active/y								
send SUP_STATE/active								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
decide to request confirm								
decide to prepare								
decide to confirm								
decide to cancel								
remove persistent information							Z	
record contradiction	R1	R1	R1	R1	R1			
disruption I	Z	Z	Z	Z	Z		R1	Z
disruption II	D1		F1	G2				
disruption III	B1							
disruption IV								

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

	Y1	Z
receive ENROL/rsp-req		Y1
receive ENROL/no-rsp-req		Y1
receive RESIGN/rsp-req	Y1	Y1
receive RESIGN/no-rsp-req	Z	Z
receive PREPARED	Y1	Y1
receive PREPARED/cancel	Y1	Y1
receive CONFIRMED/auto	Q1	Q1
receive CONFIRMED/response	Z	Z
receive CANCELLED	Y1	Y1
receive HAZARD	P2	P2
receive INF_STATE/active/y	Y1	Y1
receive INF_STATE/active	Y1	Z
receive INF_STATE/unknown	Z	Z
send ENROLLED		
send RESIGNED		
send PREPARE		
send CONFIRM_ONE_PHASE		
send CONFIRM		
send CANCEL		
send CONTRADICTION		
send SUP_STATE/active/y		
send SUP_STATE/active		
send SUP_STATE/prepared-rcvd/y		
send SUP_STATE/prepared-rcvd		
send SUP_STATE/unknown	Z	
decide to request confirm		
decide to prepare		
decide to confirm		
decide to cancel		
remove persistent information		
record contradiction		
disruption I	Z	
disruption II		
disruption III		
disruption IV		

2600  
2601

2601

2602

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

	i 1	a1	b1	c1	d1	e1	e2	f1	f2
send ENROL/rsp-req	a1								
send ENROL/no-rsp-req	b1								
send RESIGN/rsp-req				c1					
send RESIGN/no-rsp-req				z					
send PREPARED						e1			
send PREPARED/cancel							e2		
send CONFIRMED/auto									
send CONFIRMED/response									
send CANCELLED			z		z				
send HAZARD									
send INF_STATE/active/y		a1	b1		d1				
send INF_STATE/active			b1		d1				
send INF_STATE/unknown									
receive ENROLLED		b1							
receive RESIGNED				z					
receive PREPARE		d1	d1	c1	d1	e1	e2		
receive CONFIRM_ONE_PHASE		s2	s2	c1		s1	s1		
receive CONFIRM						f1	f2	f1	f2
receive CANCEL		n1	n1	z	n1	g1	g2		
receive CONTRADICTION									
receive SUP_STATE/active/y		b1	b1	c1		e1	e2		
receive SUP_STATE/active		b1	b1	c1		e1	e2		
receive SUP_STATE/prepared-rcvd/y						e1	e2		
receive SUP_STATE/prepared-rcvd						e1	e2		
receive SUP_STATE/unknown		z	z	z	z	x1	x2		
decide to resign				c1	c1				
decide to be prepared				e1	e1				
decide to be prepared/cancel				e2	e2				
decide to confirm autonomously						h1			
decide to cancel autonomously						j1	z1		
apply ordered confirmation								m1	m1
remove persistent information									
detect problem		p1	p1		p1	p2	p2	p2	p2
detect and record problem									
disruption I		z	z	z	z			e1	e2
disruption II					b1				
disruption III									

2603

2604

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

	g1	g2	h1	h2	j1	j2	k1	k2	l1	l2
send ENROL/rsp-req send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD			h1		j1		k1		l1	
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown										
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown			h1 s3 h2	h2	j1 s4 k1 j2	j2	k1 k2	k2	l1 l2	l2
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				m1		z		z		z
disruption I disruption II disruption III	e1	e2	h1		j1		j1 k1 j1		h1	l1 h1

2605  
2606



**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	z				
send CANCELLED		z			
send HAZARD			p1	p2	q1
send INF_STATE/active/y					
send INF_STATE/active					
send INF_STATE/unknown					
receive ENROLLED			p1		q1
receive RESIGNED					
receive PREPARE			p1	p2	q1
receive CONFIRM_ONE_PHASE			s5	s5	s6
receive CONFIRM	m1			p2	q1
receive CANCEL		n1	p1	p2	q1
receive CONTRADICTION			z	z	z
receive SUP_STATE/active/y			p1	p2	q1
receive SUP_STATE/active			p1	p2	q1
receive SUP_STATE/prepared-rcvd/y				p2	q1
receive SUP_STATE/prepared-rcvd				p2	q1
receive SUP_STATE/unknown		z	p1	p2	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	z	z	z		
disruption II		d1			
disruption III		b1			

Table 13: Inferior state table– request confirm states

	s1	s2	s3	s4	s5	s6
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						
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 receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown	s1	s2	s3	s4	s5	s6
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			s3			s6
disruption I disruption II disruption III	e1	z		z	z	

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

	x1	x2	y1	y2	z	z1
send ENROL/rsp-req send ENROL/no-rsp-req send 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 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 z y1 z1 y1 y1 m1 y2 y1 y1 z z y1 y2 z z1 y2 y2 y2 y2 z 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						z z
disruption I disruption II disruption III	e1	e2				

2610

2611

2612

## 2612 **Failure Recovery**

### 2613 **Types of failure**

2614 BTP is designed to ensure the delivery of a consistent decision for a business transaction to  
2615 the parties involved, even in the event of failure. Failures can be classified as:

2616  
2617 **Communication failure:** messages between BTP actors are lost and not  
2618 delivered. BTP assumes the carrier protocol ensures that messages are either  
2619 delivered correctly (without corruption) or are lost, but does not assume that all  
2620 losses are reported or that messages sent separately are delivered in the order of  
2621 sending.  
2622

2623 **Node failure (system failure, site failure):** a machine hosting one or more BTP  
2624 actors stops processing and all its volatile data is lost. BTP assumes a site fails by  
2625 stopping – it either operates correctly or not at all, it never operates incorrectly.  
2626

2627  
2628 Communication failure may become known to a BTP implementation by an indication from  
2629 the lower layers or may be inferred (or suspected) by the expiry of a timeout. Recovery from  
2630 a communication failure requires only that the two actors can again send messages to each  
2631 other and continue or complete the progress of the business transaction. In the state tables for  
2632 the Superior:Inferior relationship, each side is either waiting to make a decision or can send a  
2633 message. For some states, the message to be sent is a repetition of a regular message; for  
2634 other states, the INFERIOR\_STATE or SUPERIOR\_STATE message can be sent, requesting  
2635 a response. Thus, following a communication failure, either side can prompt the other to re-  
2636 establish the relationship. Receiving one of the \*\_STATE messages asking for a response  
2637 does not require an immediate response – especially if an implementation is waiting to  
2638 determine a decision (perhaps because it is itself waiting for a decision from elsewhere), an  
2639 implementation may choose not to reply until it wishes too.

2640  
2641 A node failure is distinguished from communication failure because there is loss of volatile  
2642 state. To ensure consistent application of the decision of a business transaction, BTP requires  
2643 that some state information will be persisted despite node failure. Exactly what real events  
2644 correspond to node failure but leave the persistent information undamaged is a matter for  
2645 implementation choice, depending on application requirements; however, for most  
2646 application uses, power failure should be survivable (an exception would be if the data  
2647 manipulated by the associated operations was volatile). There will always be some level of  
2648 event sufficiently catastrophic to lose persistent information and the ability to recover–  
2649 destruction of the computer or bankruptcy of the organisation, for example.

2650  
2651 Recovery from node failure involves recreating the endpoint in a node that has access to the  
2652 persistent information for incomplete transactions. This may be a recreation of the original  
2653 node (including the ability to perform application work) using the same addresses; or there  
2654 may be a distinct recovery entity, which can access the persistent data, but has a different  
2655 address; other implementation approaches are possible. Restoration of the endpoint from  
2656 persistent information will often result in a partial loss of state, relative to the volatile state  
2657 reached before the failure. This is modelled in the state tables by the “disruption” events.  
2658 After recovery from node failure, the implementation behaves much as if a communication  
2659 failure had occurred.

