

Business Transaction Protocol

An OASIS Committee Specification

CURRENT STATUS : committee draft for review

Version 1.0 [0.9.5.1]

DD Mmm 2002 [22 April 2002 18:25]

<i>Working Draft 0.9</i>	24 October 2001
<i>Working Draft 0.9.1 – includes all issues agreed 16 Jan 2002, and 82 (deferred)</i>	18 January 2002
<i>Working Draft 0.9.2 – all issues as agreed 13 February 2002</i>	13 February 2002
<i>Working Draft 0.9.2.1 – issues 2, 3, 15, 19, 50, 67, 95</i>	26 February 2002
<i>Working Draft 0.9.2.2 – as accepted 27 Feb 2002+ corrections, issues 29, 60, 97, 99</i>	12 March 2002
<i>Working Draft 0.9.2.3 – 0.9.2.2 and issue 106, 96, 98</i>	18 March 2002
<i>Working Draft 0.9.2.4 – as accepted 27 Mar 2002 + 61, 87, 100, 107, 108, 109, inclusion of model</i>	3 April 2002
<i>Review Draft 0.9.5 – review draft – all changes merged</i>	3 April 2002
<i>Review Draft 0.9.5.1 –revised soln for 87, 2nd solution for 108, additional diagrams for 66, formatting cleanup, inferior-handle, garbles.</i>	22 April 2002

[Change marks relative to 0.9.5](#)

9 Copyright and related notices

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

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

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

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

28

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

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

41

Acknowledgements

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

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

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

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

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

58 *In memory of Ed Felt*

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

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

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

65 on 11 September 2001.

66

Typographical and Linguistic Conventions and Style

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

Cancel
Participant
Application Message

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

Coordinator

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

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

BEGIN
CONTEXT
RESIGN

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

BEGIN/atom

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

BEGIN/atom & CONTEXT

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

ENROL + VOTE

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

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

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

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

93	Contents	
94	Copyright and related notices.....	2
95	Acknowledgements	3
96	Typographical and Linguistic Conventions and Style	5
97	Contents	6
98	Part 1. Purpose and Features of BTP	10
99	Introduction	10
100	Development and Maintenance of the Specification.....	11
101	Structure of this specification.....	12
102	Conceptual Model	13
103	Example Core	13
104	Business transactions.....	14
105	External Effects	15
106	Two-phase outcome	16
107	Actors and roles.....	17
108	Superior:Inferior relationship.....	17
109	Business transaction trees.....	18
110	Atoms and Cohesions	20
111	Participants, Sub-Coordinator and Sub-Composers.....	21
112	Business transaction creation	21
113	Business transaction propagation	23
114	Creation of Intermediates (Sub-Coordinator and Sub-Composers).....	24
115	“Checking” and context-reply	25
116	Message sequence	26
117	Control of inferiors	30
118	Evolution of confirm-set.....	33
119	Confirm-set of intermediates	37
120	Optimisations and variations	40
121	Spontaneous prepared.....	40
122	One-shot	41
123	Resignation.....	42
124	One-phase confirmation	43
125	Autonomous cancel, autonomous confirm and contradictions.....	43
126	Recovery and failure handling.....	44
127	Types of failure	44
128	Persistent information.....	45
129	Recovery messages.....	46
130	Redirection	47
131	Terminator:Decider failures and transaction timelimit	48
132	Contradictions and hazard	48
133	Relation of BTP to application and carrier protocols	50
134	Other elements	52
135	Identifiers.....	52
136	Addresses.....	53
137	Qualifiers.....	54
138	Part 2. Normative Specification of BTP	54
139	Actors, Roles and Relationships.....	54
140	Relationships.....	55

141	Roles	56
142	Roles involved in the outcome relationships	57
143	Superior	57
144	Inferior.....	58
145	Enroller.....	59
146	Participant.....	59
147	Sub-coordinator	60
148	Sub-composer.....	60
149	Roles involved in the control relationships.....	61
150	Decider	61
151	Coordinator.....	61
152	Composer.....	61
153	Terminator	62
154	Initiator	63
155	Factory.....	63
156	Other roles	64
157	Redirector	64
158	Status Requestor	64
159	Summary of relationships	65
160	Abstract Messages and Associated Contracts	66
161	Addresses.....	66
162	Request/response pairs.....	67
163	Compounding messages	68
164	Extensibility.....	70
165	Messages.....	70
166	Qualifiers.....	70
167	Messages not restricted to outcome or control relationships.....	71
168	CONTEXT	71
169	CONTEXT_REPLY.....	72
170	REQUEST_STATUS.....	73
171	STATUS.....	74
172	FAULT	76
173	REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES	78
174	Messages used in the outcome relationships	78
175	ENROL.....	78
176	ENROLLED.....	79
177	RESIGN.....	80
178	RESIGNED	81
179	PREPARE	82
180	PREPARED.....	82
181	CONFIRM.....	84
182	CONFIRMED	84
183	CANCEL.....	85
184	CANCELLED	86
185	CONFIRM_ONE_PHASE	87
186	HAZARD	88
187	CONTRADICTION	90
188	SUPERIOR_STATE	90
189	INFERIOR_STATE	92

190	REDIRECT	94
191	Messages used in control relationships.....	95
192	BEGIN.....	95
193	BEGUN	96
194	PREPARE_INFERIORS.....	97
195	CONFIRM_TRANSACTION.....	98
196	TRANSACTION_CONFIRMED	100
197	CANCEL_TRANSACTION.....	101
198	CANCEL_INFERIORS	102
199	TRANSACTION_CANCELLED	103
200	REQUEST_INFERIOR_STATUSES.....	104
201	INFERIOR_STATUSES.....	105
202	Groups – combinations of related messages.....	107
203	CONTEXT & application message	107
204	CONTEXT_REPLY & ENROL	108
205	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED	109
206	CONTEXT_REPLY & ENROL & application message (& PREPARED).....	109
207	BEGUN & CONTEXT	110
208	BEGIN & CONTEXT	110
209	Standard qualifiers	110
210	Transaction timelimit	110
211	Inferior timeout.....	111
212	Minimum inferior timeout.....	112
213	Inferior name	112
214	State Tables	113
215	Status queries.....	114
216	Decision events.....	114
217	Disruptions – failure events.....	115
218	Invalid cells and assumptions of the communication mechanism.....	115
219	Meaning of state table events	115
220	Persistent information.....	119
221	Superior state table	123
222	Inferior state table.....	127
223	Persistent information	132
224	XML representation of Message Set.....	132
225	Addresses.....	133
226	Qualifiers	133
227	Identifiers.....	134
228	Message References	134
229	Messages.....	134
230	CONTEXT	134
231	CONTEXT_REPLY.....	134
232	REQUEST_STATUS	135
233	STATUS	135
234	FAULT	135
235	ENROL.....	136
236	ENROLLED.....	137
237	RESIGN.....	137
238	RESIGNED	137

239	PREPARE	138
240	PREPARED.....	138
241	CONFIRM.....	138
242	CONFIRMED	138
243	CANCEL	139
244	CANCELLED	139
245	CONFIRM_ONE_PHASE	139
246	HAZARD	140
247	CONTRADICTION	140
248	SUPERIOR_STATE	140
249	INFERIOR_STATE	141
250	REDIRECT	141
251	BEGIN.....	141
252	BEGUN	142
253	PREPARE_INFERIORS	142
254	CONFIRM_TRANSACTION.....	142
255	TRANSACTION_CONFIRMED	143
256	CANCEL_TRANSACTION	143
257	CANCEL_INFERIORS	143
258	TRANSACTION_CANCELLED	144
259	REQUEST_INFERIOR_STATUSES	144
260	INFERIOR_STATUSES.....	144
261	Standard qualifiers	145
262	Transaction timelimit	145
263	Inferior timeout.....	145
264	Minimum inferior timeout.....	145
265	Inferior name	145
266	Compounding of Messages.....	145
267	XML Schemas	146
268	XML schema for BTP messages	146
269	XML schema for standard qualifiers	158
270	Carrier Protocol Bindings	159
271	Carrier Protocol Binding Proforma.....	159
272	Bindings for request/response carrier protocols	161
273	Request/response exploitation rules	161
274	SOAP Binding	163
275	Example scenario using SOAP binding	165
276	SOAP + Attachments Binding.....	166
277	Conformance	168
278	Part 3. Glossary	171
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 numerous software product companies (listed on page 3), grouped in the Business Transactions Technical Committee (BT TC) of OASIS.

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

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

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

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

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

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.

Development and Maintenance of the Specification

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

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

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

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

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

bt-spec@lists.oasis-open.org

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

The main list of the committee is:

business-transaction@lists.oasis-open.org

Structure of this specification

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

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

Part 1 contains

- Executive Summary
- This document structure description
- Conceptual Model

Part 2 contains the following sections:

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

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

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

374 **Conceptual Model**

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

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

386 **Example Core**

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

- 391 1. *Manufacturer A*'s production scheduling system sends an Order message to a
392 *Supplier*.
- 393 2. The *Supplier's* order processing system sends back an order confirmation with the
394 details of the order.
- 395 3. *Manufacturer A* orders delivery from a *Shipper* for the ordered parts.
- 396 4. The *Shipper* evaluates the request and based on its truck schedule it sends back a
397 positive or negative reply.
- 398 5. Some shipments need to be insured based on their value, where they are shipped
399 from, and method of transportation. *Manufacturer A* sends an Order message to an
400 *Insurer* when this is necessary.
- 401 6. The *Insurer* responds with a bid or a no-bid response.

402 Problems have arisen with some of these interactions.

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

- There were communications problems with supplier Z that resulted in an order not being confirmed. The shipper arrived to pick up the order and supplier Z knew nothing about it.
- Goods have been shipped without insurance when company policy dictated that insurance was required.

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

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

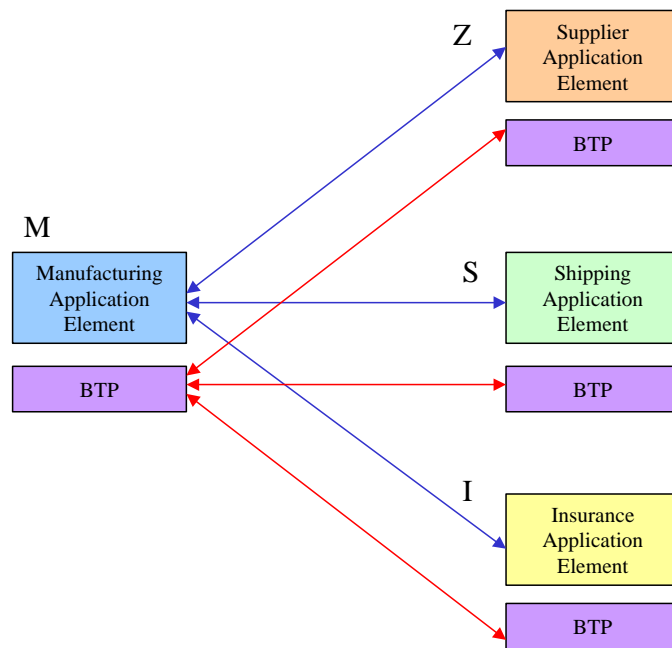


Figure 1 – Manufacturer Example

Business transactions

A **Business Transaction** can be defined as a consistent change in the state of a business relationship between two or more **parties**. A business relationship is any distributed state held by the parties which is subject to contractual constraints agreed by those parties. For example, an

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

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

External Effects

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

In general, to be able to satisfy such contracts a BTP-enabled **service** must support in some manner provisional or tentative state changes (the transaction’s **provisional effect**) and completion either through confirmation (**final effect**) or cancellation (**counter-effect**). The meaning of provisional, final, and counter-effect are specific to the application and to the implementation of the application. In the example, the reservation of the order is the provisional effect, the completion of the purchase is the final effect.

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

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

Table 1 Some alternatives for provisional, final and counter effects

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

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

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

Two-phase outcome

The BTP protocol coordinates the transitions into and out of the event states described above by sending messages between the transaction parties. This involves a two-phase exchange. First the application elements exchange messages that determine the characteristics and cause the performance of the provisional effect; then a separate message, to the BTP element, asking for the performance of the final or the counter effect.

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

- determines whether it is able achieve its provisional effect and then ensure it will be able either to cancel (counter-effect) its operation or to confirm (give final effect to) its operation, whichever is subsequently instructed, and

498 • reports its ability to confirm-or-cancel (its preparedness) to a central coordinating
499 entity.

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

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

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

509 **Actors and roles**

510 BTP centres on the bilateral relationship between the computer systems of the coordinating entity
511 and those of one of the parties in the overall business transaction. For each bilateral relationship
512 in a business transaction, a software agent within the coordinating entity’s systems plays the BTP
513 role of Superior and a software agent within the systems of the party play the BTP role of
514 Inferior. The concept “**role**” refers strictly to the participation in a particular relationship in a
515 particular business transaction. The software agent performing a role is termed an **Actor**. An
516 Actor is distinguished from other Actors by being distinguishably addressable. The same Actor
517 may perform multiple roles in the same business transaction (including the case where a Superior
518 is also an Inferior), and may also perform the same or different roles in multiple business
519 transactions, either concurrently or consecutively.

520 **Superior:Inferior relationship**

521 A basic case of a single Superior:Inferior relationship, including the association with application
522 elements, is illustrated in Figure 2. In many cases, including the manufacturer supply example,
523 the application element associated with the superior will directly initiate the application
524 exchanges –as does the manufacturer’s application client to the supplier’s server, for example –
525 but this is not invariably the case. It is possible that the first direct communication between the
526 application elements is from one associated with an inferior to the one associated with the
527 superior – for example, with an application that requested quotes by advertising the identity and
528 location of the Superior along with invitation to quote; incoming quotes would be the first direct
529 application message exchanged. In all cases the topmost application element in a tree or subtree
530 will be aware of the business transaction first. How the identity of the transaction and the address
531 of the BTP Superior are communicated to the secondary application element is a matter for the
532 application protocol and not strictly part of BTP, although it will commonly be done by
533 associating a BTP CONTEXT message with application messages..

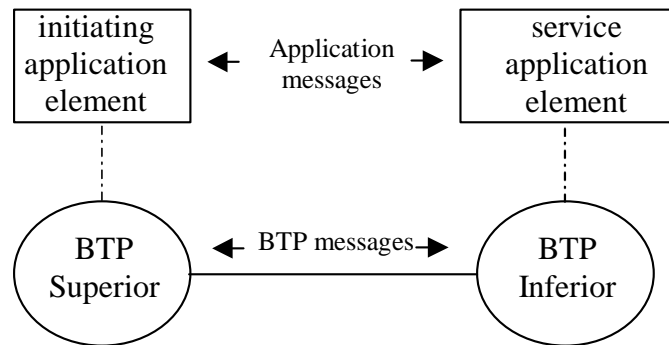


Figure 2 Basic Superior:Inferior relationship for BTP

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

Business transaction trees

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

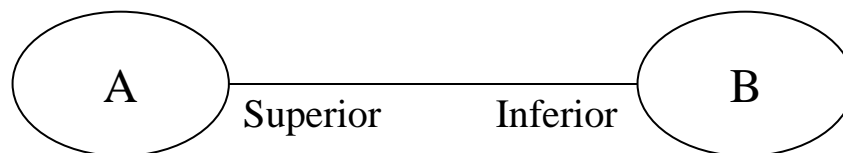


Figure 3 Simple two-party business transaction

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

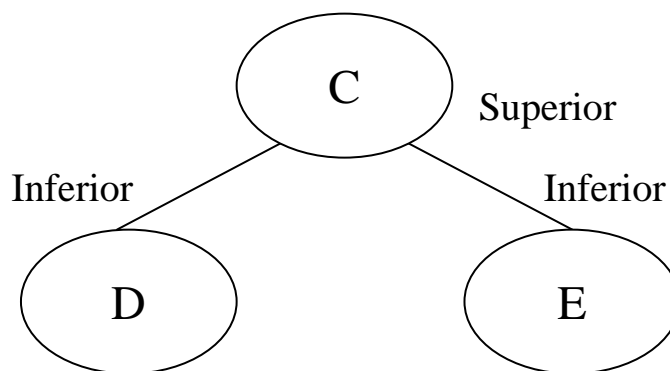


Figure 4 Business transaction with two inferiors

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

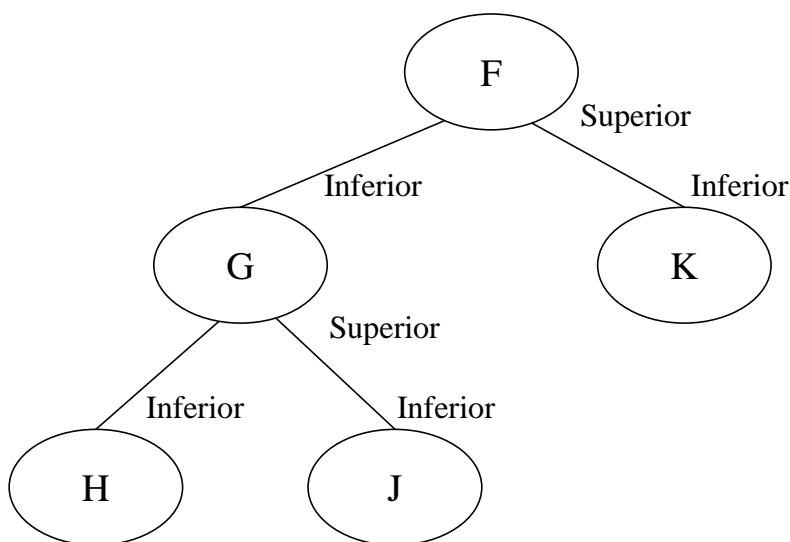


Figure 5 Business transaction with an Intermediate (interposition)

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

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

Atoms and Cohesions

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

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

- a) the application element initiated the business transaction (causing the creation of the Superior), and instructed that all Inferiors of the Superior should confirm or all should cancel; the Superior is an **Atom Coordinator**;
- b) the application element initiated the business transaction, but deferred the choice of which Inferiors should confirm until later, allowing it (the application element) to choose some subset to be confirmed, others to cancel; the Superior is a **Cohesion Composer**;
- c) the application element was itself involved in an existing business transaction, and the Superior in this relationship is the Inferior in another one; this application element instructed that all Inferiors of this Superior should confirm, but only if confirmation is instructed from above or all should cancel; the Superior is an (atomic) **Sub-coordinator**;
- d) the application element was itself involved in an existing business transaction, and the Superior in this relationship is the Inferior in another one; this application element deferred the choice of which Inferiors should be candidates to confirm until later, allowing it (the application element) to choose some subset to be confirmed, given that confirmation is instructed from above, others to cancel; the Superior is a (cohesive) **Sub-composer**.

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

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

623 For a cohesion, the set of Inferiors that eventually confirm is called the **confirm-set**. The term is
624 also used to mean the set of Inferiors that have been chosen to (potentially) confirm before the
625 final outcome is decided – if the cohesion is eventually cancelled, then confirm-set cancels. (See
626 section “Evolution of confirm-set”). The confirm-set of an Atom is all of the Inferiors.