2660  
2661  
2662  
2663  
2664  
2665  
2666  
2667  
2668  
2669  
2670  
2671  
2672  
2673  
2674  
2675  
2676  
2677  
2678  
2679  
2680  
2681  
2682  
2683  
2684  
2685  
2686  
2687  
2688  
2689  
2690  
2691  
2692  
2693  
2694  
2695  
2696  
2697  
2698  
2699  
2700  
2701  
2702  
2703  
2704  
2705  
2706

## Persistent information

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

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

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

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

- Inferior identity (this may be an index used to locate the information)
- Superior address (as on CONTEXT)
- Superior identifier (as on CONTEXT)
- default-is-cancel value (as on PREPARED)

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

2707 A Superior must record corresponding information to allow it to re-establish communication  
2708 with the Inferior:

2709 Inferior address (as on ENROL)

2710 Inferior identifier (as on ENROL)

2711

2712 A Superior that is the Decider for the business transaction need only persist this information  
2713 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A  
2714 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as  
2715 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as  
2716 Superior (to this Inferior) as part of the persistent information of its decision to be prepared  
2717 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and  
2718 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If  
2719 CONFIRM is received, the persistent information may be changed to show the confirm  
2720 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.  
2721 If the persistent information is left unchanged and there is a node failure, on recovery the  
2722 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision  
2723 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

2724

2725 After failure, an implementation may not be able to restore an endpoint to the appropriate  
2726 state immediately – in particular, the necessary persistent information may be inaccessible,  
2727 although the implementation can respond to received BTP messages. In such a case, a  
2728 Superior may reply to any BTP message except INFERIOR\_STATE/\* (i.e. with a “reply-  
2729 requested” value “false”) with SUPERIOR\_STATE/inaccessible and an Inferior to any BTP  
2730 message except SUPERIOR\_STATE/\* with “INFERIOR\_STATE/inaccessible. Receipt of  
2731 the \*\_STATE/inaccessible messages has no effect on the endpoint state.

2732

### 2733 Redirection

2734

2735 As described above, BTP uses the presume-abort model for recovery. A corollary of this is  
2736 that there are cases where one side will attempt to re-establish communication when there is  
2737 no persistent information for the relationship at the far-end. In such cases, it is important the  
2738 side that is attempting recovery can distinguish between unsuccessful attempts to connect to  
2739 the holder of the persistent information and when the information no longer exists. If the peer  
2740 information does not exist, this side can draw conclusions and complete appropriately; if they  
2741 merely fail to get through they are stuck in attempting recovery.

2742

2743 Two mechanisms are provided to make it possible that even when one side of a  
2744 Superior:Inferior relationship has completed, that a message can eventually get through to  
2745 something that can definitively report the status, distinguishing this case from a temporary  
2746 inability to access the state of a continuing transaction element. The mechanisms are:

- 2747 o Address fields which provide a “callback address” can be a set of addresses,  
2748 which are alternatives one of which is chosen as the target address for the  
2749 future message. If the sender of that message finds the address does not work,  
2750 it can try a different alternative.
- 2751 o The REDIRECT message can be used to inform the peer that an address  
2752 previously given is no longer valid and to supply a replacement address (or  
2753 set of addresses). REDIRECT can be issued either as a response to receipt of  
2754 a message or spontaneously.

2755  
2756 The two mechanisms can be used in combination, with one or more of the original set of  
2757 addresses just being a redirector, which does not itself ever have direct access to the state  
2758 information for the transaction, but will respond to any message with an appropriate  
2759 REDIRECT.

2760  
2761 An alternative implementation approach is to have a single addressable entity that uses the  
2762 same address for all transactions, distinguishing them by identifier, and which always  
2763 recovers to use the same address. Such an implementation would not need to supply  
2764 "backup" addresses (and would only use REDIRECT if it was being permanently migrated).

### 2765 Terminator:Decider failures

2766  
2767 BTP does not provide facilities or impose requirements on the recovery of  
2768 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator  
2769 may survive failures (by retaining knowledge of the Decider's address and identifier), but this  
2770 is an implementation option. Although a Decider (if it decides to confirm) will persist  
2771 information about the confirm decision, it is not required, after failure, to remain accessible  
2772 using the inferior address it offered to the Terminator. Any such recovery is an  
2773 implementation option.

2774  
2775 A Decider's address (as returned on BEGUN) may be a set of addresses, allowing a failed  
2776 Decider to be recovered at a different address.

2777  
2778 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and  
2779 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for  
2780 a REQUEST\_CONFIRM that will never arrive, the standard qualifier "Transaction timelimit"  
2781 can be used (by the Initiator) to inform the Decider when it can assume the Terminator will  
2782 not issue REQUEST\_CONFIRM and so it (the Decider) should initiate cancellation.

2783  
2784

## 2785 XML representation of Message Set

2786  
2787 This section describes the syntax for BTP messages in XML. These XML messages represent  
2788 a midpoint between the abstract messages and what actually gets sent on the wire.

2789  
2790 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)  
2791 [3121](#)

2792  
2793 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:xml

2794  
2795 In addition to an XML schema, this specification uses an informal syntax to describe the  
2796 structure of the BTP messages. The syntax appears as an XML instance, but the values  
2797 contain data types instead of values. The following symbols are appended to some of the  
2798 XML constructs: ? (zero or one), \* (zero or more), + (one or more.) The absence of one of  
2799 these symbols corresponds to "one and only one."

2800

### 2801 Addresses

2802

2803 As described in the “Abstract Message and Associated Contracts – Addresses” section, a BTP  
2804 address comprises three parts, and for a target address only the “additional information” field  
2805 is inside the BTP messages. For all BTP messages whose abstract form includes a target  
2806 address parameter, the corresponding XML representation includes a “target-additional-  
2807 information” element. This element may be omitted if it would be empty.  
2808

2809 For other addresses, all three fields are represent, as in:

```
2810 <btp:some-address>  
2811   <btp:binding-name>...carrier binding URI...</btp:binding-name>  
2812   <btp:binding-address>...carrier specific  
2813   URI address...</btp:binding-address>  
2814   <btp:additional-information>...optional additional addressing  
2815   information...</btp:additional-information> ?  
2816 </btp:some-address>
```

2817  
2818  
2819  
2820 A "published" address can be a set of <some-address>, which are alternatives which can be  
2821 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to  
2822 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice  
2823 of which address to use (depending on which binding is preferable.) In the case where  
2824 multiple addresses are used for redundancy, a **priority** attribute can be specified to help the  
2825 receiver choose among the addresses- the address with the highest priority should be used,  
2826 other things being equal. The **priority** is used as a hint and does not enforce any behaviour in  
2827 the receiver of the message. Default priority is a value of 1.  
2828

### 2829 Qualifiers

2830 The “Qualifier name” is used as the element name, within the namespace of the “Qualifier  
2831 group”.

2832

### 2833 Examples:

```
2834 <btpq:inferior-timeout  
2835   xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers"  
2836   xmlns:btp="urn:oasis:names:tc:BTP:xml "  
2837   btp:must-be-understood="false"  
2838   btp:to-be-propagated="false">1800</auth:username>  
2839  
2840 <auth:username  
2841   xmlns:auth="http://www.example.com/ns/auth"  
2842   xmlns:btp="urn:oasis:names:tc:BTP:xml "  
2843   btp:must-be-understood="true"  
2844   btp:to-be-propagated="true">jtauber</auth:username>
```

2845  
2846 Attributes **must-be-understood** has default value “true” and **to-be-propagated** has default  
2847 value “false”.  
2848

### 2849 Identifiers

2850 Unspecified length strings made of up hexadecimal digits (0->9, A->F). Note: lower case a->>f  
2851 are not valid.  
2852



2853 Examples: "01", "FAB224234CCCC2"

2854

2855 Note – Use of hexadecimal digits avoids problems with character-code representations. The  
2856 only operation the BTP implementations have to perform on identifiers is to match them.

2857

## 2858 Message References

2859 Each BTP message has an optional **id** attribute to give it a unique identifier. An application  
2860 can make use of those identifiers, but no processing is enforced.

2861

## 2862 Messages

2863

### 2864 CONTEXT

2865

```
2866 <btp:context id? superior-type="cohesion|atom">  
2867   <btp:superior-address> +  
2868     ...address...  
2869   </btp:superior-address>  
2870   <btp:superior-identifier>...hexstring...</btp:superior-  
2871 identifier>  
2872   <btp:qualifiers> ?  
2873     ...qualifiers...  
2874   </btp:qualifiers>  
2875 </btp:context>
```

2876

2877

### 2878 CONTEXT-REPLY

2879

```
2880 <btp:context-reply id? superior-type="cohesion|atom">  
2881   <btp:superior-address> +  
2882     ...address...  
2883   </btp:superior-address>  
2884   <btp:superior-identifier>...hexstring...</btp:superior-  
2885 identifier>  
2886   <completion-status>completed|related|repudiated</completion-  
2887 status>  
2888   <btp:qualifiers> ?  
2889     ...qualifiers...  
2890   </btp:qualifiers>  
2891 </btp:context>
```

2892

2893

### 2894 BEGIN

2895

```
2896 <btp:begin id? transaction-type="cohesion|atom">  
2897   <btp:target-additional-information>  
2898     ...additional address information...  
2899   </btp:target-additional-information>  
2900   <btp:reply-address>  
2901     ...address...  
2902   </btp:reply-address>  
2903   <btp:qualifiers> ?  
2904     ...qualifiers...
```

2904

2905 </btp:qualifiers>  
2906 </btp:begin>  
2907  
2908

## BEGUN

2909  
2910  
2911 <btp:begin id? transaction-type="cohesion|atom">  
2912 <btp:target-additional-information>  
2913 ...additional address information...  
2914 </btp:target-additional-information>  
2915 <btp:decider-address> ?  
2916 ...address...  
2917 </btp:decider-address>  
2918 <btp:transaction-identifier>...hexstring...</btp:transaction-  
2919 identifier> ?  
2920 <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?  
2921 <btp:inferior-address> ?  
2922 ...address...  
2923 </btp:inferior-address>  
2924 <btp:qualifiers> ?  
2925 ...qualifiers...  
2926 </btp:qualifiers>  
2927 </btp:begin>  
2928  
2929

## ENROL

2930  
2931  
2932 <btp:enrol reply-requested="true|false" id?>  
2933 <btp:target-additional-information>  
2934 ...additional address information...  
2935 </btp:target-additional-information>  
2936 <btp:superior-identifier>...hexstring...</btp:superior-  
2937 identifier>  
2938 <btp:reply-address> ?  
2939 ...address...  
2940 </btp:reply-address>  
2941 <btp:inferior-address> +  
2942 ...address...  
2943 </btp:inferior-address>  
2944 <btp:inferior-identifier>...hexstring...</btp:inferior-  
2945 identifier>  
2946 <btp:qualifiers> ?  
2947 ...qualifiers...  
2948 </btp:qualifiers>  
2949 </btp:enrol>  
2950  
2951

## ENROLLED

2952  
2953  
2954 <btp:enrolled id?>  
2955 <btp:target-additional-information>  
2956 ...additional address information...  
2957 </btp:target-additional-information>

```
2958 <btp:inferior-identifier>...hexstring...</btp:inferior-
2959 identifier>
2960 <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?
2961 <btp:qualifiers> ?
2962 ...qualifiers...
2963 </btp:qualifiers>
2964 </btp:enrolled>
```

## 2967 RESIGN

```
2968
2969 <btp:resign response-requested="true|false" id?>
2970 <btp:target-additional-information>
2971 ...additional address information...
2972 </btp:target-additional-information>
2973 <btp:superior-identifier>...hexstring...</btp:superior-
2974 identifier>
2975 <btp:inferior-address> +
2976 ...address...
2977 </btp:inferior-address>
2978 <btp:inferior-identifier>...hexstring...</btp:inferior-
2979 identifier>
2980 <btp:qualifiers> ?
2981 ...qualifiers...
2982 </btp:qualifiers>
2983 </btp:resign>
```

## 2986 RESIGNED

```
2987
2988 <btp:resigned id?>
2989 <btp:target-additional-information>
2990 ...additional address information...
2991 </btp:target-additional-information>
2992 <btp:inferior-identifier>...hexstring...</btp:inferior-
2993 identifier>
2994 <btp:qualifiers> ?
2995 ...qualifiers...
2996 </btp:qualifiers>
2997 </btp:resigned>
```

## 3000 PREPARE

```
3001
3002 <btp:prepare id?>
3003 <btp:target-additional-information>
3004 ...additional address information...
3005 </btp:target-additional-information>
3006 <btp:inferior-identifier>...hexstring...</btp:inferior-
3007 identifier> ?
3008 <btp:reply-address> ?
3009 ...address...
3010 </btp:reply-address>
```

```
3011 <btp:transaction-identifier>...hexstring...</btp:transaction-
3012 identifier> ?
3013 <btp:inferiors-list> ?
3014 <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3015 +
3016 </btp:inferiors-list>
3017 <btp:qualifiers> ?
3018 ...qualifiers...
3019 </btp:qualifiers>
3020 </btp:prepare>
3021
3022
```

## PREPARED

```
3023
3024
3025 <btp:prepared default-is-cancel="false|true" id?>
3026 <btp:target-additional-information>
3027 ...additional address information...
3028 </btp:target-additional-information>
3029 <btp:superior-identifier>...hexstring...</btp:superior-
3030 identifier>
3031 <btp:inferior-address> +
3032 ...address...
3033 </btp:inferior-address>
3034 <btp:inferior-identifier>...hexstring...</btp:inferior-
3035 identifier>
3036 <btp:qualifiers> ?
3037 ...qualifiers...
3038 </btp:qualifiers>
3039 </btp:prepared>
3040
3041
```

## CONFIRM

```
3042
3043
3044 <btp:confirm id?>
3045 <btp:target-additional-information>
3046 ...additional address information...
3047 </btp:target-additional-information>
3048 <btp:inferior-identifier>...hexstring...</btp:inferior-
3049 identifier>
3050 <btp:qualifiers> ?
3051 ...qualifiers...
3052 </btp:qualifiers>
3053 </btp:confirm>
3054
3055
```

## CONFIRMED

```
3056
3057
3058 <btp:confirmed confirmed-received="true|false" id?>
3059 <btp:target-additional-information>
3060 ...additional address information...
3061 </btp:target-additional-information>
3062 <btp:superior-identifier>...hexstring...</btp:superior-
3063 identifier>
```

```
3064 <btp:inferior-address> ?
3065     ...address...
3066 </btp:inferior-address>
3067 <btp:inferior-identifier>...hexstring...</btp:inferior-
3068 identifier> ?
3069 <btp:decider-address> ?
3070     ...address...
3071 </btp:decider-address>
3072 <btp:transaction-identifier>...hexstring...</btp:transaction-
3073 identifier> ?
3074 <btp:qualifiers> ?
3075     ...qualifiers...
3076 </btp:qualifiers>
3077 </btp:confirmed>
3078
```

### CANCEL

```
3081
3082 <btp:cancel id?>
3083 <btp:target-additional-information>
3084     ...additional address information...
3085 </btp:target-additional-information>
3086 <btp:inferior-identifier>...hexstring...</btp:inferior-
3087 identifier> ?
3088 <btp:reply-address> ?
3089     ...address...
3090 </btp:reply-address>
3091 <btp:transaction-identifier>...hexstring...</btp:transaction-
3092 identifier> ?
3093 <btp:inferiors-list> ?
3094     <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3095 </btp:inferiors-list>
3096 <btp:qualifiers> ?
3097     ...qualifiers...
3098 </btp:qualifiers>
3099 </btp:cancel>
3100
3101
```

### CANCELLED

```
3102
3103
3104 <btp:cancelled id?>
3105 <btp:target-additional-information>
3106     ...additional address information...
3107 </btp:target-additional-information>
3108 <btp:superior-identifier>...hexstring...</btp:superior-
3109 identifier>
3110 <btp:inferior-address> +
3111     ...address...
3112 </btp:inferior-address> ?
3113 <btp:inferior-identifier>...hexstring...</btp:inferior-
3114 identifier> ?
3115 <btp:decider-address> ?
3116     ...address...
3117 </btp:decider-address>
```

```
3118 <btp:transaction-identifier>...hexstring...</btp:transaction-
3119 identifier> ?
3120 <btp:qualifiers> ?
3121 ...qualifiers...
3122 </btp:qualifiers>
3123 </btp:cancelled>
3124
3125
```

## HAZARD

```
3126
3127
3128 <btp:hazard level="mixed|possible" id?>
3129 <btp:target-additional-information>
3130 ...additional address information...
3131 </btp:target-additional-information>
3132 <btp:superior-identifier>...hexstring...</btp:superior-
3133 identifier>
3134 <btp:inferior-address> +
3135 ...address...
3136 </btp:inferior-address>
3137 <btp:inferior-identifier>...hexstring...</btp:inferior-
3138 identifier>
3139 <btp:qualifiers> ?
3140 ...qualifiers...
3141 </btp:qualifiers>
3142 </btp:hazard>
3143
3144
```

## CONTRADICTION

```
3145
3146
3147 <btp:contradiction id?>
3148 <btp:target-additional-information>
3149 ...additional address information...
3150 </btp:target-additional-information>
3151 <btp:inferior-identifier>...hexstring...</btp:inferior-
3152 identifier>
3153 <btp:qualifiers> ?
3154 ...qualifiers...
3155 </btp:qualifiers>
3156 </btp:contradiction>
3157
3158
```

## SUPERIOR\_STATE

```
3159
3160
3161 <btp:superior-state reply-requested="true|false" id?>
3162 <btp:target-additional-information>
3163 ...additional address information...
3164 </btp:target-additional-information>
3165 <btp:inferior-identifier>...hexstring...</btp:inferior-
3166 identifier>
3167 <btp:status>active|prepared-
3168 received|inaccessible|unknown</btp:status>
3169 <btp:qualifiers> ?
3170 ...qualifiers...
```