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

632 If the Superior is not an Inferior, the determination of when, if and, for a Cohesion, what it should
633 confirm depends on the controlling application. This “top-most” Superior has a different
634 relationship to the controlling application to that of an Inferior to its Superior: an Inferior reports
635 that it is prepared to the Superior, which instructs it whether to cancel or to confirm; the top-most
636 Superior is asked by the application element to attempt to confirm, but, dependent on the
637 preparedness of its Inferiors, the top-most Superior makes the final decision. Consequently the
638 top-most Superior is termed the **Decider**; the application element that asks it to confirm is the
639 **Terminator**.

640 **Participants, Sub-Coordinators and Sub-Composers**

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

652 **Business transaction creation**

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

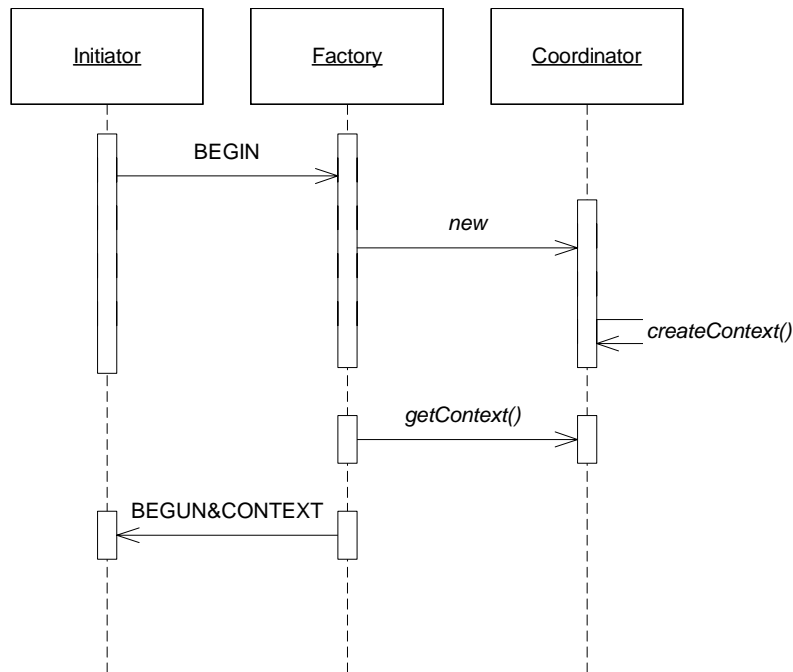


Figure 6 – Creation of a business transaction

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

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

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

Business transaction propagation

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

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

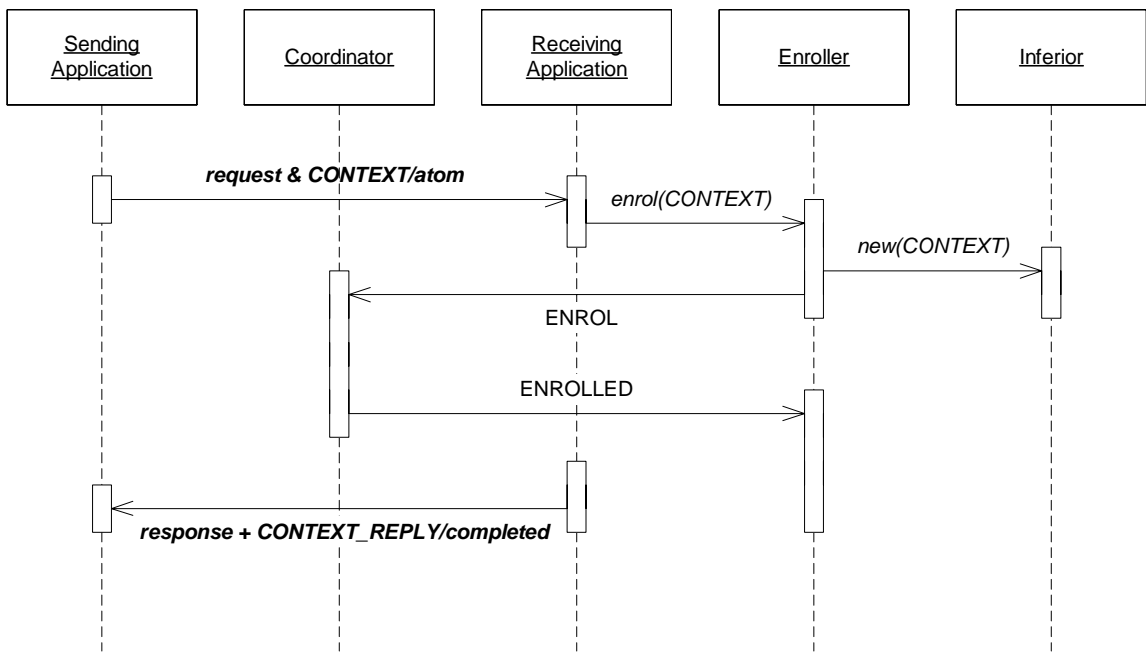


Figure 7 Sequence diagram of propagation

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

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

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

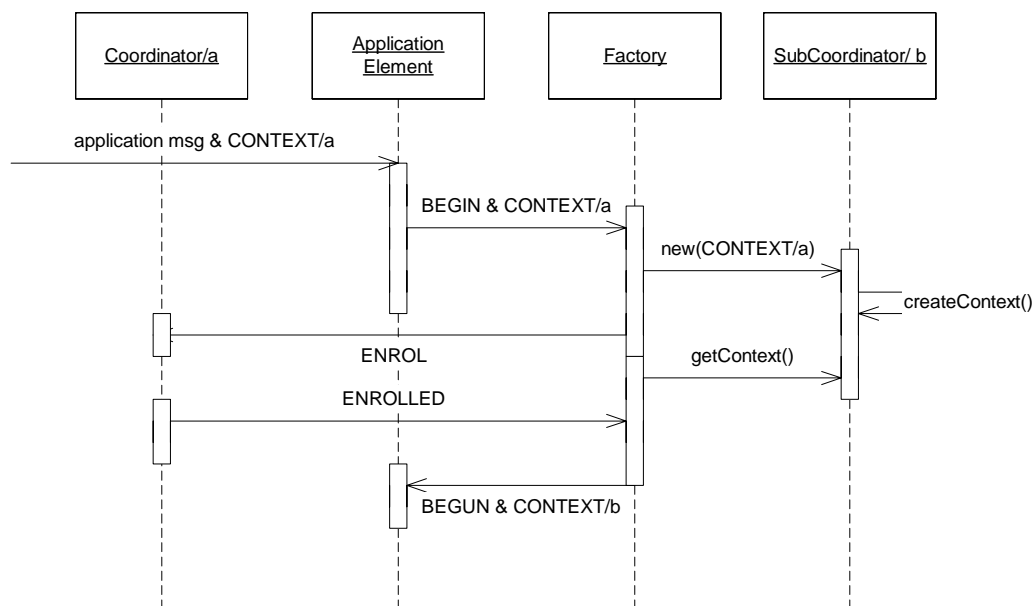


Figure 8 – Creation of a Sub-coordinator

The creation of a new Inferior and establishment of a Superior:Inferior relationship does not always imply that the BTP Actors are under the control of different business parties or application elements. In particular, an application element may begin a Cohesion, then create and enrol (atomic) Sub-coordinators as Inferiors of the Composer, then associate a different Sub-coordinator's CONTEXT with each of several aspects of the application work, transmitting that CONTEXT with the application messages for that aspect to the other parties in the business transaction. Those parties can then create Participants (or other Inferiors) that are enrolled with

the appropriate Sub-coordinator. Later, the application element (as Terminator, or its equivalent) can choose which of the Cohesion Composers' Inferiors to cancel and which to confirm. By interposing its own atomic Sub-coordinator the initiating application element can indicate to the other parties that some associated set of application work will be confirmed or cancelled as a unit. This may allow the receiving parties to share information between application operations and to make one Participant responsible for applying the outcome to several operations.

"Checking" and context-reply

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

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

CONTEXT_REPLY can also indicate that attempted enrollments failed. This can occur if the Enroller is unable to contact the Superior, but it able to return a CONTEXT_REPLY to wherever the CONTEXT came from.

~~Section explaining becoming prepared?~~

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

769 Message sequence

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

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

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

780 Figure 9 shows the message exchange for the conventional progression of a simple transaction to
781 confirmation with a single Superior:Inferior relationship, assuming the standard control
782 relationship. Two application elements using a request/response application message exchange
783 are involved – the first is represented as the Initiator and Terminator, the second as the Service
784 and Enroller. The Decider/Superior is shown as a Coordinator, but with only one Inferior there
785 would be no difference with a cohesion Composer. The Factory:Coordinator events are non-
786 standardised, but represent interactions that must occur in some form. There are other interactions
787 between the various application groups – Initiator-Terminator and Participant-Enroller-Service
788 that are not shown – in particular the Service:Participant relationship.

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

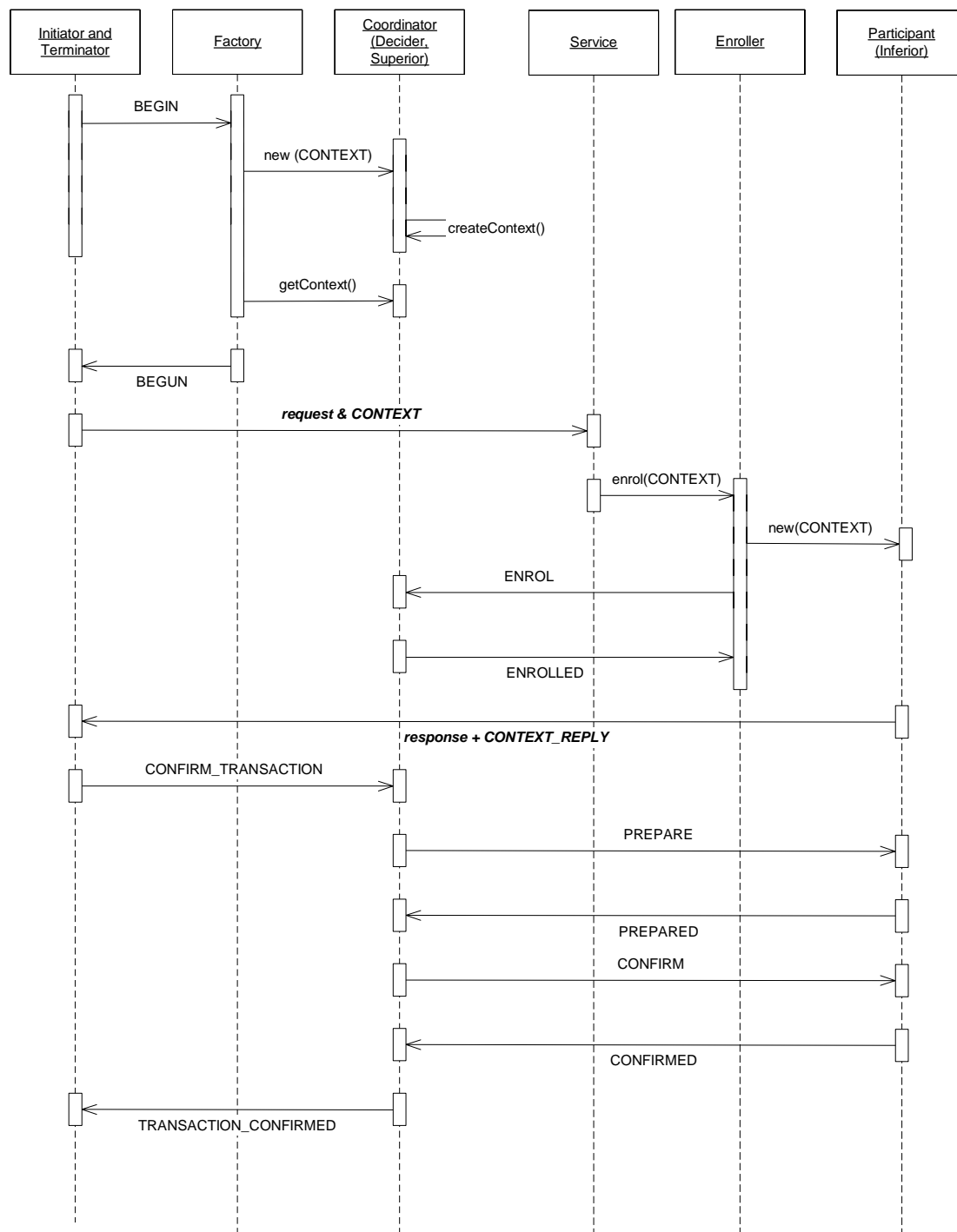


Figure 9 A conventional message sequence for a simple transaction

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

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

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

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

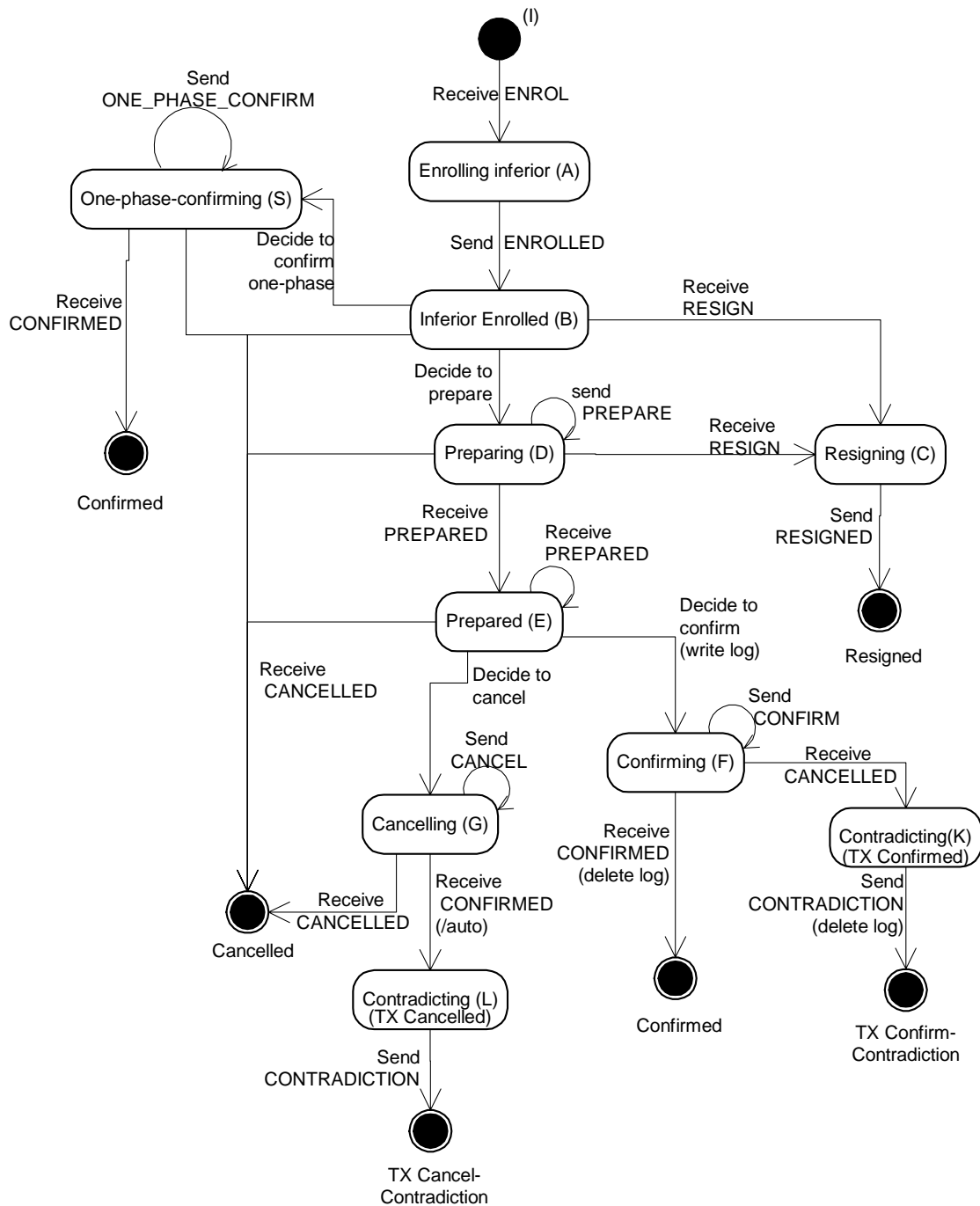


Figure 10 State diagram for Superior side of a Superior:Inferior relationship

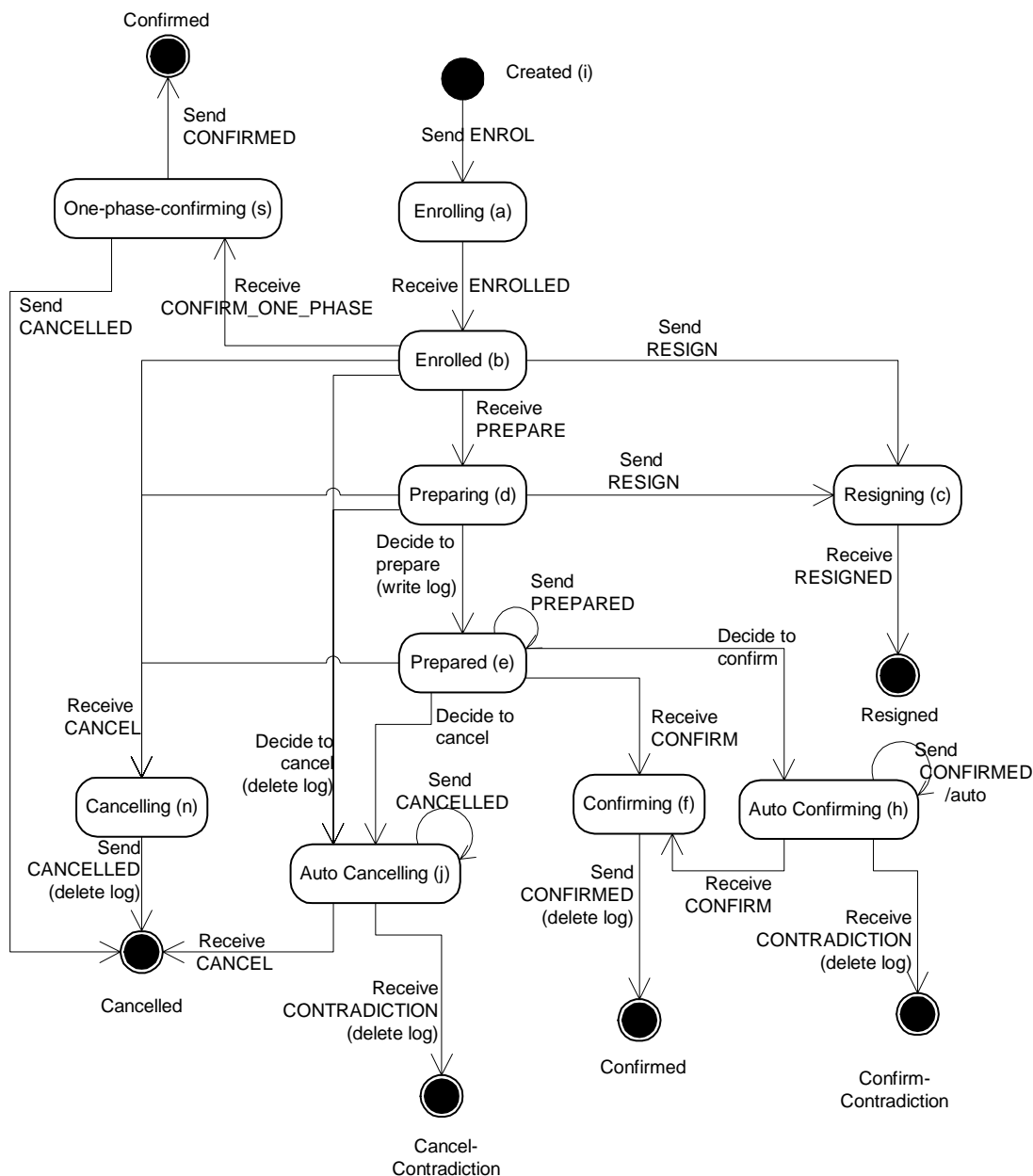


Figure 11 State diagram for Inferior side of Superior:Inferior relationship

Control of inferiors

In the case as shown in Figure 12, where the CONTEXT has been propagated from one application element (A) to others (B, C, and from C to D,E), the determination of whether to create and enrol Inferiors is, in general, up to the receiving application element – this is an aspect of the fundamental autonomy of the parties involved in a business transaction. This autonomy may be constrained in particular situations, by inter-party agreement or where the application elements are in fact under common control.

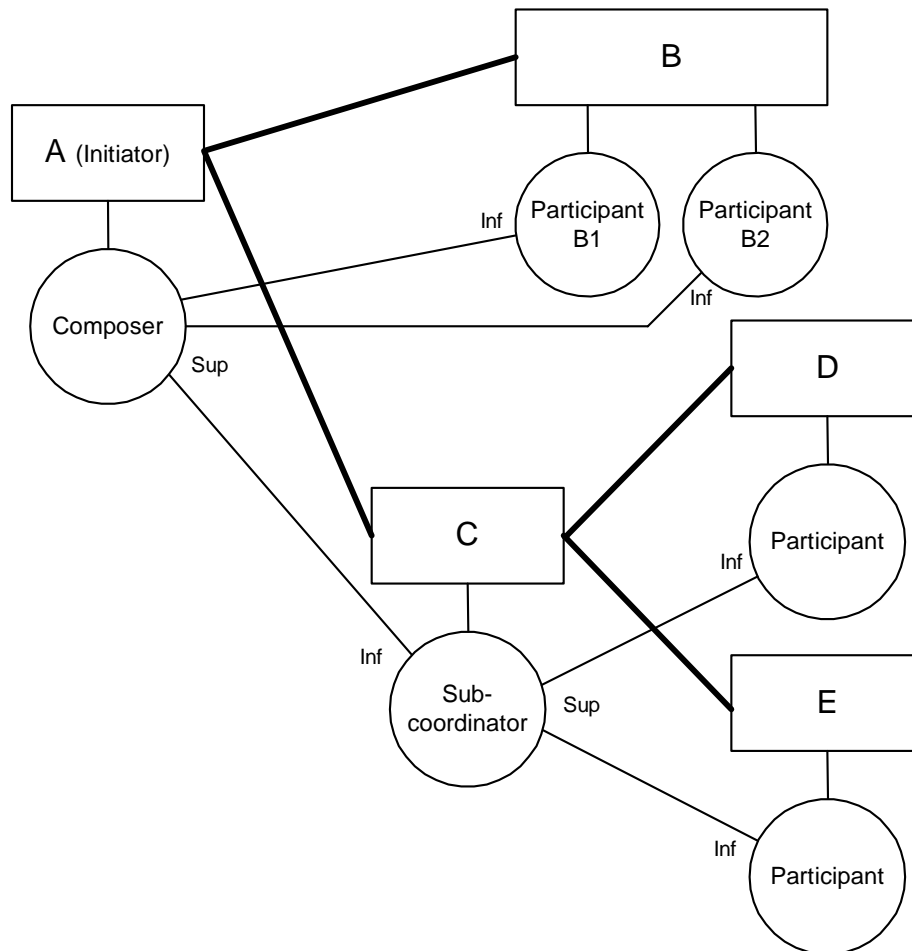


Figure 12 Transaction tree showing various application:Participant relationships

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

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

857 confirm-set for the Cohesion. For an Atom Coordinator, visibility of the Inferiors is useful but
858 less important, since no selection can be made among which will be in the confirm-set – for an
859 Atom, all Inferiors are ipso facto members of the confirm-set.

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

- 863 a) The application element can create an Atom Sub-coordinator as an immediate
864 Inferior of the Cohesion Composer and propagate the Sub-coordinator's CONTEXT
865 associated with application messages concerned with the particular part of the
866 application work; any Inferiors (however many there may be) enrolled with Sub-
867 coordinator can be assumed to be responsible for (some of) that part of the
868 application, and the Terminator application element can just deal with the immediate
869 Inferior of the Composer that it created.
- 870 b) The application element can propagate the Composer's own CONTEXT, and the
871 receiving application element can create its own Inferior (or Inferiors) which will be
872 responsible for some part of the application, and send ENROL(s) to the Composer (as
873 Superior). Application messages concerned with that part of the application are
874 associated, directly or indirectly, with ~~each~~ ENROL, and the Terminator
875 application element can thus determine what ~~each~~ Inferior is responsible for.

876 In both cases, the means by which the application message and the BTP CONTEXT or ENROL
877 are associated ~~is~~ are ultimately application-specific, and there are several ways this can be done.

- 878 • -At the abstract message level, BTP defines the concept of transmitting “related” BTP
879 and application messages – particular bindings to carrier protocols can specify
880 interoperable ways to represent this relatedness (e.g. the BTP message can be in a
881 “header” field of the carrier protocol, the application message in the body).
- 882 • An application message may contain fields that identify or point to the BTP message (e.g.
883 the “inferior-identifier” from the ENROL may be a field of the application message).
- 884 • -BTP messages, including CONTEXT and ENROL, can ~~also~~ carry “qualifiers” –
885 extension fields that are not core parts of BTP or are not defined by BTP at all. The
886 standard qualifier “inferior-name” or application-specific qualifiers can be used to
887 associate application information and the BTP message. The qualifiers received from the
888 Inferiors on ENROL are visible to the Terminator application on the
889 INFERIOR_STATUSES message. The application design will need to ensure that the ;
890 ~~allowing a~~ Terminator ~~to~~ can determine which parts of the application work are associated
891 with each Inferior.

892 NOTE -- For example, a service receiving an invocation associated with a cohesion
893 CONTEXT, but where the application design meant that there would be no more
894 than one Inferior enrolled as a result of that invocation, could be required to include
895 information identifying the service and the invocation in the “inferior-name”
896 qualifier on the consequent ENROL. These qualifiers would be visible to the
897 Terminator on INFERIOR_STATUSES, allowing the Terminator to determine which
898 “inferior-identifiers” to include in the “inferiors-list” parameter of the

899 *CONFIRM_TRANSACTION which defines which Inferiors are to be confirmed.*
900 *Among other alternatives, the “inferior-identifier” itself could be a field of the*
901 *application response – this would also be applicable where there could be multiple*
902 *Inferiors enrolled as a consequence of one invocation for the Terminator to choose*
903 *between.*

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

906 **Evolution of confirm-set**

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

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

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

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

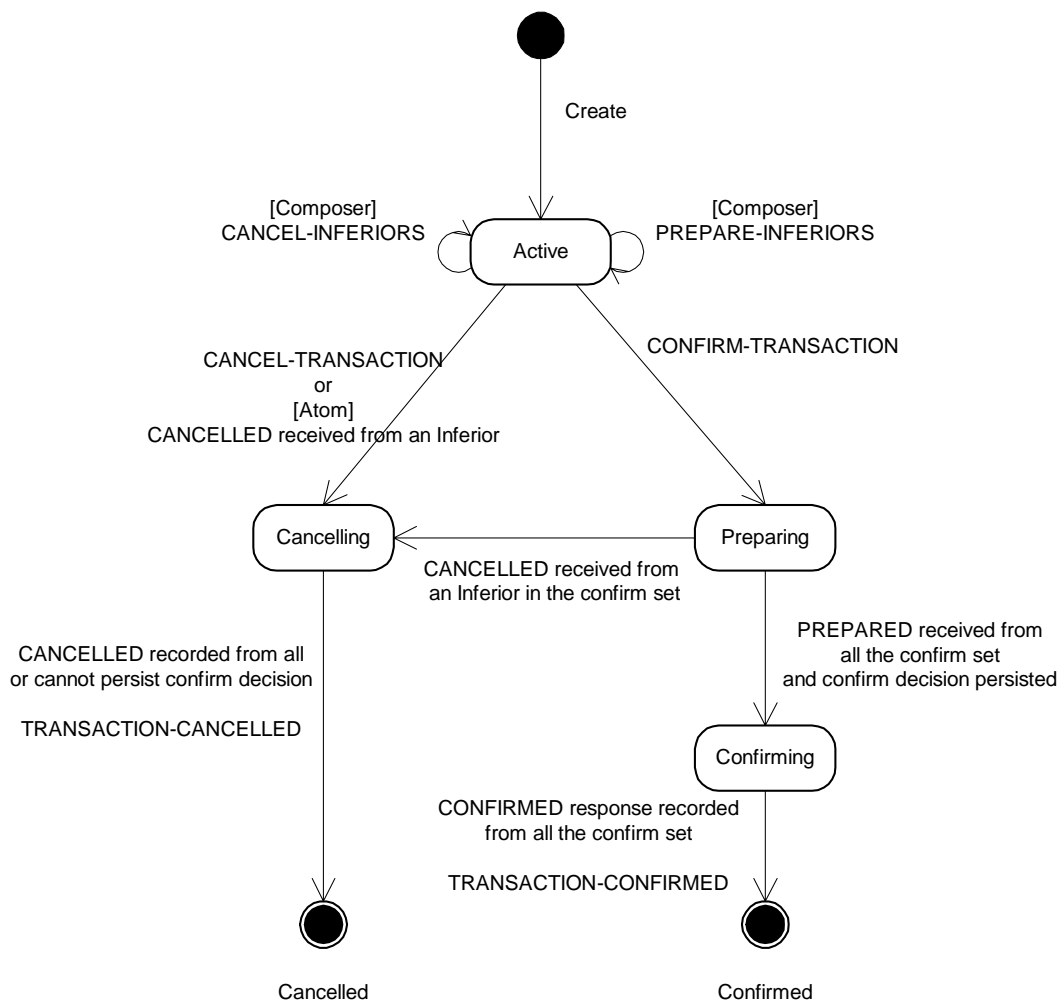


Figure 13 State diagram for a Composer or Coordinator (i.e. Decider)

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

- Inferiors are enrolled – ENROL is received by the Composer – adding to the set of Inferiors of the Composer.
- Inferiors may resign - RESIGN is received from an Inferior (see section Resignation below). The Inferior is immediately removed from the set of Inferiors, as if it had never been enrolled (a RESIGNED message may be sent to the Inferior, but it no longer “counts” in any of the Composer-wide considerations here).
- CANCELLED may be received from an Inferior; there is no required immediate effect, but if this is a Coordinator the Atom will certainly cancel eventually (and an implementation may choose to initiate cancellation immediately).
- PREPARED may be received; there is no immediate effect

- 936 • The Terminator may issue `PREPARE_INFERIORS` to the Composer (as Decider)
937 for some subset of the Inferiors; `PREPARE` is sent to each and any of the Inferiors
938 in the subset, excluding any from `RESIGN`, `CANCELLED` or `PREPARED` has been
939 received; the sending of `PREPARE` will induce the Inferiors to reply with
940 `PREPARED`, `CANCELLED` or `RESIGN`; when replies have been received from all,
941 the Composer (as Decider) replies to the Terminator with `INFERIOR_STATUSES`,
942 reporting the replies received (which may in fact have been received before the
943 `PREPARE_INFERIORS`). `PREPARE_INFERIORS` is not issued to Atom
944 Coordinators.

 - 945 • The Terminator may issue `CANCEL_INFERIORS` to the Composer (as Decider) for
946 some subset of the Inferiors; `CANCEL` is sent to each and any of the Inferiors in the
947 subset, excluding any from `RESIGN` or `CANCELLED` has been received; the
948 sending of `CANCEL` will normally induce the Inferiors to reply with `CANCELLED`
949 – there are some exception cases; when replies have been received from all, the
950 Composer (as Decider) replies to the Terminator with `INFERIOR_STATUSES`,
951 reporting the replies received. `CANCEL_INFERIORS` is not issued to Atom
952 Coordinators. `CANCEL_INFERIORS` may be issued for an Inferior regardless of
953 whether `PREPARED` has been received from it.

 - 954 • The Terminator may issue `REQUEST_INFERIOR_STATUSES` to the Composer
955 (as Decider) for all or some subset of the Inferiors; the Composer immediately
956 replies with `INFERIOR_STATUSES`, reporting the current state of the Inferiors as
957 known to the Superior.
- 958 Eventually, the Terminator issues one of the completion messages – `CANCEL_TRANSACTION`
959 or `CONFIRM_TRANSACTION`. These messages have a flag that determines whether the
960 Terminator wishes to be informed of contradictory and heuristic decisions or hazards within the
961 transaction – this affects when the reply from the Composer (as Decider) is sent to the
962 Terminator. (See section “Autonomous cancel, autonomous confirm and contradictions” for
963 details on contradictory and heuristic cases).
- 964 If the message is `CANCEL_TRANSACTION`, `CANCEL` is sent to all Inferiors that it has not
965 already been sent to, and from which neither `RESIGN` or `CANCELLED` have been received. If
966 the Terminator indicates it does not want to be informed of contradictions, the Composer will
967 immediately reply with `TRANSACTION_CANCELLED`. Otherwise, if and when `CANCELLED`
968 or `RESIGN` has been received from all Inferiors, the Composer replies to the Terminator with
969 `TRANSACTION_CANCELLED`; but if `HAZARD` or `CONFIRMED` is received from any
970 Inferior, the reply is `INFERIOR_STATUSES`, identifying which Inferior(s) had problems.
- 971 If the completion message is `CONFIRM_TRANSACTION`, the `inferiors-list` parameter of the
972 message defines the confirm-set. If the parameter is absent (which it must be for an atom
973 Coordinator), then all Inferiors (excluding only those that have resigned) are the confirm-set;
974 otherwise the confirm-set is only the Inferiors identified in the `inferiors-list` parameter (less any
975 from which `RESIGN` has been received). The processing to arrive at the confirm decision is:
- 976 • If at the point of receiving `CONFIRM_TRANSACTION` or at any point before making
977 the confirm decision (see below), `CANCELLED` is received, then the transaction is
978 cancelled and processing continues as if `CANCEL_TRANSACTION` had been received.

- 979 • If there any Inferiors **not** in the confirm-set from which neither CANCELLED or
980 RESIGN has been received, CANCEL is sent to them (this cannot happen for Atom
981 Coordinators)
- 982 • If initially or later, there is exactly one Inferior in the confirm-set, and either PREPARE
983 has not been sent to it, or PREPARED has been received from it, then at implementation
984 or configuration option, CONFIRM_ONE_PHASE can be sent to that Inferior. This
985 delegates the confirm decision to the Inferior
- 986 • If at any point, RESIGN is received from an Inferior, it is immediately removed from
987 the confirm-set (this may trigger the decision making)
- 988 • If there are any Inferiors in the confirm-set from which none of PREPARED,
989 CANCELLED has been received and to which PREPARE has not yet been sent,
990 PREPARE is sent to that Inferior
- 991 • If initially or later, PREPARED has been received from all Inferiors in the confirm-set,
992 the Composer *makes the confirm decision*; it persists (or attempts to persist) information
993 identifying the Inferiors in the confirm-set; if this fails, the transaction is cancelled and
994 processing continues as if CANCEL_TRANSACTION had been received; if the
995 information is persisted, the confirm decision has been made.

996 When the confirm decision is made, CONFIRM is sent to all the Inferiors in the confirm-set. And,
997 if on the CONFIRM_TRANSACTION the Terminator indicated it did not wish to be informed of
998 contradictions, TRANSACTION_CONFIRMED is sent to the Terminator.

999 If the Terminator indicated it wanted to be informed of contradictions, the Composer replies to it
1000 with TRANSACTION_CONFIRMED if and when CONFIRMED has been received from all the
1001 Inferiors in the confirm-set and CANCELLED or RESIGN has been received from any other
1002 Inferiors. If other replies (CANCELLED from a confirm-set Inferior, CONFIRMED from other
1003 Inferiors, HAZARD from any) are received, the reply to the Terminator is
1004 INFERIOR_STATUSES, identifying which Inferior(s) had problems.