3171 </btp:qualifiers>  
3172 </btp:superior-state>

3173  
3174  
3175

### INFERIOR\_STATE

3176  
3177

```
<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|prepared-
received|inaccessible|unknown</btp:status>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:inferior-state>
```

3184  
3185  
3186  
3187  
3188  
3189  
3190  
3191  
3192  
3193  
3194  
3195  
3196  
3197

### CONFIRM\_ONE\_PHASE

3198  
3199

```
<btp:confirm-one-phase report-hazard="true|false" id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:confirm-one-phase>
```

3200  
3201  
3202  
3203  
3204  
3205  
3206  
3207  
3208  
3209  
3210  
3211

### REQUEST\_CONFIRM

3212  
3213

```
<btp:request_confirm report-hazard="true|false" id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:reply-address>
    ...address...
  </btp:reply-address>
  <btp:transaction-identifier>...hexstring...</btp:transaction-
identifier>
  <btp:inferiors-list> ?
```

3214  
3215  
3216  
3217  
3218  
3219  
3220  
3221  
3222  
3223

```

3224     <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3225 +
3226 </btp:inferiors-list>
3227 <btp:qualifiers> ?
3228     ...qualifiers...
3229 </btp:qualifiers>
3230 </btp:request_confirm>
3231
3232

```

## REQUEST\_STATUSES

```

3233
3234
3235 <btp:request_statuses id?>
3236 <btp:target-additional-information>
3237     ...additional address information...
3238 </btp:target-additional-information>
3239 <btp:reply-address>
3240     ...address...
3241 </btp:reply-address>
3242 <btp:transaction-identifier>...hexstring...</btp:transaction-
3243 identifier>
3244 <btp:inferiors-list> ?
3245     <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3246 +
3247 </btp:inferiors-list>
3248 <btp:qualifiers> ?
3249     ...qualifiers...
3250 </btp:qualifiers>
3251 </btp:request_statuses>
3252
3253

```

## INFERIOR\_STATUSES

```

3254
3255
3256 <btp:inferior_statuses id?>
3257 <btp:target-additional-information>
3258     ...additional address information...
3259 </btp:target-additional-information>
3260 <btp:decider-address>
3261     ...address...
3262 </btp:decider-address>
3263 <btp:transaction-identifier>...hexstring...</btp:transaction-
3264 identifier>
3265 <btp:status-list>
3266     <btp:status-item> +
3267         <btp:inferior-handle>...hexstring...</btp:inferior-
3268 handle>
3269         <btp:status>active|resigned|preparing|prepared|
3270             autonomously-confirmed|autonomously-cancelled|
3271             confirming|confirmed|cancelling|cancelled|
3272             cancel-contradiction|confirm-contradiction|
3273             hazard</btp:status>
3274     <btp:qualifiers> ?
3275         ...qualifiers...
3276 </btp:qualifiers>
3277 </btp:status-item>

```



```
3278 </btp:status-list>
3279 <btp:qualifiers> ?
3280 ...qualifiers...
3281 </btp:qualifiers>
3282 </btp:inferior_statuses>
```

## REQUEST\_STATUS

```
3285
3286
3287 <btp:request_status id?>
3288 <btp:target-additional-information>
3289 ...additional address information...
3290 </btp:target-additional-information>
3291 <btp:reply-address>
3292 ...address...
3293 </btp:reply-address>
3294 <btp:inferior-identifier>...hexstring...</btp:inferior-
3295 identifier> ?
3296 <btp:transaction-identifier>...hexstring...</btp:transaction-
3297 identifier> ?
3298 <btp:qualifiers> ?
3299 ...qualifiers...
3300 </btp:qualifiers>
3301 </btp:request_status>
```

## STATUS

```
3302
3303
3304
3305
3306 <btp:status id?>
3307 <btp:target-additional-information>
3308 ...additional address information...
3309 </btp:target-additional-information>
3310 <btp:inferior-address> ?
3311 ...address...
3312 </btp:inferior-address>
3313 <btp:inferior-identifier>...hexstring...</btp:inferior-
3314 identifier> ?
3315 <btp:decider-address> ?
3316 ...address...
3317 </btp:decider-address>
3318 <btp:transaction-identifier>...hexstring...</btp:transaction-
3319 identifier> ?
3320 <btp:status-value>created|enrolling|active|resigning|
3321 resigned|preparing|prepared|
3322 confirming|confirmed|cancelling|cancelled|
3323 cancel-contradiction|confirm-contradiction|
3324 hazard|contradicted|unknown|inaccessible</btp:status-
3325 value>
3326 <btp:qualifiers> ?
3327 ...qualifiers...
3328 </btp:qualifiers>
3329 </btp:status>
```

3331  
3332  
3333  
3334  
3335  
3336  
3337  
3338  
3339  
3340  
3341  
3342  
3343  
3344  
3345  
3346  
3347  
3348  
3349  
3350  
3351  
3352  
3353

## REDIRECT

```
<btp:redirect id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:superior-identifier>...hexstring...</btp:superior-
identifier> ?
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
  <btp:old-address> +
    ...address...
  </btp:old-address>
  <btp:new-address> +
    ...address...
  </btp:new-address>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:redirect>
```

3354  
3355  
3356  
3357  
3358  
3359  
3360  
3361  
3362  
3363  
3364  
3365  
3366  
3367  
3368  
3369  
3370  
3371

## FAULT

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

3372  
3373  
3374  
3375  
3376  
3377  
3378  
3379  
3380  
3381  
3382

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

- o general
- o unknown-parameter
- o wrong-state
- o communication-failure
- o invalid-superior
- o duplicate-inferior
- o unknown-inferior

3383 Revisions of this specification may add other fault type names, which shall be simple strings  
3384 of letters, numbers and hyphens. If other specifications define fault type names to be used  
3385 with BTP, the names shall be URIs.

3386  
3387 Fault data can take on various forms:

3388  
3389 Free text:

3390  
3391 `<btp:fault-data>...string data...</btp:fault-data>`  
3392

3393 Identifier:

3394  
3395 `<btp:fault-data>...hexstring...</btp:fault-data>`  
3396

3397  
3398 Inferior Identity:

3399  
3400 `<btp:fault-data>`  
3401 `<btp:inferior-address> +`  
3402 `...address...`  
3403 `</btp:inferior-address>`  
3404 `<btp:inferior-identifier>...hexstring...</btp:inferior-`  
3405 `identifier>`  
3406 `</btp:fault-data>`  
3407  
3408

### 3409 **Standard qualifiers**

3410 The informal syntax for these messages assumes the namespace prefix “btpq” is associated  
3411 with the URI “urn:oasis:names:tc:BTP:qualifiers”.

3412

### 3413 **Transaction timelimit**

3414  
3415 `<btpq:transaction-timelimit>`  
3416 `<btpq:timelimit>`  
3417 `...time in seconds...`  
3418 `</btpq:timelimit>`  
3419 `</btpq:transaction-timelimit>`  
3420

3420

### 3421 **Inferior timeout**

3422 `<btpq:inferior-timeout>`  
3423 `<btpq:timeout>`  
3424 `...time in seconds...`  
3425 `</btpq:timeout>`  
3426 `<btpq:intended-decision>confirm|cancel</btpq:intended-decision>`  
3427 `</btpq:inferior-timeout>`  
3428

3428

### 3429 **Minimum inferior timeout**

3430 `<btpq:minimum-inferior-timeout>`  
3431 `<btpq:minimum-timeout>`  
3432 `...time in seconds...`  
3433 `</btpq:minimum-timeout>`

3433

3434 `</btpq:minimum-inferior-timeout>`  
3435

## 3436 **Compounding of Messages**

3437  
3438 Bundling (semantically insignificant combination) of BTP messages is indicated with the  
3439 "btp:messages" element, with the bundled messages as child elements. For example:

```
3440 <btp:messages>  
3441   <btp:enrol>...</btp:enrol>  
3442   <btp:prepared>...</btp:prepared>  
3443 </btp:messages>
```

3444  
3445  
3446 Relating BTP messages to one another is achieved through containment. For example:

```
3447 <btp:context-reply>  
3448   ...<completion-status>related</completion-status> ...  
3449   <btp:enrol>...</btp:enrol>  
3450 </btp:context-reply>
```

3451  
3452  
3453  
3454 The carrier protocol binding specifies how a relation between application and BTP messages  
3455 is represented.  
3456

3457

3458

## Carrier Protocol Bindings

3459

3460

3461

3462

3463

3464

3465

3466

3467

3468

3469

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

3470

### Carrier Protocol Binding Proforma

3471

3472

3473

A BTP carrier binding specification should provide the following information:

3474

3475

3476

3477

3478

3479

3480

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

3481

3482

3483

3484

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

3485

3486

3487

3488

3489

**BTP message representation:** This section will define how BTP messages are represented. For many bindings, this will be the normal string encoding of the XML, in accordance with the XML schema defined in this document.