1005 [Figure 14 shows an example message sequence for a Composer with three Inferiors. The](#)
1006 [Terminator \(application element\) chooses to prepare Inferiors 1 and 3 explicitly – the numbers in](#)
1007 [parentheses on the Terminator:Composer messages represent the inferior-identifiers in the](#)
1008 [“inferior-list” parameters. Both 1 and 3 prepare successfully, but the Terminator then decides to](#)
1009 [make 1 and 2 the confirm-set; that is, if the transaction confirms only 1 and 2 are confirmed. The](#)
1010 [Terminator issues CONFIRM_TRANSACTION to the Composer. A PREPARED message has](#)
1011 [not been received from Inferior 2 yet, so the Composer issues PREPARE to it, and waits for the](#)
1012 [PREPARED. At the same time, it sends CANCEL to Inferior 3, which has been excluded from](#)
1013 [the confirm-set by the CONFIRM_TRANSACTION. After the PREPARED is received from](#)
1014 [Inferior 2, the Composer makes the confirm decision and issues CONFIRM to the Inferiors, and](#)
1015 [waits for the CONFIRMED messages before reporting to the Terminator. The](#)
1016 [CONFIRM_TRANSACTION in this case did not ask for reporting of hazards \(see below\) – if it](#)
1017 [had not, the TRANSACTION_CONFIRMED would have been sent at the same time as the](#)
1018 [CONFIRM messages.](#)

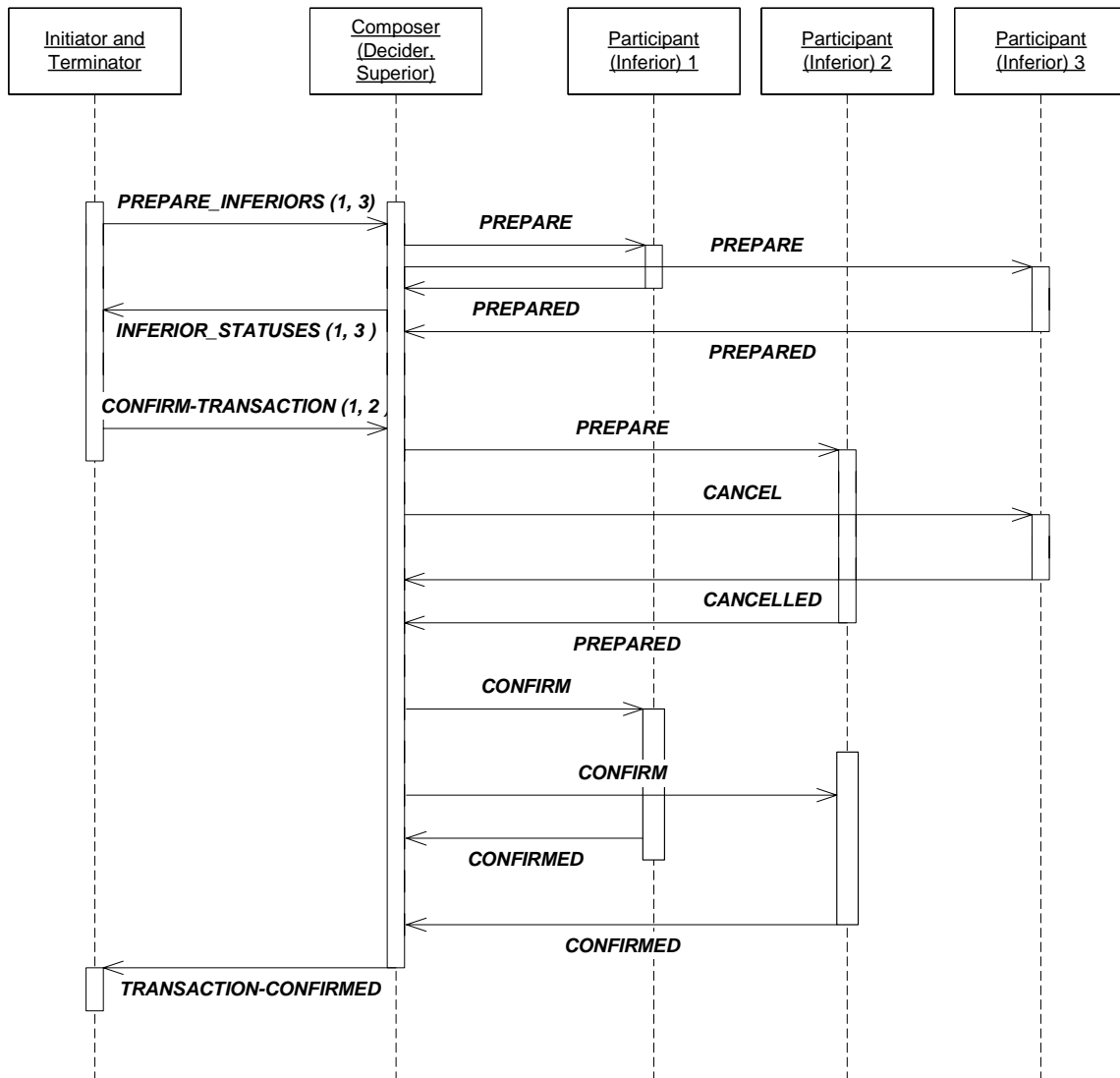


Figure 14 Termination sequence for a composer

Confirm-set of intermediates

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

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

Error! Reference source not found. [is a state diagram for a Sub-composer or Sub-coordinator.](#)

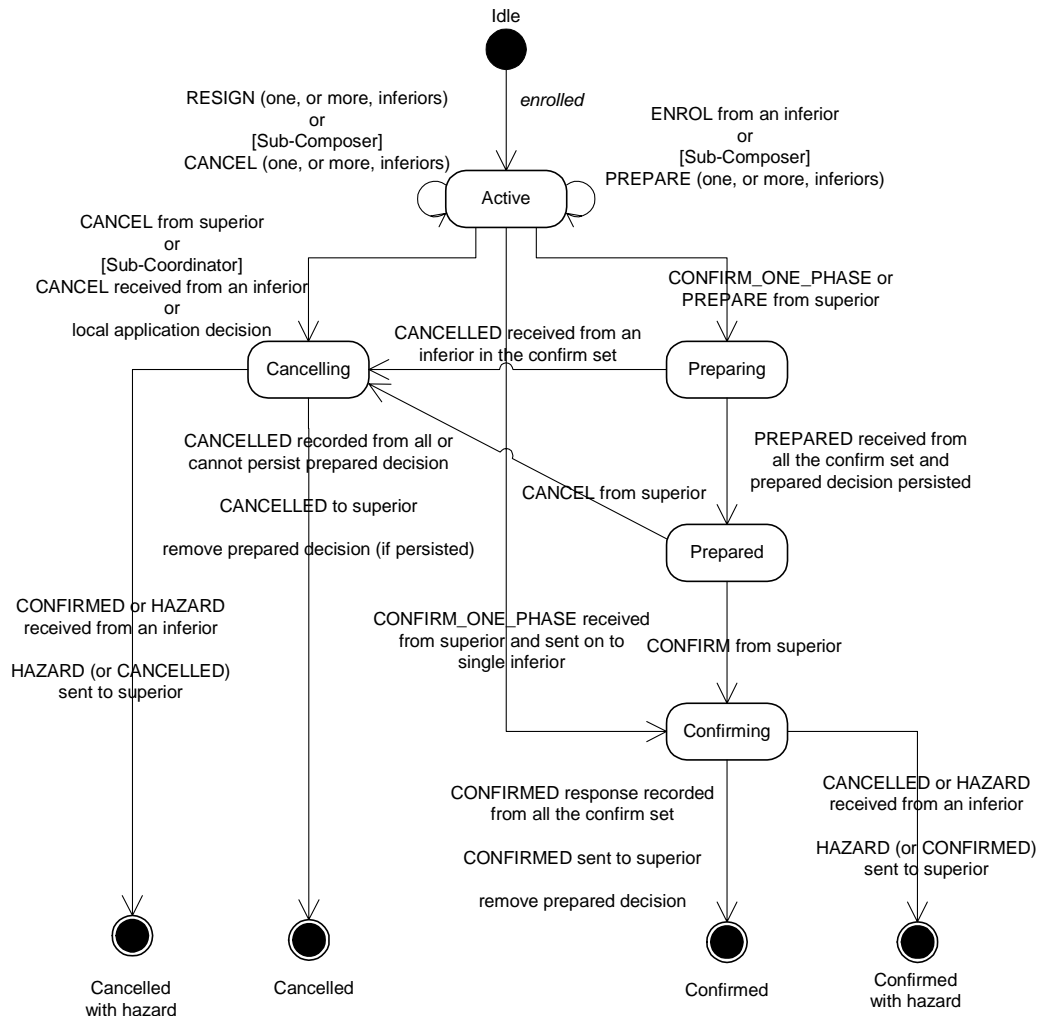


Figure 15 State diagram for Sub-coordinator or Sub-composer

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

- ENROL messages can be received, adding a new Inferior
- Inferiors may resign - RESIGN is received from an Inferior. The Inferior is immediately removed from the set of Inferiors
- CANCELLED may be received from an Inferior
- PREPARED may be received from an Inferior

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

1044 sending of messages. This is especially the case if the Intermediate knows (from application
 1045 knowledge, perhaps involving received or sent CONTEXT_REPLY messages) that there will be
 1046 no further enrolments. In particular:

- 1047 • If CANCELLED is received from an Inferior, and this is a Sub-coordinator, the Sub-
 1048 coordinator can itself cancel - CANCEL is sent to other Inferiors, and CANCELLED to
 1049 the Superior
- 1050 • If RESIGN is received from the only Inferior and there will be no other enrolments, the
 1051 Intermediate can itself resign, sending RESIGN to the Superior
- 1052 • If PREPARED is received from the Superior, it is known there will be no other
 1053 enrolments and this is a Sub-coordinator, the Sub-coordinator can become prepared
 1054 (assuming successful persistence of the appropriate information) and send PREPARED
 1055 to the Superior.

1056 For a Sub-composer, application logic will invariably be involved in determining what effect a
 1057 CANCELLED and PREPARED from an Inferior have – though in a real implementation, this
 1058 logic may be delegated to the BTP-support software.

1059 The Intermediate may initiate cancellation or the two-phase outcome exchange, either as a result
 1060 of receiving the corresponding message (CANCEL, PREPARE) from the Superior, or triggered
 1061 by its own controlling application element. For a Sub-composer, this may be partial - a Sub-
 1062 composer might be instructed by the application element to cancel some Inferiors and send
 1063 PREPARE to others. Receipt of PREPARE from the Superior will often have a similar effect to a
 1064 Decider receiving CONFIRM_TRANSACTION – PREPARE is propagated to all Inferiors that
 1065 have not indicated they are PREPARED. However, exactly what happens on receiving PREPARE
 1066 will depend on the application – receipt of the PREPARE may be visible to the application
 1067 element and cause it to initiate further application activity (perhaps causing enrolment of new
 1068 Inferiors) before it is determined whether to propagate PREPARE, and with a Sub-composer,
 1069 some of the Inferiors may be instructed to cancel instead.

1070 Assuming the Intermediate does not cancel as a whole (in which case CANCEL would be sent to
 1071 all Inferiors), the Intermediate will at some point attempt to become prepared. If it is a Sub-
 1072 coordinator, this will require that PREPARED has been received from all Inferiors. For a Sub-
 1073 composer, application logic will determine from which Inferiors PREPARED is required, with
 1074 the others being cancelled. In either case, the Intermediate will persist the information about the
 1075 Inferiors that are to be in the confirm-set and about the Superior, if this persisting is successful,
 1076 send PREPARED to its own Superior.

1077 If CANCEL is subsequently received from the Superior, this is propagated to all the Inferiors and
 1078 the persistent information removed (or effectively removed as far as recovery is concerned). It is
 1079 not important which order this is done in, since the recovery sequence will ensure that a cancel
 1080 outcome is eventually delivered anyway.

1081 If CONFIRM is received from the Superior (which can only be after sending PREPARED to the
 1082 Superior), this is likewise propagated to the Inferiors. For a Sub-coordinator, CONFIRM is
 1083 invariably sent to all Inferiors. However, for a Sub-composer it is possible further application
 1084 logic intervenes and some of the Inferiors are rejected from the confirm-set at this late stage.

1085 (This can only occur when the application work, as defined by the contract to the Superior, can be
1086 performed by some sub-set of the Inferiors.) The Intermediate may, but is not required to, change
1087 the persistent information to reflect the confirm outcome (though a Sub-composer that selects
1088 only some Inferiors probably will need to re-write the information to ensure the correct subset are
1089 confirmed despite possible failures). If the information is not changed, then, on recovery, the
1090 Intermediate will find itself to be in a prepared state and will interrogate the Superior to re-
1091 determine the outcome. If the information is changed, a recovered Intermediate can immediately
1092 continue with ordering confirmation to its Inferiors.

1093 If CONFIRM_ONE_PHASE is received from the Superior, either before or after the Intermediate
1094 has become PREPARED, the effect is very similar to a Decider receiving
1095 CONFIRM_TRANSACTION. If there is only one Inferior, the CONFIRM_ONE_PHASE may
1096 be propagated to that Inferior. Otherwise, the Intermediate behaves as a Decider, making a
1097 confirm decision if it can.

1098 If one or more Inferiors make contradictory autonomous decisions, or HAZARD is received from
1099 an Inferior, the Intermediate may report this to the Superior using HAZARD. However, BTP does
1100 not require this. Since the Superior may be owned and controlled by a different organisation,
1101 there may be business reasons not to report such problems.

1102 **Optimisations and variations**

1103 **Spontaneous prepared**

1104 As described above, before a Superior can order confirmation to an Inferior, the Inferior must
1105 become “prepared”, meaning that it is ready to confirm or to cancel as it so ordered and send the
1106 PREPARED message as a report of this. In the conventional message sequence, as shown above,
1107 the Inferior attempts to become prepared when it receives a PREPARE message from the
1108 Superior. The PREPARE in turn is sent by the Superior when it receives an appropriate request
1109 from its controlling application (or from its own Superior, if there is one). The application
1110 controlling the Superior will request the sending of PREPARE when it determines that no further
1111 application work associated with this Inferior (or, perhaps with the whole business transaction)
1112 will occur.

1113 However, for some applications, the application element controlling the Inferior will know that
1114 the application work for which the Inferior will be responsible is complete before a PREPARE is
1115 sent from the Superior. In fact, because the application element has autonomy in determining how
1116 application work is to be allocated to Inferiors, it is possible for the Inferior-side application
1117 element to know the work is complete **for a particular Inferior** when Superior-side application
1118 element will be sending more message to the Inferior-side. (The future work will, probably,
1119 require the enrollment of additional Inferiors.)

1120 BTP consequently allows the application element controlling an Inferior to cause the Inferior to
1121 become prepared, and to send PREPARED to the Superior without PREPARE having been
1122 received from the Superior. From the perspective of the BTP Superior the Inferior sends
1123 PREPARED spontaneously. Apart from this, a spontaneous PREPARED message is the same as,
1124 and has the same effect and implications as one induced by a PREPARE message.

1125 One-shot

1126 In the “conventional” message sequence shown above and assuming the Initiator, Terminator and
1127 Coordinator on the one side, and “Service”, Enroller and Participant on the other are located
1128 within their respective parties, there are eight messages passed in one direction or the other
1129 between the two parties. There are four round-trip exchanges: the application request and
1130 response exchange, the ENROL/ENROLLED exchange (going in the opposite direction and
1131 overlapped with the application exchange), then PREPARE/PREPARED and the
1132 CONFIRM/CONFIRMED. However, if the application exchange is a single request/response, it
1133 is possible to reduce these eight to two round-trips– the first of which merges the first three of the
1134 conventional sequence. The fundamental two-phase nature of BTP (or any coordination
1135 mechanism) means there have to be at least two round trips – one before the confirm-or-cancel
1136 decision is made at the Superior, one after. This merging of the exchanges is termed “one-shot”,
1137 as it requires only one exchange to take the relationship from non-existent to waiting for the
1138 confirm-or-cancel decision.

1139 **Error! Reference source not found.** shows a typical “one-shot” message sequence. The diagram
1140 distinguishes an additional aspect of the application elements, labelled “context-handler”. This is
1141 not a role in the BTP model, but is used only to distinguish a set of responsibilities and actions. In
1142 a real implementation these might be performed by the user application itself, or might be
1143 performed by the BTP-supporting infrastructure on the path between the application elements.
1144 (Figure 9 could be redrawn to show the context-handlers, but to no particular benefit) As in the
1145 conventional case, the CONTEXT is sent related to the application request (the creation of the
1146 CONTEXT by the Factory is not shown and is the same as the conventional case). The “context-
1147 handler” is aware of the sending of the CONTEXT.

1148 On the responder (service side), however, when the application element creates the Inferior, the
1149 ENROL is not sent immediately, but retained. The application performs the “provisional effect”
1150 implied by the received message and the Inferior becomes prepared and issues a PREPARED
1151 message, which is also retained. When the application response is available, it is sent with the
1152 retained messages and the CONTEXT_REPLY (which indicates that the related ENROL will
1153 complete the enrolments implied by the earlier transmission of the CONTEXT).

1154 When this group of messages is received by the context-handler on the client side, the contained
1155 ENROL and PREPARED messages are forwarded to the Superior (whose address was on the
1156 original CONTEXT and so is known to the context-handler). An ENROLLED message is sent
1157 back to the context-handler, assuring it that the enrolment was successful and the application can
1158 progress. If enrollment fails and the business transaction is atomic, confirmation must be
1159 prevented – this responsibility falls on the context-handler and the client application, since the
1160 failure of the enrolment implies that Superior itself is inaccessible. If enrolment fails and the
1161 business transaction is a cohesion, the appropriate response is a matter for the application.

1162 With “one-shot”, if there are multiple Inferiors created as a result of a single application message,
1163 there is an ENROL and PREPARED message for each one sent related with the
1164 CONTEXT_REPLY. If an operation fails, a CANCELLED message may be sent instead of a
1165 PREPARED – if the Superior is atomic, this will ensure it cancels, if cohesive, the client
1166 application will be aware of this and behave appropriately.

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

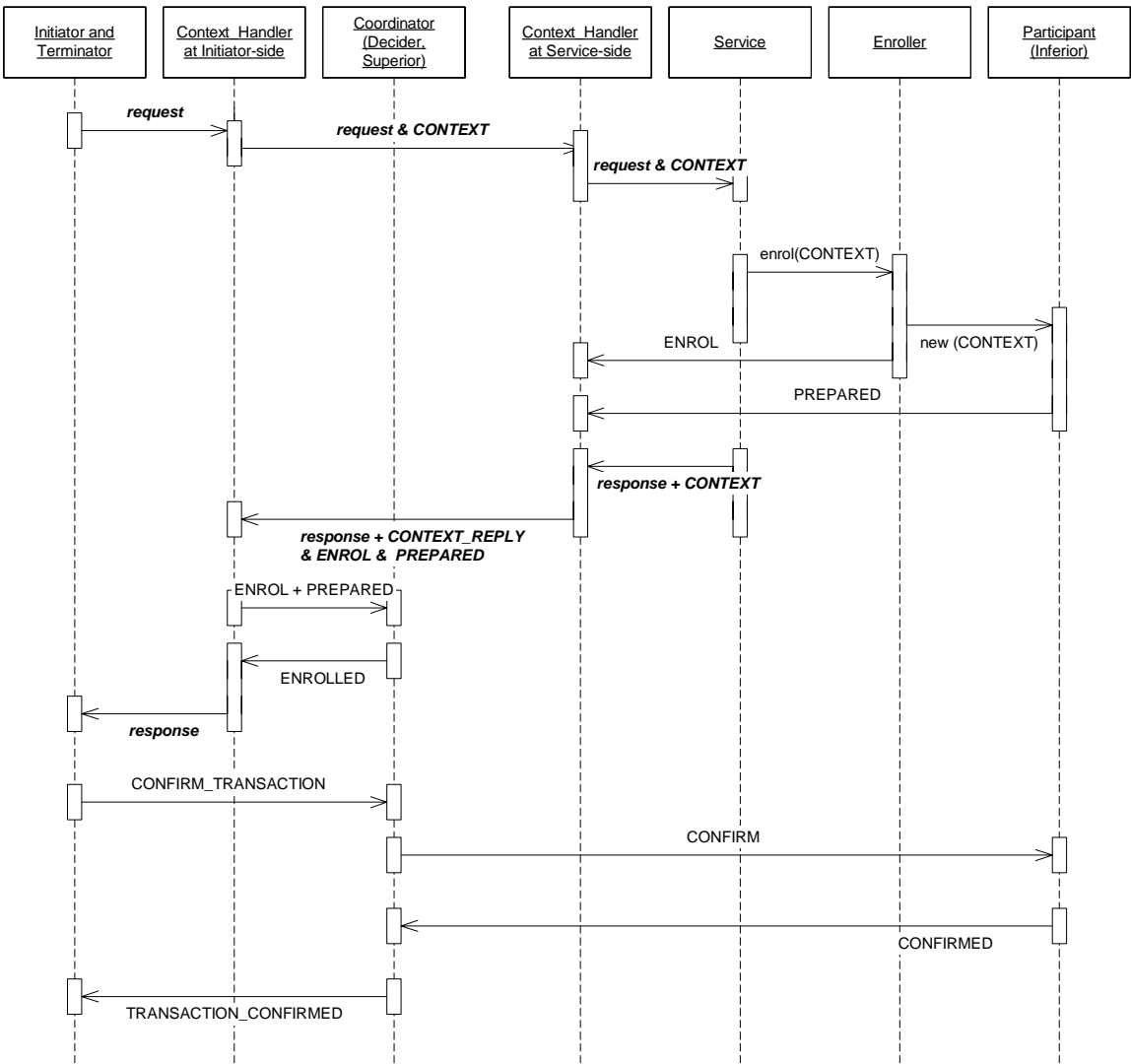


Figure 16 A message sequence showing the “one-shot” optimisation

Resignation

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

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

One-phase confirmation

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

Autonomous cancel, autonomous confirm and contradictions

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

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

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

1224 If the Superior's decision is contradicted by the autonomous decision, the Superior may need to
1225 record this, report it to management systems or inform the Terminator application or its own
1226 Superior. When this has been done (details are implementation-specific, but may be constrained
1227 by the application), the Superior sends a CONTRADICTION message to the Inferior. If an
1228 outcome message was sent earlier (crossing the announcement of the autonomous decision), the
1229 Inferior will already know there was a contradiction, but the receipt of the CONTRADICTION
1230 message informs the Inferior that the Superior knows and has done whatever it considers
1231 necessary to cope.

1232 As mentioned, BTP allows an Inferior to inform the Superior, with a qualifier on the PREPARED
1233 message, that the promise to remain in the prepared state will expire. In turn this allows the
1234 application on the Superior side to avoid risking a contradictory decision by making and sending
1235 its own decision in time. The Superior side can also indicate, with another qualifier, a minimum
1236 time for which it expects the prepared promise to remain valid.

1237

1238 As well as deliberate and forewarned autonomous decisions, BTP recognises that failures and
1239 exceptional conditions may force unplanned autonomous decisions. In the protocol sequence
1240 these are treated exactly like planned autonomous decisions – if they contradict, the Superior will
1241 be informed and a CONTRADICTION message sent to the Inferior.

1242 Autonomous decisions, planned or unplanned, are equivalent to the heuristic decisions of other
1243 transaction systems. The term is avoided in BTP since it may carry implications that it only
1244 occurs in an unplanned manner.

1245 **Recovery and failure handling**

1246 **Types of failure**

1247 BTP is designed to ensure the delivery of a consistent decision for a business transaction to the
1248 parties involved, even in the event of failure. Failures can be classified as:

1249 **Communication failure:** messages between BTP actors are lost and not delivered. BTP
1250 assumes the carrier protocol ensures that messages are either delivered correctly (without
1251 corruption) or are lost, but does not assume that all losses are reported nor that messages
1252 sent separately are delivered in the order of sending.

1253 **Node failure (system failure, site failure):** a machine hosting one or more BTP actors
1254 stops processing and all its volatile data is lost. BTP assumes a site fails by stopping – it
1255 either operates correctly or not at all, it never operates incorrectly.

1256 Communication failure may become known to a BTP implementation by an indication from the
1257 lower layers or may be inferred (or suspected) by the expiry of a timeout. Recovery from a
1258 communication failure requires only that the two actors can again send messages to each other
1259 and continue or complete the progress of the business transaction.

1260 A node failure is distinguished from communication failure because there is loss of volatile state.
1261 To ensure consistent application of the decision of a business transaction, BTP requires that some
1262 state information will be persisted despite node failure. Exactly what real events correspond to

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

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

Persistent information

BTP **requires** that certain state information is persisted – these are information that records an Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s autonomous decision. Requiring the first two to be persistent ensures that a consistent decision can be reached for the business transaction and that it is delivered to all involved nodes, despite failure. Requiring an Inferior’s autonomous decision to be persistent allows BTP to ensure that, if the autonomous 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 node 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 allows continuation of the business transaction if the application desires it. 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, for example, on enrolment). This means that on recovery one side may have persistent information while the other does not. This occurs, among other cases, when an Inferior has decided to be prepared but the Superior never confirmed (so the decision is “presumed” to be cancelled), and when the Superior did confirm, the Inferior applied the confirmation and removed its persistent information but the acknowledgement message (CONFIRMED) was never received by the Superior.

Information to be persisted when an Inferior decides to be prepared has to 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 the addressing and identification information for the Superior. The information needed to apply the confirm or cancel decision will depend on the application and the associated operations.

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

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

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

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

Recovery messages

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

While in the active phase – i.e. prior to entering completion – there is no appropriate last message that can be sent. However, for active-phase recovery there needs to be some way for the BTP actors to determine that the peer is still there and still aware of the Superior:Inferior relationship. In this case, the peers can interrogate each other using the INFERIOR_STATE or

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

1345 SUPERIOR_STATE messages, informing the peer of their own state and requesting a response –
1346 which may be the opposite message, or one of the main BTP messages (which perhaps had been
1347 lost). If it is another SUP|INFERIOR_STATE message, that reply does not ask for a response.
1348 Receiving a SUP|INFERIOR_STATE messages that asks for a response does not require an
1349 immediate response – especially if an implementation is waiting to determine a decision (perhaps
1350 because it is itself waiting for a decision from elsewhere), an implementation may choose not to
1351 reply until it wishes too.

1352 The SUP|INFERIOR_STATE messages are also used as replies when the receiver of **any** of the
1353 Superior:Inferior message has determined that there is no corresponding state information – the
1354 targeted Superior or Inferior does not exist (or is known to have completed and is no longer an
1355 active entity). The SUP|INFERIOR_STATE messages with a status of “unknown” is the
1356 indication that the state information does not exist.

1357 The SUP|INFERIOR_STATE messages are also available as replies to any Superior:Inferior
1358 message in the (transient, one hopes) case where, after failure an implementation cannot currently
1359 determine whether the persistent information exists or not, or what its state is, and so cannot give
1360 a definitive answer. The SUP|INFERIOR_STATE messages with a status of “inaccessible” is the
1361 indication that the existence of state information cannot be determined. The receiver of such a
1362 message should normally treat it as a “retry later” suggestion.

1363 **Redirection**

1364 As described above, BTP uses the presume-abort model for recovery. A corollary of this is that
1365 there are cases where one side will attempt to re-establish communication when there is no
1366 persistent information for the relationship at the far-end, because that side either never reached a
1367 state where the state was persisted, or had been persisted, but then progressed to remove the state
1368 information. In such cases, it is important the side that is attempting recovery can distinguish
1369 between unsuccessful attempts to connect to the holder of the persistent information and when the
1370 information no longer exists. If the peer information does not exist, the side that is attempting
1371 recovery can draw appropriate conclusions (that the peer either was never prepared, never
1372 confirmed or has already completed) and complete its part of the transaction; if it merely fails to
1373 get through, it is stuck in attempting recovery.

1374 Two mechanisms are provided to assist implementation flexibility while allowing completion of
1375 Superior:Inferior relationships when only one side has any persistent information. The
1376 mechanisms are:

- 1377 • Address fields which provide the address that will be used by the peer to send messages
1378 to an actor (effectively a “callback address”) can be a set of addresses, which are
1379 alternatives, one of which is chosen as the target address for the future message. If the
1380 sender of that message finds the address does not work, it can try a different alternative.
- 1381 • The REDIRECT message can be used to inform the peer that an address previously
1382 given is no longer valid and to supply a replacement address (or set of addresses).
1383 REDIRECT can be issued either as a response to receipt of a message or spontaneously.

1384 The two mechanisms can be used in combination, with one or more of the original set of
1385 addresses just being a redirector, which does not itself ever have direct access to the state
1386 information for the transaction, but will respond to any message with an appropriate REDIRECT.

1387 REDIRECT as a message is only used on the Superior:Inferior relationship, where each side
1388 holds the address of the other. On the other relationships (e.g. Terminator:Decider), one side (e.g.
1389 Terminator) has the address of the other, and initiates all the message exchanges. However, the
1390 entity whose address is known to the other may itself move - e.g. if a Coordinator, which will be
1391 both Decider and Superior changes its address as a Superior, it will probably change its address as
1392 a Decider too. In this case, a FAULT reply to a misdirected message can be used, assuming there
1393 is some entity available at, or on the path to the old address that understands BTP sufficiently to
1394 provide the redirection information.

1395 Some implementations, in which a single addressable entity with one, constant address deals with
1396 all transactions, distinguishing them by identifier, will not need to supply “backup” addresses
1397 (and would only use REDIRECT if permanently migrated).

1398 **Terminator:Decider failures and transaction timelimit**

1399 BTP does not provide facilities or impose requirements on the recovery of Terminator:Decider
1400 relationships, other than allowing messages to be repeated. A Terminator may survive failures (by
1401 retaining knowledge of the Decider’s address and identifier), but this is an implementation option.
1402 Although a Decider (if it decides to confirm) will persist information about the confirm decision,
1403 it is not required, after failure, to remain accessible using the address it originally gave to the
1404 Initiator (and used by the Terminator). Any such recovery is an implementation option.

1405 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and thus
1406 no way of detecting that a Terminator has failed. The Decider always has the right to initiate
1407 cancellation, but if the application (Terminator) and the Decider have different views about how
1408 long a “long time” is, then either the Decider might wait unnecessarily for a completion request
1409 (e.g. CONFIRM_TRANSACTION) that will never arrive, or it might initiate cancellation while
1410 the application is still active. To avoid these irritations, a standard qualifier “Transaction
1411 timelimit” can be used (by the Initiator) to inform the Decider when it can assume the Terminator
1412 will not request confirmation and so it (the Decider) should initiate cancellation.

1413 **Contradictions and hazard**

1414 As described above (see “Autonomous cancel, autonomous confirm and contradictions”), in
1415 some circumstances an Inferior may apply a decision that is contradictory to the decision of the
1416 Superior. This can occur in a semi-planned manner, when the Inferior has announced a timeout on
1417 the PREPARED message but no outcome message has been received, or as a result of an
1418 exceptional condition that forces the Inferior to break the promise implicit in PREPARED,
1419 regardless of timers. In both cases, this is considered an autonomous decision by the Inferior. An
1420 autonomous decision, of itself, does not imply a contradiction – it only results in a contradiction if
1421 the decision is opposite to that of the Superior (in the case of a cohesive Superior, opposite to the
1422 decision that applies to this Inferior).

1423 In order to ensure that a contradiction is detected despite node and communication failures, it is
1424 required that information about the taking of the autonomous decision be persisted until a BTP

1425 message received from the Superior indicates either that there was no contradiction (the decisions
1426 were in line – CANCEL is received after an autonomous cancel or CONFIRM is received after an
1427 autonomous confirm) or that the Superior is aware of the contradiction (CONTRADICTION is
1428 received). Note that the Inferior will become aware of the fact of the contradiction when it
1429 receives the “wrong” message, but must retain the record of its own decision until it receives the
1430 CONTRADICTION message, which tells it the Superior knows too.

1431 The Superior’s action on becoming aware of the contradiction is not determined by this
1432 specification. In particular, if the Superior is a Sub-coordinator or Sub-composer, it is not
1433 required by this specification to report the contradiction to its own Superior (which may, for
1434 example, be controlled by a different organisation). The Superior may report the problem to
1435 management systems or record it for manual repair. However, BTP does provide mechanisms to
1436 report the contradiction to the next higher Superior (if there is one) or to the Terminator
1437 application element.

1438 A contradiction occurring in an Inferior will usually mean the immediate Superior has a “mixed”
1439 condition – some of the application work it was responsible for has confirmed, some has
1440 cancelled (and contrary to any cohesion confirm-set selection). If the Superior is a Sub-
1441 coordinator or Sub-composer, it can report the mixed condition to its own Superior with the
1442 HAZARD message. If the Superior is the top-most in the tree, it can report the problem with the
1443 INFERIOR_STATUSES message, which will detail the state of all the Inferiors. **Error!**
1444 **Reference source not found.** [shows a message sequence in a transaction tree with two levels.](#)
1445 [The Participant makes an autonomous cancel decision, but the Coordinator decides to confirm.](#)
1446 [The confirm decision from the Coordinator, passed on by the Sub-coordinator crosses with the](#)
1447 [CANCELLED message from the Participant. The Participant waits for the CANCELLED from](#)
1448 [the Sub-coordinator, which chooses to report the problem with HAZARD to the Coordinator.](#)

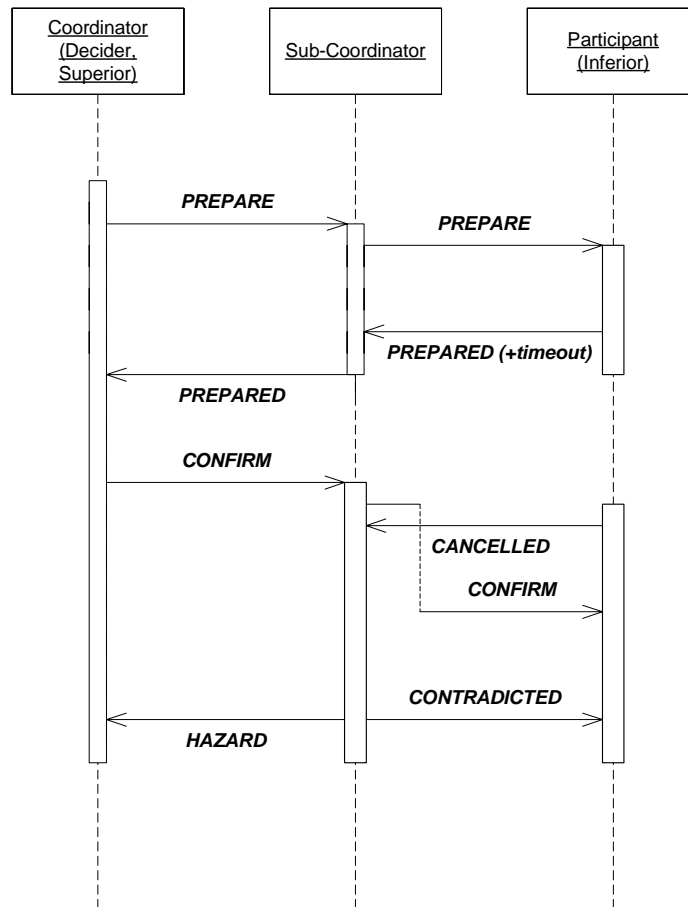


Figure 17 Message sequence showing contradiction, reported with HAZARD

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

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

Relation of BTP to application and carrier protocols

BTP messages are communicated between actors in two distinguishable circumstances:

- a) in establishing and progressing the outcome and control relationships between BTP actors, and between application elements and BTP actors – Initiator:Factory, Terminator:Decider, Superior:Inferior etc.

1466 b) in association with application messages that are communicated between application
1467 elements.

1468 In the first case, interoperable communication requires a specification of how the abstract BTP
1469 messages are represented and encoded, and how they are transmitted. This specification is a
1470 **carrier protocol binding** (or just “binding”, if the context is clear). BTP allows bindings to a
1471 multiplicity of carrier protocols. The only requirement that BTP makes is that the transmission of
1472 a message either delivers an uncorrupted message or fails. BTP does not require that the carrier
1473 report failure to deliver a message, to either side, nor that messages are delivered in the order they
1474 are sent (though implementations can take advantage of information from a richer carrier, which
1475 can improve performance in various ways). BTP messages communicated in this way have
1476 semantics that are defined in this specification – a PREPARE message (for example), refers back
1477 to the ENROL via the “inferior-identifier” parameter and is an instruction to the Inferior to
1478 become and report that it is prepared.

1479 In the second case, the full semantics cannot be defined in this specification. Interoperation with
1480 BTP requires that the parties have a common understanding of what is being confirmed or
1481 cancelled, but this mutual understanding is defined by the contract of the application, not by BTP.
1482 (The contract may be explicit or implicit, declared by one side as take-it-or-leave-it, or may be
1483 negotiated in some way.) Part of this contract will include how the combination of the application
1484 protocol (i.e. the application messages and their sequencing) and BTP operate such that the two
1485 sides are agreed as to which application operations are part of which business transaction. This
1486 will often be achieved by sending application messages and BTP messages in “association” in
1487 some way – thus an application message sent in association with a CONTEXT can be specified
1488 (by the application contract) to mean that if work is done as result of the receipt of the message,
1489 one or more Inferiors should be enrolled to apply the confirm/cancel decision to that work.
1490 Similarly, an application message may be sent associated with an ENROL with the contractual
1491 understanding that the message refers to some application work that has been made the
1492 responsibility of the Inferior being enrolled.

1493 The concrete representation of this “association” is also a matter for the application protocol
1494 specification. There are several ways this can be done, including:

- 1495 • the BTP message is contained within the application message, or both are contained
1496 within a larger construct;
- 1497 • the application message contains a field that is the superior-identifier or inferior-
1498 identifier that is also present on the CONTEXT or the ENROL
- 1499 • the BTP message contains a qualifier that references (a field of) the application message
1500 in some way (e.g. if the application message is an invoice, the qualifier might contain the
1501 invoice number)
- 1502 • the encoding of the BTP and application messages reference each other (e.g. using XML
1503 id and refid attributes)

1504 In all cases, the application specification⁵ will need to define the mechanism so that both parties
1505 have common understanding. Many applications will use the same mechanism and their
1506 specifications can therefore take advantage of standard patterns, and their implementations of
1507 standard tools.

1508 The association of an application message with a BTP message is analogous to the concept of
1509 “related” BTP messages. “Related” BTP messages are sent as a group, with a declared and
1510 defined semantic for the group. Associated application and BTP messages can be considered as
1511 “related”, with the proviso that the semantic is defined by the application, not by BTP.

1512 There is no necessary relationship between how the application messages and any associated BTP
1513 messages are transmitted by carrier protocols, and the carrier binding for the BTP messages. BTP
1514 messages are invariably sent to a BTP actor whose address has been passed to the sender by some
1515 means – thus a CONTEXT contains the address of the Superior to which ENROLs will be sent,
1516 and the ENROL contains the address of the Inferior. Similarly, BEGUN contains the address (as
1517 Decider) of the new Composer or Coordinator. These addresses are all sets of addresses (possibly
1518 of cardinality one), and each individual address identifies which binding is to be used. Thus, for
1519 example, when a CONTEXT is sent associated with an application message, the ENROL will
1520 travel on a carrier binding identified by the particular address from the CONTEXT that the
1521 Enroller chooses to use – which may have no relationship to how the application message arrived.

1522 Despite this, it will be common that the application binding and the BTP binding will use the
1523 same carrier. This is the case in the bindings specified in this edition of the specification, which
1524 define a binding of BTP to SOAP 1.1 over HTTP. Included in this SOAP/HTTP binding
1525 specification, are rules that allow an application to associate (relate) a single CONTEXT or a
1526 single ENROL (carried in the SOAP header) with the application message(s) carried in the SOAP
1527 body.

1528 **Other elements**

1529 **Identifiers**

1530 An Identifier is a globally unambiguous identification of the state corresponding to one of
1531 Decider, Superior or Inferior. Where a single entity has more than one of these roles (at the same
1532 node in the same transaction, as with a Sub-coordinator that is both Superior and Inferior), the
1533 Identifiers may be the same or different, at implementation option - they are distinguished by
1534 which messages the Identifier is used on. (A Superior has only one Superior-identifier, although it
1535 may be in multiple Superior:Inferior relationships, each with a separate state in terms of the state
1536 table).

1537 The state identified by an Identifier can be accessed by BTP messages sent to any of the addresses
1538 supplied with the Identifier in the appropriate message (CONTEXT, BEGUN, ENROL), or as
1539 updated by REDIRECT. An Identifier itself has no location implications. (Identifiers are
1540 specified, in the XML representation, as syntactically URIs - by their use as names of BTP

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

1541 entities, they are URNs. If an Identifier happens to specify an network location (i.e. it is a URL),
1542 it is treated as an opaque value by BTP)

1543 Identifiers are specified as being globally unambiguous - the same Identifier only ever identifies
1544 one Decider, Superior or Inferior over all systems and all time. In practice, an Identifier could be
1545 re-used if there is no possibility of the colliding values being confused. However implementations
1546 are recommended to use truly unambiguous Identifiers (that is to use them as URNs).