3490

3491

3492

3493

3494

3495

3496

3497

3498

3499

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

3500

3501

3502

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

3503 specification could just say “the CONTEXT may be carried as a parameter of an  
3504 application invocation”). Alternatively, the binding may specify a general method that  
3505 represents the relationship between application and BTP messages.  
3506

3507 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly  
3508 but are treated as implicit in application messages or other BTP messages. This may depend  
3509 on particular parameter values of the BTP messages or the application messages.  
3510

3511 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier  
3512 protocol and of BTP shall be defined. This may include definition of which carrier protocol  
3513 exceptions are equivalent to a FAULT/communication-failure message.  
3514

3515 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.  
3516 If BTP addresses with different bindings are be considered to match (for purposes of  
3517 identifying the peer Superior/Inferior and redirection), this should be specified here.  
3518

3519 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are  
3520 imposed by use of this binding should be listed. This would include limitations on which  
3521 messages can be sent, which event sequences are supported and restrictions on parameter  
3522 values. Such limitations may reduce the usefulness of an implementation, but may be  
3523 appropriate in certain environments.  
3524

3525 **Other:** Other features of the binding, especially any that will potentially affect interoperability  
3526 should be specified here. This may include restrictions or requirements on the use or support  
3527 of optional carrier parameters or mechanisms>  
3528

## 3529 **SOAP Binding**

3530  
3531 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)  
3532 specification.  
3533

3534 **Binding name:** soap-http-1

3535  
3536 **Binding address format:** shall be a URL of type HTTP.

3537  
3538 **BTP message representation:** The string representation of the XML, as specified in the  
3539 XML schema defined in this document shall be used. BTP messages conform to the  
3540 rules of the Section 5 (of the SOAP 1.1 specification) SOAP Encoding as specified by the  
3541 URI: "http://schemas.xmlsoap.org/soap/encoding".  
3542

3543 **Mapping for BTP messages (unrelated):** If no application message is being sent at the  
3544 same time, BTP messages shall be contained in a btp:messages element which shall be an  
3545 immediate child element of the SOAP-Body. There shall be precisely one btp:messages  
3546 element. Any number of BTP messages with the same binding address in their target  
3547 address may be carried in the same btp:messages element.  
3548

3549 If an application message is being sent at the same time, the mapping for related  
3550 messages shall be used, as if the BTP messages were related to the application  
3551 message. (There is no ambiguity in whether the BTP messages are related, because  
3552 only CONTEXT can be related to an application message.)  
3553

3554 **Mapping for BTP messages related to application messages:** All BTP messages sent  
3555 with an application message, whether related to the application message or not, shall  
3556 be sent in a single btp:messages element in the SOAP:Header. There shall be  
3557 precisely one btp:messages element in the SOAP:Header.  
3558

3559 **Implicit messages:** A SOAP fault, or other communication failure received in response to a  
3560 SOAP request that had a CONTEXT in the SOAP:Header shall be treated as if a  
3561 CONTEXT\_REPLY/repudiated had been received. See also the discussion under “other”  
3562 about the SOAP mustUnderstand attribute.  
3563

3564 **Faults:** A SOAP fault or other communication failure shall be treated as  
3565 FAULT/communication-failure.  
3566

3567 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding  
3568 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-  
3569 http-1” if the binding address and additional information fields match.  
3570

3571 **Limitations on BTP use:** None  
3572

3573 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor  
3574 attribute. The SOAPAction HTTP header is left to be application specific when there are  
3575 application messages in the SOAP:Body, as an already existing web service that is being  
3576 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP  
3577 header shall be omitted when the SOAP message carries only BTP messages in the  
3578 SOAP:Body.  
3579

3580 The SOAP mustUnderstand attribute, when used on the btp:messages containing a the BTP  
3581 CONTEXT, ensures that the server (as a whole) determines whether any enrolments are  
3582 necessary and reply with CONTEXT\_REPLY as appropriate. If mustUnderstand if false, a  
3583 server can ignore the CONTEXT (if BTP is not supported there). It is an implementation or  
3584 configuration option whether a CONTEXT\_REPLY/ok is assumed to be implicit in the HTTP  
3585 response in such a case. (If no CONTEXT\_REPLY/ok is assumed, it will be impossible for  
3586 the business transaction to confirm) .  
3587

---

3588 Note – some SOAP implementations may not support the mustUnderstand  
3589 attribute sufficiently to enforce these requirements. If using such an  
3590 implementation on the service side, it may be necessary to assume an  
3591 CONTEXT\_REPLY/ok.

---

3592

## Example scenario using SOAP binding

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

```
3593 <soap:Envelope
3594     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3595     soap-
3596     env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
3597
3598     <soap:Header>
3599
3600         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
3601             <btp:context superior-type="atom">
3602                 <btp:superior-address>
3603                     <btp:binding>soap-http-1</btp:binding>
3604                     <btp:binding-
3605 address>http://client.example.com/soaphandler</btp:binding-
3606 address>
3607                 <btp:additional-information>btpengine</btp:additional-
3608 information>
3609                 </btp:superior-address>
3610                 <btp:superior-identifier>1001</btp:superior-identifier>
3611                 <btp:qualifiers>
3612                     <btpq:transaction-timelimit
3613 xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers">1800</btpq:transact
3614 ion-timelimit>
3615                     </btp:qualifiers>
3616                 </btp:context>
3617             </btp:messages>
3618
3619         </soap:Header>
3620
3621         <soap:Body>
3622
3623             <ns1:orderGoods
3624 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
3625                 <custID>ABC8329045</custID>
3626                 <itemID>224352</itemID>
3627                 <quantity>5</quantity>
3628             </ns1:orderGoods>
3629
3630         </soap:Body>
3631
3632     </soap:Envelope>
```

The example below shows CONTEXT\_REPLY and a related (and therefore contained) ENROL message sent from services.example.com to client.example.com, in reply to the previous message. There is no application response, so the BTP messages are in the SOAP:Body.



```

3646 <soap:Envelope
3647     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3648     soap-
3649     env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
3650
3651     <soap:Header>
3652     </soap:Header>
3653
3654     <soap:Body>
3655
3656         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
3657             <btp:context-reply>
3658                 <btp:superior-address>
3659                     <btp:binding>soap-http-1</btp:binding>
3660                     <btp:binding-address>
3661                         http://client.example.com/soaphandler
3662                     </btp:binding-address>
3663                     <btp:additional-information>
3664                         btpengine
3665                     </btp:additional-information>
3666                 </btp:superior-address>
3667                 <btp:superior-identifier>1001</btp:superior-identifier>
3668                 <completion-status>related</completion-status>
3669
3670                 <btp:enrol reply-requested="false">
3671                     <btp:target-additional-information>
3672                         btpengine
3673                     </btp:target-additional-information>
3674                     <btp:superior-identifier>
3675                         1001
3676                     </btp:superior-identifier>
3677                     <btp:inferior-address>
3678                         <btp:binding>soap-http-1</btp:binding>
3679                         <btp:binding-address>
3680                             http://services.example.com/soaphandler
3681                         </btp:binding-address>
3682                     </btp:inferior-address>
3683                     <btp:inferior-identifier>
3684                         AAAB
3685                     </btp:inferior-identifier>
3686                 </btp:enrol>
3687
3688             </btp:context-reply>
3689
3690         </btp:messages>
3691
3692     </soap:Body>
3693
3694 </soap:Envelope>
3695
3696
3697

```

3698 **SOAP + Attachments Binding**

3699

3700 This binding describes how BTP messages will be carried using SOAP as in the [SOAP](#)  
3701 [Messages with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-  
3702 http-1. The two bindings only differ when application messages are sent  
3703

3704 **Binding name:** soap-attachments-http-1

3705

3706 **Binding address format:** [as for soap-http-1](#)

3707

3708 **BTP message representation:** As for soap-http-1

3709

3710 **Mapping for BTP messages (unrelated):** As for “soap-http-1” , except the  
3711 SOAP:Envelope containing the SOAP-Body containing the BTP messages shall be in  
3712 a MIME body part, as specified in [SOAP Messages with Attachments](#) specification. If an  
3713 application message is being sent at the same time, the mapping for related messages  
3714 for this binding shall be used, as if the BTP messages were related to the application  
3715 message(s).

3716

3717 **Mapping for BTP messages related to application messages:** MIME packaging shall be  
3718 used. One of the MIME multipart/related parts shall contain a SOAP:Envelope,  
3719 whose SOAP:Headers element shall contain precisely one btp:messages element,  
3720 containing any BTP messages. Any BTP CONTEXT in the btp:messages is  
3721 considered to be related to the application message(s) in the SOAP:Body, and to also  
3722 any of the MIME parts referenced from the SOAP:Body (using the “href” attribute).

3723

3724 **Implicit messages:** As for soap-http-1.

3725

3726 **Faults:** As for soap-http-1.

3727

3728 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding  
3729 string “soap-http-1” is considered to match one that has the binding string “soap-  
3730 attachments-http-1” if the binding address and additional information fields match.

3731

3732 **Limitations on BTP use:** None

3733

3734 **Other:** As for soap-http-1

3735

3736 *Example using SOAP + Attachments binding*

3737

```
3738 MIME-Version: 1.0  
3739 Content-Type: Multipart/Related; boundary=MIME_boundary;  
3740 type=text/xml;  
3741         start="someID"  
3742  
3743 --MIME_boundary  
3744 Content-Type: text/xml; charset=UTF-8
```

```

3745 Content-ID: someID
3746
3747 <?xml version='1.0' ?>
3748 <soap:Envelope
3749     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3750     soap-
3751 env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
3752
3753     <soap:Header>
3754
3755         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
3756             <btp:context superior-type="atom">
3757                 <btp:superior-address>
3758                     <btp:binding>soap-http-1</btp:binding>
3759                     <btp:binding-address>
3760                         http://client.example.com/soaphandler
3761                     </btp:binding-address>
3762                 </btp:superior-address>
3763                 <btp:superior-identifier>1001</btp:superior-identifier>
3764             </btp:context>
3765         </btp:messages>
3766
3767     </soap:Header>
3768
3769     <soap:Body>
3770         <orderGoods href="cid:anotherID"/>
3771     </soap:Body>
3772
3773 </soap:Envelope>
3774
3775 --MIME_boundary
3776 Content-Type: text/xml
3777 Content-ID: anotherID
3778
3779     <ns1:orderGoods
3780 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
3781         <custID>ABC8329045</custID>
3782         <itemID>224352</itemID>
3783         <quantity>5</quantity>
3784     </ns1:orderGoods>
3785
3786
3787 --MIME_boundary--
3788
3789

```

### XML Schema for SOAP Bindings

```

3790
3791
3792 <?xml version="1.0"?>
3793 <schema targetNamespace="urn:oasis:names:tc:BTP:xml"
3794     xmlns="http://www.w3.org/2001/XMLSchema"
3795     xmlns:tns="urn:oasis:names:tc:BTP:xml">
3796
3797     <complexType name="qualifier_type">

```

```

3798     <simpleContent>
3799         <extension base="string">
3800             <attribute name="must-be-understood" type="boolean"/>
3801             <attribute name="to-be-propagated" type="boolean"/>
3802         </extension>
3803     </simpleContent>
3804 </complexType>
3805 <element name="qualifier" type="tns:qualifier_type"/>
3806 <element name="qualifiers">
3807     <complexType>
3808         <sequence>
3809             <element ref="tns:qualifier" maxOccurs="unbounded"/>
3810         </sequence>
3811     </complexType>
3812 </element>
3813
3814 <complexType name="address">
3815     <sequence>
3816         <element name="binding-name" type="string"/>
3817         <element name="binding-address" type="string"/>
3818         <element name="additional-information" type="string"
3819 minOccurs="0"/>
3820     </sequence>
3821 </complexType>
3822
3823 <simpleType name="identifier">
3824     <restriction base="string">
3825         <pattern value="([0-9,A-Z])*"/>
3826     </restriction>
3827 </simpleType>
3828
3829 <simpleType name="superior-type">
3830     <restriction base="string">
3831         <enumeration value="cohesion"/>
3832         <enumeration value="atom"/>
3833     </restriction>
3834 </simpleType>
3835
3836 <simpleType name="transaction-type">
3837     <restriction base="string">
3838         <enumeration value="cohesion"/>
3839         <enumeration value="atom"/>
3840     </restriction>
3841 </simpleType>
3842
3843
3844 <element name="context">
3845     <complexType>
3846         <sequence>
3847             <element name="superior-address" type="tns:address"
3848 maxOccurs="unbounded"/>
3849             <element name="superior-identifier"
3850 type="tns:identifier"/>
3851             <element ref="tns:qualifiers" minOccurs="0"/>

```

```

3852         </sequence>
3853         <attribute name="id" type="ID" use="optional"/>
3854         <attribute name="superior-type" type="tns:superior-type"
3855 use="required"/>
3856     </complexType>
3857 </element>
3858
3859     <element name="context-reply">
3860         <complexType>
3861             <sequence>
3862                 <element name="superior-address" type="tns:address"
3863 maxOccurs="unbounded"/>
3864                 <element name="superior-identifier"
3865 type="tns:identifier"/>
3866                 <element name="completion-status">
3867                     <simpleType>
3868                         <restriction base="string">
3869                             <enumeration value="completed"/>
3870                             <enumeration value="related"/>
3871                             <enumeration value="repudiated"/>
3872                         </restriction>
3873                     </simpleType>
3874                 </element>
3875                 <element ref="tns:qualifiers" minOccurs="0"/>
3876             </sequence>
3877             <attribute name="id" type="ID"/>
3878             <attribute name="superior-type" type="tns:superior-type"
3879 use="required"/>
3880         </complexType>
3881     </element>
3882
3883     <element name="begin">
3884         <complexType>
3885             <sequence>
3886                 <element name="target-additional-information"
3887 type="string"/>
3888                 <element name="reply-address" type="tns:address"/>
3889                 <element ref="tns:qualifiers" minOccurs="0"/>
3890             </sequence>
3891             <attribute name="id" type="ID"/>
3892             <attribute name="transaction-type" type="tns:superior-type"
3893 use="required"/>
3894         </complexType>
3895     </element>
3896
3897     <element name="begun">
3898         <complexType>
3899             <sequence>
3900                 <element name="target-additional-information"
3901 type="string"/>
3902                 <element name="decider-address" type="tns:address"
3903 minOccurs="0"/>
3904                 <element name="transaction-identifier"
3905 type="tns:identifier" minOccurs="0"/>

```

```

3906         <element name="inferior-handle" type="tns:identifier"
3907 minOccurs="0" />
3908         <element name="inferior-address" type="tns:address"
3909 minOccurs="0" />
3910         <element ref="tns:qualifiers" minOccurs="0" />
3911     </sequence>
3912     <attribute name="id" type="ID" />
3913     <attribute name="transaction-type" type="tns:superior-type"
3914 use="required" />
3915 </complexType>
3916 </element>
3917
3918     <element name="enrol">
3919         <complexType>
3920             <sequence>
3921                 <element name="target-additional-information"
3922 type="string" />
3923                 <element name="superior-identifier"
3924 type="tns:identifier" />
3925                 <element name="reply-address" type="tns:address"
3926 minOccurs="0" />
3927                 <element name="inferior-address" type="tns:address"
3928 minOccurs="1" maxOccurs="unbounded" />
3929                 <element name="inferior-identifier"
3930 type="tns:identifier" />
3931                 <element ref="tns:qualifiers" minOccurs="0" />
3932             </sequence>
3933             <attribute name="id" type="ID" />
3934             <attribute name="reply-requested" type="boolean" />
3935         </complexType>
3936     </element>
3937
3938
3939     <element name="enrolled">
3940         <complexType>
3941             <sequence>
3942                 <element name="target-additional-information"
3943 type="string" />
3944                 <element name="inferior-identifier"
3945 type="tns:identifier" />
3946                 <element name="inferior-handle" type="tns:identifier"
3947 minOccurs="0" />
3948                 <element ref="tns:qualifiers" minOccurs="0" />
3949             </sequence>
3950             <attribute name="id" type="ID" />
3951         </complexType>
3952     </element>
3953
3954     <element name="resign">
3955         <complexType>
3956             <sequence>
3957                 <element name="target-additional-information"
3958 type="string" />