1547 **Addresses**

1548 In most cases, BTP actors that need to communicate are informed of each others addresses from
1549 received BTP messages. When an Inferior is to be enrolled, a CONTEXT message which
1550 contains the address of the Superior will have been received or otherwise passed to the Enroller
1551 and the Inferior. The ENROL message received by the Superior contains the address of the
1552 Inferior. The BEGUN returned from a Factory to the Initiator contains the address of the Decider,
1553 and this can be passed to the Terminator or any Status Requestor.

1554 The addresses carried in these messages (which are effectively “call-back” addresses, to be used
1555 as the destination of future messages) are sets of tripartite addresses. Each contains an identifier
1556 (binding name) for the binding to an underlying transport, or carrier protocol, a “binding
1557 address”, in a format specific to the carrier which is the information necessary to connect using
1558 that carrier, and an optional additional information field. This additional information is opaque to
1559 all but the future destination (which also created this address for itself) and is used however the
1560 implementation there wishes (e.g. it can be used to distinguish a particular program object, or to
1561 relay on, perhaps over a different protocol). The multiple members of the set allow support of
1562 multiple carrier bindings (including both different versions of standard bindings and proprietary
1563 bindings) and for relocation of the BTP actor.

1564 When a message is actually to be sent, the sender, possessing the set of addresses for the
1565 destination, chooses one - restricting its choice to bindings that it supports obviously, but not
1566 otherwise constrained by the specification. The binding address will be used by the senders
1567 carrier implementation (depending on the protocol, the address may or may not be transmitted –
1568 with http, for example, it is), The additional information, if present, will be included in the BTP
1569 message. The chosen address is considered the “target-address” when considering the abstract
1570 message, but only the additional information will normally appear within the encoded BTP-
1571 message (the encoding used is part of the binding specification, which could require that all of the
1572 address is (redundantly) transmitted, if the specifier so chose).

1573 Where a BTP message invokes a reply – as with the Initiator:Factory, Terminator:Decider and
1574 Status Requestor:various roles – the receiver (Factory, Decider, etc) of the message will not know
1575 *a priori* the address of the sender. Accordingly, in these cases the abstract messages are specified
1576 as containing a single “reply-address”. Depending on the binding, and the particular use of the
1577 binding, the “reply-address” may be directly represented in the encoding of the BTP message, or
1578 may be implicit in the carrier protocol. Similar considerations apply in the Superior:Inferior
1579 relationship, where although the addresses are normally known by the other side, there are cases
1580 when a message is received, and must be responded to, but the peer is unknown. Accordingly, the
1581 Superior:Inferior messages contain (in abstract) a single “senders-address”. As with the ~~the~~
1582 “reply-address”es, it may be implicit in the carrier protocol.

1583 The CONTEXT message does not contain a “target-address”, even as an abstract message, as it is
1584 never transmitted between BTP actors on its own – it is always either related to a BTP BEGIN or
1585 BEGUN message, or is passed between application elements with some (application-detailed)
1586 association with application messages.

1587 **Qualifiers**

1588 Qualifiers are elements of the BTP messages used to exchange additional information between
1589 the actors. Qualifiers can be specified in the BTP specification (“standard qualifiers”), by industry
1590 groups, by BTP implmentors or for the purposes of particular applications. Of the standard
1591 qualifiers in this version of the specification some are constraints on the BTP contract, such as
1592 time limits, and some are further identifiers used to distinguish specific parties in the BTP
1593 interchange. Non-standard qualifiers could extend the protocol or carry application-specific
1594 information.

1595 **Part 2. Normative Specification of BTP**

1596 **Actors, Roles and Relationships**

1597 Actors are software agents which process computations. BTP actors are addressable for the
1598 purposes of receiving application and BTP protocol messages transmitted over some underlying
1599 communications or carrier protocol. (See section “Addressing” for more detail.)

1600 BTP actors play roles in the sending, receiving and processing of messages. These roles are
1601 associated with responsibilities or obligations under the terms of software contracts defined by
1602 this specification. (These contracts are stated formally in the sections entitled “Abstract Messages
1603 and Associated Contracts” and “State Tables”.) A BTP actor’s computations put the contracts into
1604 effect.

1605 A role is defined and described in terms of a single business transaction. An implementation
1606 supporting a role may, as an addressable entity, play the same role in multiple business
1607 transactions, simultaneously or consecutively, or a separate addressable entity may be created for
1608 each transaction. This is a choice for the implementer, and the addressing mechanisms allow
1609 interoperation between implementations that make different choices.

1610 Within a single transaction, one actor may play several roles, or each role may be assigned to a
1611 distinct actor. This is again a choice for the implementer. An actor playing a role is termed an
1612 “actor-in-role”.

1613 Actors may interoperate, in the sense that the roles played by actors may be implemented using
1614 software created by different vendors for each actor-in-role. The section “Conformance”, gives
1615 guidelines on the groups of roles that may be implemented in a partial, interoperable
1616 implementation of BTP.

1617 The descriptions of the roles concentrate on the normal progression of a business transaction, and
1618 some of the more important divergences from this. They do not cover all exception cases – the
1619 message set definition and the state tables provide a more comprehensive specification.

1620 *Note – A BTP role is approximately equivalent to an interface in some distributed*
1621 *computing mechanisms, or a port-type in WSDL. The definition of a role includes*
1622 *behaviour.*

1623 **Relationships**

1624 There are two primary relationships in BTP.

- 1625 • Between an application element that determines that a business transaction should be
1626 completed (the role of Terminator) and the BTP actor at the top of the transaction tree (the
1627 role of Decider);
- 1628 • Between BTP actors within the tree, where one (the Superior) will inform the other (the
1629 Inferior) what the outcome decision is.

1630 These primary relationships are involved in arriving at a decision on the outcome of a business
1631 transaction, and propagating that decision to all parties to the transaction. Taking the path that is
1632 followed when a business transaction is confirmed:

- 1633 1. The Terminator determines that the business transaction should confirm, if it can; or
1634 (for a Cohesion), which parts should confirm
- 1635 2. The Terminator asks the Decider to apply the desired outcome to the tree, if it can
1636 guarantee the consistency of the confirm decision
- 1637 3. The Decider, which is Superior to one or more Inferiors, asks its Inferiors if they can
1638 agree to a confirm decision (for a Cohesion, this may not be all the Inferiors)
- 1639 4. If any of those Inferiors are also Superiors, they ask their Inferiors and so on down
1640 the tree
- 1641 5. Inferiors that are not Superiors report if they can agree to a confirm to their Superior
- 1642 6. Inferiors that are also Superiors report their agreement only if they received such
1643 agreement from their Inferiors, and can agree themselves
- 1644 7. Eventually agreement (or not) is reported to the Decider. If all have agreed, the
1645 Decider makes and persists the confirm decision (hence the term “Decider” – it
1646 decides, everything else just asked); if any have disagreed, or if the confirm decision
1647 cannot be persisted, a cancel decision is made
- 1648 8. The Decider, as Superior tells its Inferiors of the outcome
- 1649 9. Inferiors that are also Superiors tell their Inferiors, recursively down the tree
- 1650 10. The Decider replies to the Terminator’s request to confirm, reporting the outcome
1651 decision

1652 There are other relationships that are secondary to Terminator:Decider, Superior:Inferior, mostly
1653 involved in the establishment of the primary relationships. The various particular relationships

can be grouped as the “control” relationships – primarily Terminator:Decider, but also Initiator:Factory; and the “outcome” relationships – primarily Superior:Inferior, but also Enroller:Superior.

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

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

Roles

Figure 18 and Figure 19 show the BTP roles that are specialisations of the central Superior and Inferior roles.

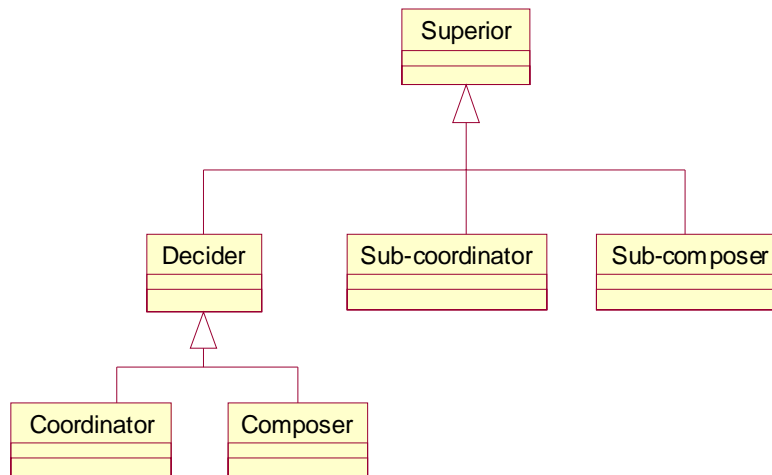


Figure 18 Superior and derived roles

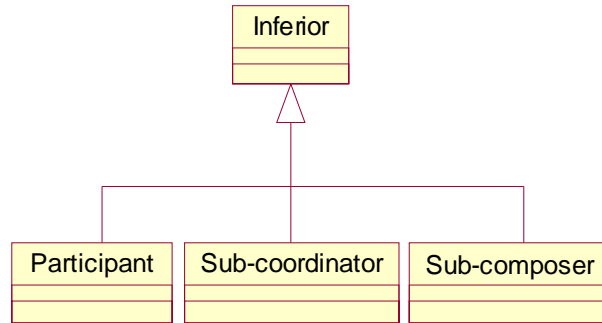


Figure 19 Inferior and derived roles

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

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

Roles involved in the outcome relationships

Superior

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

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

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

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

Superior receives	Superior sends
ENROL	ENROLLED
	PREPARE

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

1702

1703 Receipt of ENROL establishes a new Superior:Inferior relationship (unless the ENROL is a
1704 duplicate). ENROLLED is sent only if a reply is asked for on the ENROL.

1705 Inferior

1706 Responsible for applying the Outcome to some set of associated operations – the application
1707 determines which operations are the responsibility of a particular Inferior.

1708 An Inferior is **Enrolled** with a single Superior (hereafter referred to as “its Superior”),
1709 establishing a Superior:Inferior relationship. If the Inferior is able to ensure that either a confirm
1710 or cancel decision can be applied to the associated operations, and can persist information to
1711 retain that condition, it sends a PREPARED message to the Superior. When the Outcome is
1712 received from the Superior, the Inferior applies it, deletes the persistent information, and replies
1713 with CANCELLED or CONFIRMED as appropriate.

1714 If an Inferior is unable to come to a prepared state, it cancels the associated operations and
1715 informs the Superior with a CANCELLED message. If it is unable to either come to a prepared
1716 state, or to cancel the associated operations, it informs the Superior with a HAZARD message.

1717 An Inferior that has become prepared may, exceptionally, make an autonomous decision to be
1718 applied to the associated operations, without waiting for the Outcome from the Superior. It is
1719 required to persist this autonomous decision and report it to the Superior with CONFIRMED or
1720 CANCELLED as appropriate. If, when CONFIRM or CANCEL is received, the autonomous
1721 decision and the decision received from the Superior are contradictory, the Inferior must retain
1722 the record of the autonomous decision until receiving a CONTRADICTION message.

Inferior receives	Inferior sends
PREPARE	
CONFIRM	
CANCEL	
RESIGNED	

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

1723

1724 Enroller

1725 Causes the enrolment of an Inferior with a Superior. This role is distinguished because in some
1726 implementations the enrolment request will be performed by the application, in some the
1727 application will ask the actor that will play the role of Inferior to enrol itself, and a Factory may
1728 enrol a new Inferior (which will also be Superior) as a result of receiving BEGIN&CONTEXT.

Enroller sends	Enroller receives
ENROL	ENROLLER

1729

1730 ENROLLED is received only if the Enroller asked for a response when the ENROL was sent.

1731 An ENROL message sent from an Enroller that did not require an ENROLLED response may be
1732 modified *en route* to the Superior by an intermediate actor to ask for an ENROLLED response to
1733 be sent to the intermediate. (This may occur in the “one-shot” scenario, where an ENROL/no-rsp-
1734 req is received in relation to a CONTEXT_REPLY/related; the receiver of the
1735 CONTEXT_REPLY will need to ensure the enrolment is successful).

1736 Participant

1737 An Inferior which is specialized for the purposes of an application. Some application operations
1738 are associated directly with the Participant, which is responsible for determining whether a
1739 prepared condition is possible for them, and for applying the outcome. (“associated directly” as
1740 opposed to involving another BTP Superior:Inferior relationship, in which this actor is the
1741 Superior).

1742 The associated operations may be performed by the actor that has the role of Participant, or they
1743 may be performed by another actor, and only the confirm/cancel application is performed by the
1744 Participant.

1745 In either case, the Participant, as part of becoming prepared (i.e. before it can send PREPARED
1746 to the Superior), will persist information allowing it apply a confirm decision to the operations
1747 and to apply a cancel decision. The nature of this information depends on the operations.

1748 *Note – Possible approaches are:*

- 1749 • *The operations may be performed completely and the Participant persists*
- 1750 *information to perform counter-effect operations (compensating operations) to*
- 1751 *apply cancellation;*
- 1752 • *The operations may be just checked and not performed at all; the Participant*
- 1753 *persists information to perform them to apply confirmation;*
- 1754 • *The Participants persists the prior state of data affected by the operations and the*
- 1755 *operations are performed; the Participant restores the prior state to apply*
- 1756 *cancellation;*
- 1757 • *As the previous, but other access to the affected data is forbidden until the decision*
- 1758 *is known*

1759 Since a Participant is an Inferior, it sends and receives the messages for an Inferior.

1760 **Sub-coordinator**

1761 An Inferior which is also an Atomic Superior.

1762 A sub-coordinator is the Inferior in one Superior:Inferior relationship and the Superior in one or

1763 more Superior:Inferior relationships.

1764 From the perspective of its Superior (the one the sub-coordinator is Inferior to), there is no

1765 difference between a sub-coordinator and any other Inferior. From this perspective, the

1766 “associated operations” of the sub-coordinator as an Inferior include the relationships with its

1767 Inferiors.

1768 A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and

1769 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is propagated

1770 to all Inferiors.

1771 Since a Sub-coordinator is both an Inferior and a Superior, it sends and receives the messages for

1772 both.

1773 **Sub-composer**

1774 An Inferior which is also a Cohesive Superior.

1775 Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from the

1776 perspective of its Superior.

1777 A sub-composer is similar to a sub-coordinator, except that the constraints linking the different

1778 Inferiors concern only those Inferiors in the confirm-set. How the confirm-set is controlled, and

1779 when, is not defined in this specification.

1780 If the sub-composer is instructed to cancel, by receiving a CANCEL message from its Superior,

1781 the cancellation is propagated to all its Inferiors.

1782 Since a Sub-composer is both an Inferior and a Superior, it sends and receives the messages for

1783 both.

1784 **Roles involved in the control relationships**

1785 **Decider**

1786 A Superior that is not also the Inferior on a Superior:Inferior relationship. It is the top-node in the
1787 transaction tree and receives requests from a Terminator as to the desired outcome for the
1788 business transaction. If the Terminator asks the Decider to confirm the business transaction, it is
1789 the responsibility of the Decider to finally take the confirm decision. The taking of the decision is
1790 synonymous with the persisting of information identifying the Inferiors that are to be confirmed.
1791 An Inferior cannot be confirmed unless PREPARED has been received from it.

1792 A Decider is instructed to cancel by receiving CANCEL_TRANSACTION.

1793 A Decider that is an Atomic Superior (all Inferiors will have the same outcome) is a Coordinator.

1794 A Decider that is a Cohesive Superior (some Inferiors may cancel, some confirm) is a Cohesion.

Decider receives	Decider sends
CONFIRM_TRANSACTION	TRANSACTION_CONFIRMED TRANSACTION_CANCELLED INFERIOR_STATUSES
CANCEL_TRANSACTION	TRANSACTION_CANCELLED INFERIOR_STATUSES
REQUEST_INFERIOR_STATUSES	INFERIOR_STATUSES

1795

1796 A Decider is also a Superior and thus sends and receives the messages for a Superior.

1797 **Coordinator**

1798 A Decider that is an Atomic Superior. The same outcome decision will be applied to all Inferiors
1799 (excluding any from which RESIGN is received).

1800 PREPARED must be received from all remaining Inferiors for a confirm decision to be taken.

1801 A Coordinator must make a cancel decision if

- 1802 • it is instructed to cancel by the Terminator
- 1803 • if CANCELLED is received from any Inferior
- 1804 • if it is unable to persist a confirm decision

1805 Since a Coordinator is a Decider, it receives the mssages appropriate for a Decider and a
1806 Superior.

1807 **Composer**

1808 A Decider that is a Cohesive Superior. If the Terminator requests confirmation of the Cohesion,
1809 that request will determine the confirm-set of the Cohesion.

1810 PREPARED must be received from all Inferiors in the confirm-set (excluding any from which
1811 RESIGN is received) for a confirm decision to be taken.

1812 A Composer must make a cancel decision (applying to all Inferiors) if

- 1813 • it is instructed to cancel by the Terminator
- 1814 • if CANCELLED is received from any Inferior in the confirm-set
- 1815 • if it is unable to persist a confirm decision

1816 A Composer may be asked to prepare some or all of its Inferiors by receiving
1817 PREPARE_INFERIORS. It issues PREPARE to any of those Inferiors from which none of
1818 PREPARED, CANCELLED or RESIGN have been received, and replies to the
1819 PREPARE_INFERIORS with INFERIOR_STATUSES.

1820 A Composer may be asked to cancel some of its Inferiors, but not itself, by receiving
1821 CANCEL_INFERIORS.

Composer receives	Composer sends
PREPARE_INFERIORS	INFERIOR_STATUSES
CANCEL_INFERIORS	INFERIOR_STATUSES

1822 Terminator

1823 Asks a Decider to confirm the business transaction, or instructs it to cancel all or (for a Cohesion)
1824 part of the business transaction.

1825 All communications between Terminator and Decider are initiated by the Terminator. A
1826 Terminator is usually an application element.

1827 A request to confirm is made by sending CONFIRM_TRANSACTION to the target Decider. If
1828 the Decider is a Cohesion Composer, the Terminator may select which of the Composer's
1829 Inferiors are to be included in the confirm-set. If the Decider is an Atom Coordinator, all Inferiors
1830 are included. After applying the decision, the Decider replies with
1831 TRANSACTION_CONFIRMED, TRANSACTION_CANCELLED or (in the case of problems)
1832 INFERIOR_STATUSES.

1833 A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its Inferiors
1834 with PREPARE_INFERIORS. The Composer replies with INFERIOR_STATUSES.

1835 A Terminator may send CANCEL_TRANSACTION to instruct the Decider to cancel the whole
1836 business transaction.. The Decider replies with CANCEL_COMPLETE if all Inferiors cancel
1837 successfully, and with INFERIOR_STATUSES in the case of problems.. If the Decider is a
1838 Cohesion Composer, the Terminator may send CANCEL_INFERIORS to cancel some of the
1839 Inferiors; the Decider always replies with INFERIOR_STATUSES.

1840 A Terminator may check the status of the Inferiors of the Decider by sending
1841 REQUEST_INFERIOR_STATUSES. The Decider replies with INFERIOR_STATUSES.

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

1842 Initiator

1843 Requests a **Factory** to create a Superior – this will either be a Decider (representing a new top-
1844 level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an existing
1845 business transaction.

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

1846

1847 The received CONTEXT is that for the new Superior.

1848 Factory

1849 Creates Superiors and returns the CONTEXT for the new Superior. The following types of
1850 Superior are created :

1851 Decider, which is either
1852 Composer or
1853 Coordinator
1854 Sub-composer
1855 Sub-coordinator
1856

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

1857

1858 If the BEGIN has no related CONTEXT, the Factory creates a Decider, either a Cohesion
1859 Composer or an Atom Coordinator, as determined by the “superior type” parameter on the
1860 BEGIN.

1861 If the BEGIN has a related CONTEXT, the new Superior is also enrolled as an Inferior of the
1862 Superior identified by the CONTEXT. The new Superior is thus a sub-composer or sub-
1863 coordinator, as determined by the “superior type” parameter on the BEGIN.

1864 **Other roles**

1865 **Redirector**

1866 Sends a REDIRECT message to inform a Superior or Inferior that an address previously supplied
1867 for the peer (i.e. an Inferior or Superior, respectively) is no longer appropriate, and to supply a
1868 new address or set of addresses to replace the old one.

1869 A Redirector may send a REDIRECT message in response to receiving a message using the old
1870 address, or may send REDIRECT at its own initiative.

1871 If a Superior moves from the superior-address in its CONTEXT, or an Inferior moves from the
1872 inferior-address in the ENROL message, the implementation **must** ensure that a Redirector
1873 catches any inbound messages using the old address and replies with a REDIRECT message
1874 giving the new address. (Note that the inbound message may itself be a REDIRECT message, in
1875 which case the Redirector shall use the new address in the received message as the target for the
1876 REDIRECT that it sends.)

1877 After receiving a REDIRECT message, the BTP actor **must** use the new address not the old one,
1878 unless failure prevents it updating its information.

Redirector receives	Redirector sends
Any message for Superior or Inferior	REDIRECT

1879 **Status Requestor**

1880 Requests and receives the current status of a transaction tree node – any of an Inferior, Superior
1881 or Decider, or the current status of the nodes relationships with its Inferiors, if any. The role of
1882 Status Requestor has no responsibilities – it is just a name for where the REQUEST_STATUS
1883 and REQUEST_INFERIOR_STATUSES comes from (REQUEST_INFERIOR_STATUSES is
1884 also issued by a Terminator to a Decider).

Status Requestor sends	Status Requestor receives
REQUEST_STATUS	STATUS
REQUEST_INFERIOR_STATUS	INFERIOR_STATUSES

1885

1886 The receiver of the request can refuse to provide the status information by replying with
1887 FAULT(StatusRefused). The information returned in STATUS will always relate to the
1888 transaction tree node as a whole (e.g. as an Inferior, even if it is also a Superior).

Summary of relationships

Figure 20 summarises the relationships between the BTP roles. BTP can be implemented using proprietary equivalents of the Terminator and Decider roles.

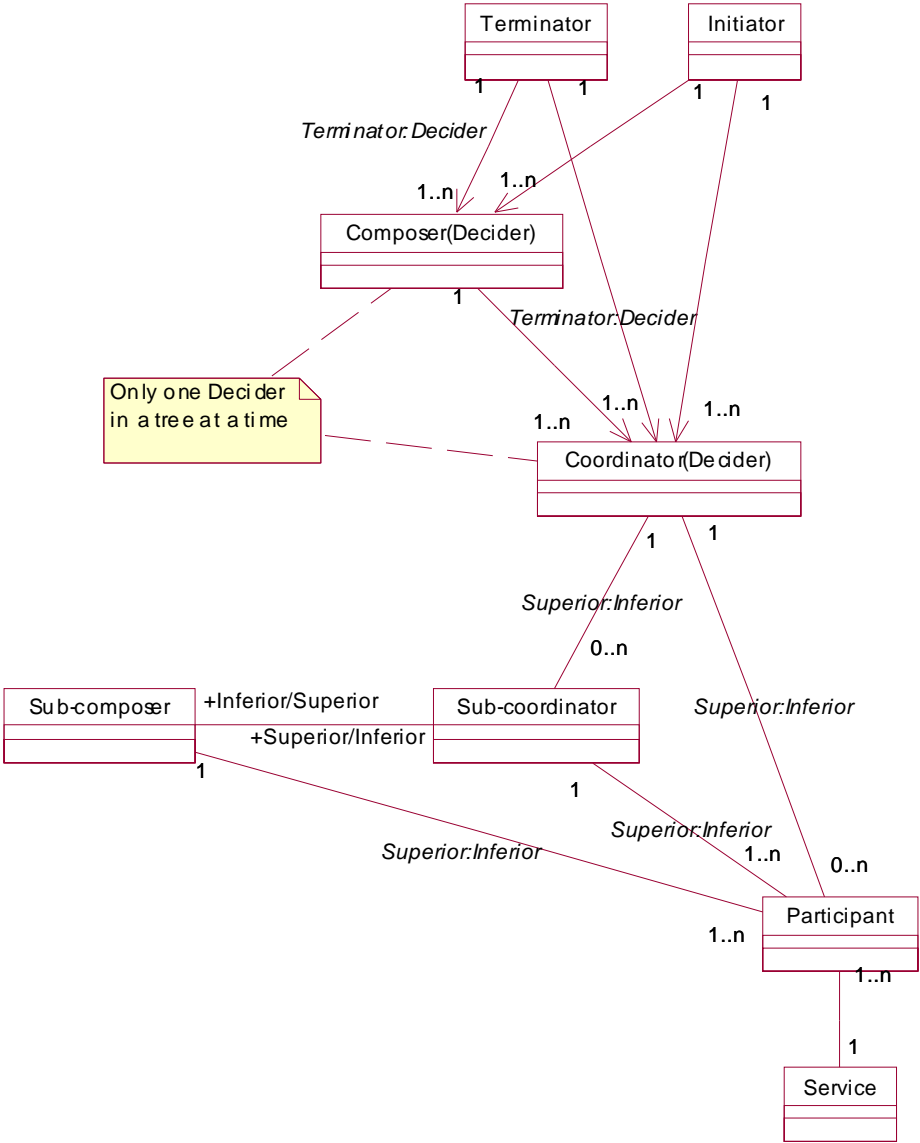


Figure 20 Summary of relationships between roles

Abstract Messages and Associated Contracts

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

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

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

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

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

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

Addresses

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

The detailed format of the address will depend on the particular carrier protocol, but at this abstract level is considered to have three parts. The first part, the "binding name", identifies the binding to a particular carrier protocol – some bindings are specified in this document, others can be specified elsewhere. The second part of the address, the "binding address", is meaningful to

1931 the carrier protocol itself, which will use it for the communication (i.e. it will permit a message to
1932 be delivered to a receiver). The third part, “additional information”, is not used or understood by
1933 the carrier protocol. The “additional information” may be a structured value.

1934 When a message is actually transmitted, the “binding name” of the target address will identify
1935 which carrier protocol is in use and the “binding address” will identify the destination, as known
1936 to the carrier protocol. The entire binding address is considered to be “consumed” by the carrier
1937 protocol implementation. All of it may be used by the sending implementation, or some of it may
1938 be transmitted in headers, or as part of a URL in the carrier protocol, but then used or consumed
1939 by the receiving implementation of the carrier protocol to direct the BTP message to a BTP-aware
1940 entity (BTP-aware in that it is capable of interpreting the BTP messages). The “additional
1941 information” of the target address will be part of the BTP message itself and used in some way by
1942 the receiving BTP-aware entity (it could be used to route the message on to some other BTP
1943 entity). Thus, for the target address, only the “additional information” field is transmitted in the
1944 BTP message and the “additional information” is opaque to parties other than the recipient.

1945 For other addresses in BTP messages, all three components will be within the message.

1946 All messages that concern a particular Superior:Inferior relationship have an identifier parameter
1947 for the target side as well as the target address. This allows full flexibility for implementation
1948 choices – an implementation can:

1949 a) Use the same binding address and additional information for multiple business
1950 transactions, using the identifier parameter to locate the relevant state
1951 information;

1952 b) Use the same binding address for multiple business transactions and use the
1953 additional information to locate the information; or

1954 c) Use a different binding address for each business transaction.

1955 Which of these choices is used is opaque to the entity sending the message – both parts of the
1956 address and the identifier originated at the recipient of this message (and were transmitted as
1957 parameters of earlier messages in the opposite direction).

1958 BTP recovery requires that the state information for a Superior or Inferior is accessible after
1959 failure and that the peer can distinguish between temporary inaccessibility and the permanent
1960 non-existence of the state information. As is explained in “**Error! Reference source not found.**”
1961 below, BTP provides mechanisms – having a set of BTP addresses for some parameters, and the
1962 REDIRECT message – that make this possible, even if the recovered state information is on a
1963 different address to the original one (as may be the case if case c) above is used).

1964 **Request/response pairs**

1965 Many of the messages combine in pairs as a request and its response. However, in some cases the
1966 response message is sent without a triggering request, or as a possible response to more than one
1967 type of request. To allow for this, the abstract message set treats each message as standalone; but
1968 where a request does expect a reply, a “reply-address” parameter will be present. For any
1969 message with a reply address parameter, in the case of certain errors, a FAULT message will be
1970 sent to the reply address instead of the expected reply.

1971 Between Superior and Inferior the address of the peer is normally known (from the “superior-
1972 address” on an earlier CONTEXT or the “inferior-address” on a received ENROL). However, in
1973 some cases a message will be received for a Superior or Inferior that is not known – the state
1974 information no longer exists. This is not an exceptional condition but occurs when one side has
1975 either not created or has removed its persistent state in accordance with the procedures, but a
1976 message has got lost in a failure, and the peer still has state information. The response to a
1977 message for an unknown (and logically non-existent) Superior is SUPERIOR_STATE/unknown,
1978 for an unknown Inferior it is INFERIOR_STATE/unknown. However, since the intended target is
1979 unknown, there is no information to locate the peer, which sent the undeliverable message. To
1980 enable the receiver to reply with the appropriate *_STATE/unknown, all the messages between
1981 Superior and Inferior have a “senders-address” parameter. If a FAULT message is to be sent in
1982 response to message which (as an abstract message) has a “senders-address” parameter, the
1983 FAULT message is sent to that address.

1984 *Note – Both reply-address and senders-address may be absent when the carrier protocol*
1985 *itself has a request/response pattern. In these cases, the reply or sender address is*
1986 *implicitly that of the sender of the request (and thus the destination of a response)*

1987 **Compounding messages**

1988 BTP messages may be sent in combination with each other, or with other (application) messages.
1989 There are two cases:

- 1990 a) Sending the messages together where the combination has semantic
1991 significance. One message is said to be “related to” the other – the combination
1992 is termed a “group”.
- 1993 b) Sending of the messages where the combination has no semantic significance,
1994 but is merely a convenience or optimisation. This is termed “bundling” – the
1995 combination is termed a “bundle”.

1996 The form A&B is used to refer to a combination (group) where message B is sent in relation to A
1997 (“relation” is asymmetric). The form A+B is used to refer to A and B bundled together- the
1998 transmission of the bundle "A+B" is semantically identical to the transmission of A followed by
1999 the transmission of B.

2000 Only certain combinations of messages are possible in a group, and the meaning of the relation is
2001 specifically defined for each such combination in the next section. A particular group is treated as
2002 a unit for transmission – it has a single target address. This is usually that of one of the messages
2003 in the group – the specification for the group defines which.

2004 A “bundle” of messages may contain both unrelated messages and groups of related messages.
2005 The only constraint on which messages and groups can be bundled is that all have the same
2006 binding address, but may have different “additional information” values. (Messages within a
2007 related group may have different addresses, where the rules of their relatedness permit this).
2008 Unless constrained by the binding, any messages or groups that are to be sent to the same binding
2009 address may be bundled – the fact that the binding addresses are the same is a necessary and
2010 sufficient condition for the sender to determine that the messages can be bundled.

2011 A particular and important case of related messages is where a BTP CONTEXT message is sent
2012 related to an application message. In this case, the target of the application message defines the
2013 destination of the CONTEXT message. The receiving implementation may in fact remove the
2014 CONTEXT before delivering the application message to the application (Service) proper, but
2015 from the perspective of the sender, the two are sent to the same place.

2016 The compounding mechanisms, and the multi-part address structures, support the “one-wire” and
2017 “one-shot” communication patterns.

2018 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,
2019 including the associated application messages, pass via the same “endpoints”. These “endpoints”
2020 may in fact be relays, routing messages on to particular actors within their domain. The onward
2021 routing will require some further addressing, but this has to be opaque to the sender. This can be
2022 achieved if the relaying endpoint ensures that all addresses for actors in its domain have the
2023 relay’s address as their binding address, and any routing information it will need in its own
2024 domain is placed in the additional information. (This may involve the relay changing addresses in
2025 messages as they pass through it on the way out). On receiving a message, it determines the
2026 within-domain destination from the received additional information (which is thus rewritten) and
2027 forwards the message appropriately. The sender is unaware of this, and merely sees addresses
2028 with the same binding address, which it is permitted to bundle. The content of the “additional
2029 information” is a matter only for the relay – it could put an entire BTP address in there, or other
2030 implementation-defined information. Note that a quite different one-wire implementation can be
2031 constructed where there is no relaying, but the receiving entity effectively performs all roles,
2032 using the received identifiers to locate the appropriate state.

2033 “One-shot” communication makes it possible to send an application message, receive the
2034 application reply, enrol an Inferior to be responsible for the confirm/cancel of the operations of
2035 those message and inform the Superior that the Inferior is prepared, all in one two-way exchange
2036 across the network (e.g. one request/reply of a carrier protocol).. The application request is sent
2037 with a related CONTEXT message. The application response is sent with a relation group of
2038 CONTEXT_REPLY/related, ENROL/no-rsp-req message and a PREPARED message. This is
2039 possible even if the Superior address is different from the address of the application element that
2040 sends the original message (if the application exchange is request/reply, there may not even be an
2041 identifiable address for the application element). The target addresses of the ENROL and
2042 PREPARED (the Superior address) are not transmitted; the actor that was originally responsible
2043 for adding the CONTEXT to the outbound application message remembers the Superior address
2044 and forwards the ENROL and PREPARED appropriately.

2045 With “one-shot”, if there are multiple Inferiors created as a result of a single application message,
2046 there is an ENROL and PREPARED message for each sent related to the CONTEXT_REPLY. If
2047 an operation fails, a CANCELLED message is sent instead of a PREPARED.

2048 If the CONTEXT has “superior-type” of “atom”, then subsequent messages to the same Service,
2049 with the same related CONTEXT/atom, can have their associated operations put under the control
2050 of the same Inferior, and only a CONTEXT_REPLY/completed is sent back with the response (if
2051 the new operations fail, it will be necessary to send back CONTEXT_REPLY/repudiated, or send
2052 CANCELLED). If the “superior type” on the CONTEXT is “cohesive”, each operation will
2053 require separate enrolment.

2054 Whether the “one-shot” mechanism is used is determined by the implementation on the
2055 responding (Inferior) side. This may be subject to configuration and may also be constrained by
2056 the application or by the binding in use.

2057 **Extensibility**

2058 To simplify interoperation between implementations of this edition of BTP with implementations
2059 of future editions, the “must-be-understood” sub-parameter as specified for Qualifiers may be
2060 defined for use with any parameter added to an existing message in a future revision of this
2061 specification. The default for “must-be-understood” shall be “true”, so an implementation
2062 receiving an unrecognised parameter without a “false” value for “must-be-understood” shall not
2063 accept it (the FAULT value “UnrecognisedParameter” is available, but other errors, including
2064 lower-layer parsing/unmarshalling errors may be reported instead). If “must-be-understood” with
2065 the value “false” is present as a sub-parameter of a parameter in any message, a receiving
2066 implementation **should** ignore the parameter.

2067 How the sub-parameter is associated with the new parameter is determined by the particular
2068 binding.

2069 No special mechanism is provided to allow for the introduction of completely new messages.

2070 **Messages**

2071 **Qualifiers**

2072 All messages have a Qualifiers parameter which contains zero or more Qualifier values. A
2073 Qualifier has sub-parameters:

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

2074

2075 **Qualifier group** ensures the Qualifier name is unambiguous. Qualifiers in the same group
2076 need not have any functional relationship. The qualifier group will typically be used to
2077 identify the specification that defines the qualifier’s meaning and use. Qualifiers may
2078 be defined in this or other standard specifications, in specifications of a particular
2079 community of users or of implementations or by bilateral agreement.

2080 **Qualifier name** this identifies the meaning and use of the Qualifier, using a name that is
2081 unambiguous within the scope of the Qualifier group.

2082 **Must-be-understood** if this has the value “true” and the receiving entity does not
 2083 recognise the Qualifier type (or does not implement the necessary functionality), a
 2084 FAULT “UnsupportedQualifier” shall be returned and the message shall not be
 2085 processed. Default is “true”.

2086 **To-be-propagated** if this has the value “true” and the receiving entity passes the BTP
 2087 message (which may be a CONTEXT, but can be other messages) onwards to other
 2088 entities, the same Qualifier value shall be included. If the value is “false”, the Qualifier
 2089 shall not be automatically included if the BTP message is passed onwards. (If the
 2090 receiving entity does support the qualifier type, it is possible a propagated message
 2091 may contain another instance of the same type, even with the same Content – this is
 2092 not considered propagation of the original qualifier.). Default is “false”.

2093 **Content** the type (which may be structured) and meaning of the content is defined by the
 2094 specification of the Qualifier.

2095 **Messages not restricted to outcome or control relationships.**

2096 The messages in this section are used between various roles. CONTEXT message is used in the
 2097 Initiator:Factory relationship (when it is related to BEGIN or to BEGUN), and related to an
 2098 application ‘message’ to propagate the business transaction between parts of the
 2099 application. CONTEXT_REPLY is used as the reply to a CONTEXT.REQUEST_STATUS can
 2100 be issued to, and STATUS returned by any of Decider, Superior or Inferior. FAULT can be used
 2101 on any relationship to indicate an error condition back to the sender of a message.

2102 **CONTEXT**

2103 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more application
 2104 messages. (The means by which this relationship is represented is determined by the binding and
 2105 the binding mechanisms of the application protocol.) The “superior-type” parameter identifies
 2106 whether the Superior will apply the same decision to all Inferiors enrolled using the same superior
 2107 identifier (“superior-type” is “atom”) or whether it may apply different decisions (“superior-type”
 2108 is “cohesion”).

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

2109

2110 **superior-address** the address to which ENROL and other messages from an enrolled
 2111 Inferior are to be sent. This can be a set of alternative addresses.

2112 **superior-identifier** identifies the Superior. This shall be globally unambiguous.

2113 **superior-type** identifies whether the CONTEXT refers to a Cohesion or an Atom. Default
 2114 is atom.

2115 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction timelimit”
 2116 is carried by CONTEXT.

2117 **reply-address** the address to which a replying CONTEXT_REPLY is to be sent. This may
 2118 be different each time the CONTEXT is transmitted – it refers to the destination of a
 2119 replying CONTEXT_REPLY for this particular transmission of the CONTEXT.

2120 There is no “target-address” parameter for CONTEXT as it is only transmitted in relation to the
 2121 application messages, BEGIN and BEGUN.