```

```

3959         <element name="superior-identifier"
3960 type="tns:identifier"/>
3961         <element name="inferior-address" type="tns:address"
3962 minOccurs="1" maxOccurs="unbounded"/>
3963         <element name="inferior-identifier"
3964 type="tns:identifier"/>
3965         <element ref="tns:qualifiers" minOccurs="0"/>
3966     </sequence>
3967     <attribute name="id" type="ID"/>
3968     <attribute name="response-requested" type="boolean"/>
3969 </complexType>
3970 </element>
3971
3972     <element name="resigned">
3973         <complexType>
3974             <sequence>
3975                 <element name="target-additional-information"
3976 type="string"/>
3977                 <element name="inferior-identifier"
3978 type="tns:identifier"/>
3979                 <element ref="tns:qualifiers" minOccurs="0"/>
3980             </sequence>
3981             <attribute name="id" type="ID"/>
3982         </complexType>
3983     </element>
3984
3985     <element name="prepare">
3986         <complexType>
3987             <sequence>
3988                 <element name="target-additional-information"
3989 type="string"/>
3990                 <element name="inferior-identifier"
3991 type="tns:identifier" minOccurs="0"/>
3992                 <element name="reply-address" type="tns:address"
3993 minOccurs="0"/>
3994                 <element name="transaction-identifier"
3995 type="tns:identifier" minOccurs="0"/>
3996                 <element name="inferiors-list" minOccurs="0">
3997                     <complexType>
3998                         <sequence>
3999                             <element name="inferior-handle"
4000 type="tns:identifier" maxOccurs="unbounded"/>
4001                         </sequence>
4002                     </complexType>
4003                 </element>
4004                 <element ref="tns:qualifiers" minOccurs="0"/>
4005             </sequence>
4006             <attribute name="id" type="ID"/>
4007         </complexType>
4008     </element>
4009
4010     <element name="prepared">
4011         <complexType>
4012             <sequence>

```

```

4013         <element name="target-additional-information"
4014 type="string"/>
4015         <element name="superior-identifier"
4016 type="tns:identifier"/>
4017         <element name="inferior-address" type="tns:address"
4018 maxOccurs="unbounded"/>
4019         <element name="inferior-identifier"
4020 type="tns:identifier"/>
4021         <element ref="tns:qualifiers" minOccurs="0"/>
4022     </sequence>
4023     <attribute name="id" type="ID"/>
4024     <attribute name="default-is-cancel" type="boolean"/>
4025 </complexType>
4026 </element>
4027
4028     <element name="confirm">
4029         <complexType>
4030             <sequence>
4031                 <element name="target-additional-information"
4032 type="string"/>
4033                 <element name="inferior-identifier"
4034 type="tns:identifier"/>
4035                 <element ref="tns:qualifiers" minOccurs="0"/>
4036             </sequence>
4037             <attribute name="id" type="ID"/>
4038         </complexType>
4039     </element>
4040
4041     <element name="confirmed">
4042         <complexType>
4043             <sequence>
4044                 <element name="target-additional-information"
4045 type="string"/>
4046                 <element name="superior-identifier"
4047 type="tns:identifier"/>
4048                 <element name="inferior-address" type="tns:address"
4049 minOccurs="0"/>
4050                 <element name="inferior-identifier"
4051 type="tns:identifier" minOccurs="0"/>
4052                 <element name="decider-address" type="tns:address"
4053 minOccurs="0"/>
4054                 <element name="transaction-identifier"
4055 type="tns:identifier" minOccurs="0"/>
4056                 <element ref="tns:qualifiers" minOccurs="0"/>
4057             </sequence>
4058             <attribute name="id" type="ID"/>
4059             <attribute name="confirmed-received" type="boolean"/>
4060         </complexType>
4061     </element>
4062
4063     <element name="cancel">
4064         <complexType>
4065             <sequence>

```



```

4066         <element name="target-additional-information"
4067 type="string"/>
4068         <element name="inferior-identifier"
4069 type="tns:identifier" minOccurs="0"/>
4070         <element name="reply-address" type="tns:address"
4071 minOccurs="0"/>
4072         <element name="transaction-identifier"
4073 type="tns:identifier" minOccurs="0"/>
4074         <element name="decider-address" type="tns:address"
4075 minOccurs="0"/>
4076         <element name="transaction-identifier"
4077 type="tns:identifier" minOccurs="0"/>
4078         <element name="inferiors-list" minOccurs="0">
4079             <complexType>
4080                 <sequence>
4081                     <element name="inferior-handle"
4082 type="tns:identifier" maxOccurs="unbounded"/>
4083                 </sequence>
4084             </complexType>
4085         </element>
4086         <element ref="tns:qualifiers" minOccurs="0"/>
4087     </sequence>
4088     <attribute name="id" type="ID"/>
4089 </complexType>
4090 </element>
4091
4092     <element name="cancelled">
4093         <complexType>
4094             <sequence>
4095                 <element name="target-additional-information"
4096 type="string"/>
4097                 <element name="superior-identifier"
4098 type="tns:identifier"/>
4099                 <element name="inferior-address" type="tns:address"
4100 maxOccurs="unbounded"/>
4101                 <element name="inferior-identifier"
4102 type="tns:identifier" minOccurs="0"/>
4103                 <element name="decider-address" type="tns:address"
4104 minOccurs="0"/>
4105                 <element name="transaction-identifier"
4106 type="tns:identifier" minOccurs="0"/>
4107                 <element ref="tns:qualifiers" minOccurs="0"/>
4108             </sequence>
4109             <attribute name="id" type="ID"/>
4110         </complexType>
4111     </element>
4112
4113     <element name="hazard">
4114         <complexType>
4115             <sequence>
4116                 <element name="target-additional-information"
4117 type="string"/>
4118                 <element name="superior-identifier"
4119 type="tns:identifier"/>

```

```

4120         <element name="inferior-address" type="tns:address"
4121 maxOccurs="unbounded" />
4122         <element name="inferior-identifier"
4123 type="tns:identifier" />
4124         <element ref="tns:qualifiers" minOccurs="0" />
4125     </sequence>
4126     <attribute name="id" type="ID" />
4127 </complexType>
4128 </element>
4129
4130 <element name="contradiction">
4131     <complexType>
4132         <sequence>
4133             <element name="target-additional-information"
4134 type="string" />
4135             <element name="inferior-identifier"
4136 type="tns:identifier" />
4137             <element ref="tns:qualifiers" minOccurs="0" />
4138         </sequence>
4139         <attribute name="id" type="ID" />
4140     </complexType>
4141 </element>
4142
4143 <element name="superior-state">
4144     <complexType>
4145         <sequence>
4146             <element name="target-additional-information"
4147 type="string" />
4148             <element name="inferior-identifier"
4149 type="tns:identifier" />
4150             <element name="status">
4151                 <simpleType>
4152                     <restriction base="string">
4153                         <enumeration value="active" />
4154                         <enumeration value="prepared-received" />
4155                         <enumeration value="inaccessible" />
4156                         <enumeration value="unknown" />
4157                     </restriction>
4158                 </simpleType>
4159             </element>
4160             <element ref="tns:qualifiers" minOccurs="0" />
4161         </sequence>
4162         <attribute name="id" type="ID" />
4163         <attribute name="reply-requested" type="boolean" />
4164     </complexType>
4165 </element>
4166
4167 <element name="inferior-state">
4168     <complexType>
4169         <sequence>
4170             <element name="target-additional-information"
4171 type="string" />
4172             <element name="superior-identifier"
4173 type="tns:identifier" />

```

```

4174         <element name="inferior-address" type="tns:address"
4175 maxOccurs="unbounded" />
4176         <element name="inferior-identifier"
4177 type="tns:identifier" />
4178         <element name="status">
4179             <simpleType>
4180                 <restriction base="string">
4181                     <enumeration value="active" />
4182                     <enumeration value="prepared-received" />
4183                     <enumeration value="inaccessible" />
4184                     <enumeration value="unknown" />
4185                 </restriction>
4186             </simpleType>
4187         </element>
4188         <element ref="tns:qualifiers" minOccurs="0" />
4189     </sequence>
4190     <attribute name="id" type="ID" />
4191     <attribute name="reply-requested" type="boolean" />
4192 </complexType>
4193 </element>
4194
4195     <element name="confirm-one-phase">
4196         <complexType>
4197             <sequence>
4198                 <element name="target-additional-information"
4199 type="string" />
4200                 <element name="inferior-identifier"
4201 type="tns:identifier" />
4202                 <element ref="tns:qualifiers" minOccurs="0" />
4203             </sequence>
4204             <attribute name="id" type="ID" />
4205             <attribute name="report-hazard" type="boolean" />
4206         </complexType>
4207     </element>
4208
4209     <element name="request-confirm">
4210         <complexType>
4211             <sequence>
4212                 <element name="target-additional-information"
4213 type="string" />
4214                 <element name="reply-address" type="tns:address" />
4215                 <element name="transaction-identifier"
4216 type="tns:identifier" />
4217                 <element name="inferiors-list" minOccurs="0">
4218                     <complexType>
4219                         <sequence>
4220                             <element name="inferior-handle"
4221 type="tns:identifier" maxOccurs="unbounded" />
4222                         </sequence>
4223                     </complexType>
4224                 </element>
4225                 <element ref="tns:qualifiers" minOccurs="0" />
4226             </sequence>
4227             <attribute name="id" type="ID" />

```

```

4228         <attribute name="report-hazard" type="boolean"/>
4229     </complexType>
4230 </element>
4231
4232     <element name="request-statuses">
4233         <complexType>
4234             <sequence>
4235                 <element name="target-additional-information"
4236 type="string"/>
4237                 <element name="reply-address" type="tns:address"/>
4238                 <element name="transaction-identifier"
4239 type="tns:identifier"/>
4240                 <element name="inferiors-list" minOccurs="0">
4241                     <complexType>
4242                         <sequence>
4243                             <element name="inferior-handle"
4244 type="tns:identifier" maxOccurs="unbounded"/>
4245                         </sequence>
4246                     </complexType>
4247                 </element>
4248                 <element ref="tns:qualifiers" minOccurs="0"/>
4249             </sequence>
4250             <attribute name="id" type="ID"/>
4251         </complexType>
4252     </element>
4253
4254     <element name="inferior-statuses">
4255         <complexType>
4256             <sequence>
4257                 <element name="target-additional-information"
4258 type="string"/>
4259                 <element name="decider-address" type="tns:address"/>
4260                 <element name="transaction-identifier"
4261 type="tns:identifier"/>
4262                 <element name="status-list">
4263                     <complexType>
4264                         <sequence>
4265                             <element name="status-item" maxOccurs="unbounded">
4266                                 <complexType>
4267                                     <sequence>
4268                                         <element name="inferior-handle"
4269 type="tns:identifier"/>
4270                                         <element name="status">
4271                                             <simpleType>
4272                                                 <restriction base="string">
4273                                                     <enumeration value="active"/>
4274                                                     <enumeration value="resigned"/>
4275                                                     <enumeration value="preparing"/>
4276                                                     <enumeration value="prepared"/>
4277                                                     <enumeration value="autonomously-
4278 confirmed"/>
4279                                                     <enumeration value="autonomously-
4280 cancelled"/>
4281                                                     <enumeration value="confirming"/>

```

```

4282         <enumeration value="confirmed"/>
4283         <enumeration value="cancelling"/>
4284         <enumeration value="cancelled"/>
4285         <enumeration value="cancel-contradiction"/>
4286         <enumeration value="confirm-contradiction"/>
4287         <enumeration value="hazard"/>
4288     </restriction>
4289     </simpleType>
4290 </element>
4291     <element ref="tns:qualifiers" minOccurs="0"/>
4292 </sequence>
4293 </complexType>
4294 </element>
4295 </sequence>
4296 </complexType>
4297 </element>
4298     <element ref="tns:qualifiers" minOccurs="0"/>
4299 </sequence>
4300     <attribute name="id" type="ID"/>
4301 </complexType>
4302 </element>
4303
4304 <element name="request-status">
4305     <complexType>
4306         <sequence>
4307             <element name="target-additional-information"
4308 type="string"/>
4309             <element name="reply-address" type="tns:address"/>
4310             <element name="inferior-identifier"
4311 type="tns:identifier" minOccurs="0"/>
4312             <element name="transaction-identifier"
4313 type="tns:identifier" minOccurs="0"/>
4314             <element ref="tns:qualifiers" minOccurs="0"/>
4315         </sequence>
4316         <attribute name="id" type="ID"/>
4317     </complexType>
4318 </element>
4319
4320 <element name="status">
4321     <complexType>
4322         <sequence>
4323             <element name="target-additional-information"
4324 type="string"/>
4325             <element name="inferior-address" type="tns:address"
4326 minOccurs="0"/>
4327             <element name="inferior-identifier"
4328 type="tns:identifier" minOccurs="0"/>
4329             <element name="decider-address" type="tns:address"
4330 minOccurs="0"/>
4331             <element name="transaction-identifier"
4332 type="tns:identifier" minOccurs="0"/>
4333             <element name="status-value">
4334                 <simpleType>
4335                     <restriction base="string">

```

```

4336         <enumeration value="created" />
4337         <enumeration value="enrolling" />
4338         <enumeration value="active" />
4339         <enumeration value="resigning" />
4340         <enumeration value="resigned" />
4341         <enumeration value="preparing" />
4342         <enumeration value="prepared" />
4343         <enumeration value="confirming" />
4344         <enumeration value="confirmed" />
4345         <enumeration value="cancelling" />
4346         <enumeration value="cancelled" />
4347         <enumeration value="cancel-contradiction" />
4348         <enumeration value="confirm-contradiction" />
4349         <enumeration value="hazard" />
4350         <enumeration value="contradicted" />
4351         <enumeration value="unknown" />
4352         <enumeration value="inaccessible" />
4353     </restriction>
4354 </simpleType>
4355 </element>
4356 <element ref="tns:qualifiers" minOccurs="0" />
4357 </sequence>
4358 <attribute name="id" type="ID" />
4359 </complexType>
4360 </element>
4361
4362 <element name="redirect">
4363 <complexType>
4364 <sequence>
4365 <element name="target-additional-information"
4366 type="string" />
4367 <element name="superior-identifier"
4368 type="tns:identifier" minOccurs="0" />
4369 <element name="inferior-identifier"
4370 type="tns:identifier" />
4371 <element name="old-address" type="tns:address"
4372 maxOccurs="unbounded" />
4373 <element name="new-address" type="tns:address"
4374 maxOccurs="unbounded" />
4375 <element ref="tns:qualifiers" minOccurs="0" />
4376 </sequence>
4377 <attribute name="id" type="ID" />
4378 </complexType>
4379 </element>
4380
4381 <element name="fault">
4382 <complexType>
4383 <sequence>
4384 <element name="target-additional-information"
4385 type="string" />
4386 <element name="superior-identifier"
4387 type="tns:identifier" minOccurs="0" />
4388 <element name="inferior-identifier"
4389 type="tns:identifier" minOccurs="0" />

```

```
4390         <element name="fault-type" type="string"/>
4391         <element name="fault-data" type="anyType"
minOccurs="0"/>
4392
4393         <element ref="tns:qualifiers" minOccurs="0"/>
4394     </sequence>
4395     <attribute name="id" type="ID"/>
4396 </complexType>
4397 </element>
4398
4399 </schema>
4400
```

4400

4401

## 4402 **Conformance**

4403

4404

4405

4406

4407

4408

4409

4410

4411

4412

4413

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:

<b>Role Group</b>	<b>Role</b>
<b>Initiator/Terminator</b>	Initiator Terminator
<b>Cohesive Hub</b>	Factory Composer (as Decider and Superior) Coordinator (as Decider and Superior) Sub-composer Sub-coordinator
<b>Atomic Hub</b>	Factory Coordinator Sub-coordinator
<b>Cohesive Superior</b>	Composer (as Superior only) Sub-Composer Coordinator (as Superior only) Sub-coordinator
<b>Atomic Superior</b>	Coordinator (as Superior only)) Sub-coordinator
<b>Participant</b>	Inferior Enroller



4414  
4415  
4416  
4417  
4418

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

4419  
4420  
4421  
4422  
4423  
4424  
4425  
4426  
4427  
4428

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)

4428 **Part 3. Appendices**

4429

4430

4431

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

4432

**A. Glossary**

4433

<b>Message</b>	A datum which is produced and then consumed.
<b>Sender</b>	The producer of a message.
<b>Receiver</b>	The consumer of a message.
<b>Transmission</b>	The passage of a message from a sender to a receiver.
<b>Endpoint</b>	A sender or receiver.
<b>Address</b>	An identifier for an endpoint.
<b>Carrier Protocol</b>	A protocol which defines how transmissions occur.
<b>Carrier Protocol Address (CPA)</b>	The address of an endpoint for a particular carrier protocol.
<b>Business Transaction Protocol Address (BTPA)</b>	A compound address consisting of a mandatory <i>carrier protocol address</i> and an optional opaque suffix. <div style="border: 1px solid black; padding: 2px; margin-top: 5px;"><i>PRF - suffix ? I've used "additional information"</i></div>
<b>Actor</b>	An entity which executes procedures, a software agent.
<b>Application</b>	An actor which uses the Business Transaction Protocol.
<b>Application Message</b>	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.
<del>Service</del>	<del>An actor which on receipt of an application messages may start an application operation which is appropriate. For example, a process which advertises an interface allowing defined RPCs to be invoked by a remote client.</del>
<u>Service</u>	<u>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.</u>
<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>	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

“The **Countereffect** will attempt so far as is possible to reverse or cancel the **Effect** such that an observer (on completion of the **Countereffect**) is unaware that the **Effect** ever occurred, but this attempt cannot be guaranteed to succeed”.

<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> <i>or</i> <b>Atom</b>	A set of participants (which may have only one member), all of which will receive instructions that will result in a homogeneous outcome. (Transitively, a set of operations, whose effect is capable of countereffect.) An atom is identified by an atom identifier.
<b>Atom Identifier</b>	A globally unique identifier assigned to an atom.
	<div style="border: 1px solid black; padding: 5px;"><i>PRF – abs msgs define as unambiguous in scope of its address-as-superior, I think.</i></div>
<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.

4434