2122 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the
 2123 “superior-type” with the appropriate value.

2124 CONTEXT_REPLY

2125 CONTEXT_REPLY is sent after receipt of CONTEXT (related to application message(s)) to
 2126 indicate whether all necessary enrolments have already completed (ENROLLED has been
 2127 received) or will be completed by ENROL messages sent in relation to the CONTEXT_REPLY
 2128 or if an enrolment attempt has failed. CONTEXT_REPLY may be sent related to an application
 2129 message (typically the response to the application message related to the CONTEXT). In some
 2130 bindings the CONTEXT_REPLY may be implicit in the application message.
 2131 CONTEXT_REPLY is used in some of the related groups to allow BTP messages to be sent to a
 2132 Superior with an application message.

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

2133
 2134 **superior-identifier** the “superior-identifier” from the CONTEXT

2135 **completion-status:** reports whether all enrol operations made necessary by the receipt of
 2136 the earlier CONTEXT message have completed. Values are

Value	meaning
<i>completed</i>	All enrolments (if any) have succeeded already
<i>incomplete</i>	Further enrolments are possible (used only in related groups with other BTP messages)
<i>related</i>	At least some enrolments are to be

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

2137

2138 **qualifiers** standardised or other qualifiers.

2139 **target-address** the address to which the CONTEXT_REPLY is sent. This shall be the
2140 “reply-address” from the CONTEXT.

2141 The form CONTEXT_REPLY/completed, CONTEXT_REPLY/related and
2142 CONTEXT_REPLY/repudiated refer to CONTEXT_REPLY messages with status having the
2143 appropriate value. The form CONTEXT_REPLY/ok refers to either of
2144 CONTEXT_REPLY/completed or CONTEXT_REPLY/related.

2145 If there are no necessary enrolments (e.g. the application messages related to the received
2146 CONTEXT did not require the enrolment of any Inferiors), then CONTEXT_REPLY/completed
2147 is used.

2148 If a CONTEXT_REPLY/repudiated is received, the receiving implementation **must** ensure that
2149 the business transaction will not be confirmed.

2150 REQUEST_STATUS

2151 Sent to an Inferior, Superior or to a Decider to ask it to reply with STATUS. The receiver may
2152 reject the request with a FAULT(StatusRefused).

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

2153

2154 **target identifier** The identifier for the business transaction, or part of business transaction
2155 whose status is sought. If the target-address is a “decider-address”, this parameter shall
2156 be the “transaction-identifier” on the BEGUN message. If the “target-address” is an
2157 “inferior-address”, this parameter shall be the “inferior-identifier” on the ENROL
2158 message. If the “target-address” is a “superior-address”, this parameter shall be the
2159 “superior-identifier” on the CONTEXT.

2160 **qualifiers** standardised or other qualifiers.

2161 **target-address** the address to which the REQUEST_STATUS message is sent. This can
 2162 be any of “decider-address”, “inferior-address” or “superior-address”.

2163 **reply-address** the address to which the replying STATUS should be sent.

2164 Types of FAULT possible (sent to “reply-address”)

2165 ***General***

2166 ***Redirect** – if the intended target now has a different address*

2167 ***StatusRefused** – if the receiver is not prepared to report its status to the sender of this*
 2168 *message*

2169 ***UnknownTransaction** – if the target-identifier is unknown*

2170 **STATUS**

2171 Sent by a Inferior, Superior or Decider in reply to a REQUEST_STATUS, reporting the overall
 2172 state of the transaction tree node represented by the sender.

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

2173

2174 **responders-identifier** the identifier of the state, identical to the “target-identifier” on the
 2175 REQUEST_STATUS.

2176 **status** states the current status of the transaction tree node represented by the sender.
 2177 Some of the values are only issued if the sender is an Inferior. If the transaction tree
 2178 node is both Superior and Inferior (i.e. is a sub-coordinator or sub-composer), and two
 2179 status values would be valid for the current state, it is the sender’s option which one is
 2180 used.

status value	Meaning from Superior	Meaning from Inferior
<i>Created</i>	Not applicable	The Inferior exists (and is addressable) but it has not been enrolled with a Superior
<i>Enrolling</i>	Not applicable	ENROL has been sent, but ENROLLED is awaited

status value	Meaning from Superior	Meaning from Inferior
<i>Active</i>	New enrolment of inferiors is possible	The Inferior is enrolled
<i>Resigning</i>	Not applicable	RESIGN has been sent; RESIGNED is awaited
<i>Resigned</i>	Not applicable	RESIGNED has been received
<i>Preparing</i>	Not applicable	PREPARE has been received; PREPARED has not been sent
<i>Prepared</i>	Not applicable	PREPARED has been sent; no outcome has been received or autonomous decision made
<i>Confirming</i>	Confirm decision has been made or CONFIRM has been received as Inferior but responses from inferiors are pending	CONFIRM has been received; CONFIRMED/response has not bee sent
<i>Confirmed</i>	CONFIRMED/responses have been received from all Inferiors	CONFIRMED/response has been sent
<i>Cancelling</i>	Cancel decision has been made but responses from inferiors are pending	CANCEL has been received or auto-cancel has been decided
<i>Cancelled</i>	CANCELLED has been received from all Inferiors	CANCELLED has been sent
<i>cancel- contradiction</i>	Not applicable	Autonomous cancel decision was made, CONFIRM received; CONTRADICTION has not been received
<i>confirm- contradiction</i>	Not applicable	Autonomous confirm decision was made, CANCEL received; CONTRADICTION has not been received
<i>Hazard</i>	A hazard has been reported from at least one Inferior	A hazard has been discovered; CONTRADICTION has not been received
<i>Contradicted</i>	Not applicable	CONTRADICTION has been received
<i>Unknown</i>	No state information for the target-identifier exists	No state information for the target-identifier exists
<i>Inaccessible</i>	There may be state information for this target-identifier but it cannot be reached/existence cannot be determined	There may be state information for this target-identifier but it cannot be reached/existence cannot be determined

2182 **qualifiers** standardised or other qualifiers.

2183 **target-address** the address to which the STATUS is sent. This will be the “reply-address”

2184 on the REQUEST_STATUS message

2185 Types of FAULT possible

2186 ***General***

2187 **FAULT**

2188 Sent in reply to various messages to report an error condition . The FAULT message is used on

2189 all the relationships as a general negative reply to a message.

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

2190

2191 **superior-identifier** the “superior-identifier” as on the CONTEXT message and as used on

2192 the ENROL message (present only if the FAULT is sent to the superior).

2193 **inferior-identifier** the “inferior-identifier” as on the ENROL message (present only if the

2194 FAULT is sent to the inferior)

2195 **fault-type** identifies the nature of the error, as specified for each of the main messages.

2196 **fault-data** information relevant to the particular error. Each “fault-type” defines the

2197 content of the “fault-data”:

fault-type	meaning	fault-data
<i>CommunicationFailure</i>	Any fault arising from the carrier mechanism and communication infrastructure.	Determined by the carrier mechanism and binding specification
<i>DuplicateInferior</i>	An inferior with the same address and identifier is already enrolled with this Superior	The identifier
<i>General</i>	Any otherwise unspecified problem	None
<i>InvalidDecider</i>	The address the message was sent to is not valid (at all or for this Terminator and transaction identifier)	The address
<i>InvalidInferior</i>	The "inferior-identifier" in the message or at least one "inferior-identifier"s in an "inferior-list" parameter is not known or does not identify a known Inferior	One or more invalid identifiers
<i>InvalidSuperior</i>	The received identifier is not known or does not identify a known Superior	The identifier
<i>StatusRefused</i>	The receiver will not report the requested status (or inferior statuses) to this StatusRequestor	None
<i>InvalidTerminator</i>	The address the message was sent to is not valid (at all or for this Decider and transaction identifier)	The address
<i>UnknownParameter</i>	A BTP message has been received with an unrecognised parameter	None
<i>UnknownTransaction</i>	The transaction-identifier is unknown	The transaction-identifier
<i>UnsupportedQualifier</i>	A qualifier has been received that is not recognised and on which "must-be-Understood" is "true".	Qualifier group and name
<i>WrongState</i>	The message has arrived when the recipient or the transaction identified by a related CONTEXT is in an invalid state.	None
<i>Redirect</i>	The target of the BTP message now has a different address	Set of BTP addresses, to be used instead of the address the BTP message was received on

2198

2199 **fault-text** Free text describing the fault or providing more information. Whether this
2200 parameter is present, and exactly what it contains are an implementation option.

2201 **qualifiers** standardised or other qualifiers.

2202 **target-address** the address to which the FAULT is sent. This may be the “reply-address”
2203 from a received message or the address of the opposite side (superior/inferior) as given
2204 in a CONTEXT or ENROL message

2205 *Note – If the carrier mechanism used for the transmission of BTP messages is capable of*
2206 *delivering messages in a different order than they were sent in, the “WrongState”*
2207 *FAULT is not sent and should be ignored if received.*

2208 **REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES**

2209 REQUEST_INFERIOR_STATUSES may be sent to and INFERIOR_STATUSES sent from any
2210 Decider, Superior or Inferior, asking it to report on the status of its relationships with Inferiors (if
2211 any). Since Deciders are required to respond to REQUEST_INFERIOR_STATUSES with
2212 INFERIOR_STATUSES but non-Deciders may just issue FAULT(StatusRefused), and
2213 INFERIOR_STATUSES is also used as a reply to other messages from Terminator to Decider,
2214 these messages are described below under the messages used in the control relationships.

2215 **Messages used in the outcome relationships**

2216 **ENROL**

2217 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a
2218 CONTEXT message in relation to an application request.

2219 The actor issuing ENROL plays the role of Enroller.

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

2220

2221 **superior-identifier.** The “superior-identifier” as on the CONTEXT message

2222 **response- requested** true if an ENROLLED response is required, false otherwise. Default
2223 is false.

2224 **inferior-address** the address to which PREPARE, CONFIRM, CANCEL and
2225 SUPERIOR_STATE messages for this Inferior are to be sent.

2226 **inferior-identifier** an identifier that identifies this Inferior. This shall be globally
2227 unambiguous..

2228 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior name” may be
2229 present.

2230 **target-address** the address to which the ENROL is sent. This will be the “superior-
2231 address” from the CONTEXT message.

2232 **reply-address** the address to which a replying ENROLLED is to be sent, if “response-
2233 requested” is true. If this field is absent and “response-requested” is true, the
2234 ENROLLED should be sent to the “inferior-address” (or one of them, at sender’s
2235 option)

2236 Types of FAULT possible (sent to “reply-address”)

2237 **General**

2238 **InvalidSuperior** – if “superior-identifier” is unknown

2239 **Redirect** – if the Superior now has a different superior-address

2240 **DuplicateInferior** – if inferior with at least one of the set “inferior-address” the same and
2241 the same “inferior-identifier” is already enrolled

2242 **WrongState** – if it is too late to enrol new Inferiors (generally if the Superior has already
2243 sent a PREPARED message to its superior or terminator, or if it has already issued
2244 CONFIRM to other Inferiors).

2245 The form ENROL/rsp-req refers to an ENROL message with “response-requested” having the
2246 value “true”; ENROL/no-rsp-req refers to an ENROL message with “response-requested” having
2247 the value “false”

2248 ENROL/no-rsp-req is typically sent in relation to CONTEXT_REPLY/related. ENROL/rsp-req is
2249 typically when CONTEXT_REPLY/completed will be used (after the ENROLLED message has
2250 been received.)

2251 **ENROLLED**

2252 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been
2253 successfully enrolled (and will therefore be included in the termination exchanges)

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

2254

2255 **inferior-identifier** The “inferior-identifier” as on the ENROL message

2256 **qualifiers** standardised or other qualifiers.

2257 **target-address** the address to which the ENROLLED is sent. This will be the “reply-
 2258 address” from the ENROL message (or one of the “inferior-address”s if the “reply-
 2259 address” was empty)

2260 **sender-address** the address from which the ENROLLED is sent. This is an address of the
 2261 Superior.

2262 No FAULT messages are issued on receiving ENROLLED.

2263 **RESIGN**

2264 Sent from an enrolled Inferior to the Superior to remove the Inferior from the enrolment. This can
 2265 only be sent if the operations of the business transaction have had no effect as perceived by the
 2266 Inferior.

2267 RESIGN may be sent at any time prior to the sending of a PREPARED or CANCELLED
 2268 message (which cannot then be sent). RESIGN may be sent in response to a PREPARE message.

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

2269

2270 **superior-identifier** The “superior-identifier” as on the ENROL message

2271 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message

2272 **response-requested** is set to “true” if a RESIGNED response is required. Default is
 2273 “false”.

2274 **qualifiers** standardised or other qualifiers.

2275 **target-address** the address to which the RESIGN is sent. This will be the superior address
 2276 as used on the ENROL message.

2277 **sender-address** the address from which the RESIGN is sent. This is an address of the
 2278 Inferior.

2279 *Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued*
 2280 *early.*

2281 Types of FAULT possible (sent to “sender-address”)

2282 **General**

2283 **InvalidSuperior** – if “superior-identifier” is unknown

2284 **InvalidInferior** – if no ENROL had been received for this “inferior-identifier”inferior-

2285 **WrongState** – if a PREPARED or CANCELLED has already been received by the
2286 Superior from this Inferior

2287 The form RESIGN/rsp-req refers to an RESIGN message with “response-requested” having the
2288 value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “response-requested”
2289 having the value “false”

2290 **RESIGNED**

2291 Sent in reply to a RESIGN/rsp-req message.

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

2292

2293 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this
2294 Inferior.

2295 **qualifiers** standardised or other qualifiers.

2296 **target-address** the address to which the RESIGNED is sent. This will be the “inferior-
2297 address” from the ENROL message.

2298 **sender-address** the address from which the RESIGNED is sent. This is an address of the
2299 Superior.

2300 After receiving this message the Inferior will not receive any more messages with this “inferior-
2301 identifier”.

2302 Types of FAULT possible (sent to “sender-address”)

2303 **General**

2304 **WrongState** - if RESIGN has not been sent

2305 **PREPARE**

2306 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor RESIGN
2307 have been received, requesting a PREPARED message. PREPARE can be sent after receiving a
2308 PREPARED message.

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

2309

2310 **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.

2311 **qualifiers** standardised or other qualifiers. The standard qualifier “Minimum~~al~~ inferior
2312 timeout” is carried by PREPARE.

2313 **target-address** the address to which the PREPARE message is sent. This will be the
2314 “inferior-address” from the ENROL message.

2315 **sender-address** the address from which the PREPARE is sent. This is an address of the
2316 Superior.

2317 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or
2318 RESIGN.

2319 Types of FAULT possible (sent to “sender-address”)

2320 ***General***

2321 ***InvalidInferior*** – if “inferior-identifier” is unknown,~~or an inferior handle on the inferiors-~~
2322 ~~list is unknown~~

2323 ***WrongState*** – if a CONFIRM or CANCEL has already been received by this Inferior.

2324 **PREPARED**

2325 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when the
2326 Inferior has determined the operations associated with the Inferior can be confirmed and can be
2327 cancelled, as may be instructed by the Superior. The level of isolation is a local matter (i.e. it is
2328 the Inferiors choice, as constrained by the shared understanding of the application exchanges) –
2329 other access may be blocked, may see applied results of operations or may see the original state.

Parameter	Type
superior-identifier	Identifier

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

2330

2331

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

2332

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

2333

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.

2334

2335

2336

2337

2338

2339

2340

2341

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

2342

2343

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

2344

2345

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

2346

2347 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.

2348

2349 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.

2350

2351

2352 Types of FAULT possible (sent to “sender-address”)

2353

General

2354

InvalidSuperior – if “superior-identifier” is unknown

2355

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

2356

2357 The form PREPARED/cancel refers to a PREPARED message with “default-is cancel” = “true”.
2358 The unqualified form PREPARED refers to a PREPARED message with “default-is cancel” =
2359 “false”.

2360 CONFIRM

2361 Sent by the Superior to an Inferior from whom PREPARED has been received.

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

2362

2363 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this
2364 Inferior.

2365 **qualifiers** standardised or other qualifiers.

2366 **target-address** the address to which the CONFIRM message is sent. This will be the
2367 “inferior-address” from the ENROL message.

2368 **sender-address** the address from which the CONFIRM is sent. This is an address of the
2369 Superior.

2370 On receiving CONFIRM, the Inferior is released from its promise to be able to undo the
2371 operations of associated with the Inferior. The effects of the operations can be made available to
2372 everyone (if they weren’t already).

2373 Types of FAULT possible (sent to “sender-address”)

2374 **General**

2375 **InvalidInferior** – if “inferior-identifier” is unknown

2376 **WrongState** – if no PREPARED has been sent by, or if CANCEL has been received by
2377 this Inferior.

2378 CONFIRMED

2379 Sent after the Inferior has applied the confirmation, both in reply to CONFIRM or when the
2380 Inferior has made an autonomous confirm decision, and in reply to a CONFIRM_ONE_PHASE if
2381 the Inferior decides to confirm its associated operations.

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

2382

2383 **superior-identifier** the “superior-identifier” as on the CONTEXT message.

2384 **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.

2385 **confirm-received** “true” if CONFIRMED is sent after receiving a CONFIRM message;
 2386 “false” if an autonomous confirm decision has been made and either if no CONFIRM
 2387 message has been received or the implementation cannot determine if CONFIRM has
 2388 been received (due to loss of state information in a failure).

2389 **qualifiers** standardised or other qualifiers.

2390 **target-address** the address to which the CONFIRMED is sent. This will be the Superior
 2391 address as on the CONTEXT message.

2392 **sender-address** the address from which the CONFIRMED is sent. This is an address of
 2393 the Inferior.

2394 Types of FAULT possible (sent to “sender-address”)

2395 *General*

2396 *InvalidSuperior* – if “superior-identifier” is unknown

2397 *InvalidInferior* – if no ENROL has been received for this “inferior-identifier”, or if
 2398 RESIGN has been received from this Inferior.

2399 *Note – A CONFIRMED message arriving before a CONFIRM message is sent, or after a*
 2400 *CANCEL has been sent will occur when the Inferior has taken an autonomous*
 2401 *decision and is not regarded as occurring in the wrong state. (The latter will cause a*
 2402 *CONTRADICTION message to be sent.)*

2403 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm-received” =
 2404 “false”; CONFIRMED/response refers to a CONFIRMED message with “confirm-received” =
 2405 “true”.

2406 **CANCEL**

2407 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.

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

2408

2409 **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.

2410 **qualifiers** standardised or other qualifiers.

2411 **target-address** the address to which the CANCEL message is sent. This will be the
2412 “inferior-address” from the ENROL message.

2413 **sender-address** the address from which the CANCEL is sent. This is an address of the
2414 Superior.

2415 When received by an Inferior, the effects of any operations associated with the Inferior should be
2416 undone. If the Inferior had sent PREPARED, the Inferior is released from its promise to be able to
2417 confirm the operations.

2418 Types of FAULT possible (sent to “sender-address”)

2419 ***General***

2420 ***InvalidInferior*** – if “inferior-identifier” is unknown, ~~or an inferior handle on the inferiors-~~
2421 ~~list is unknown~~

2422 ***WrongState*** – if a CONFIRM has been received by this Inferior.

2423 **CANCELLED**

2424 Sent when the Inferior has applied (or is applying) cancellation of the operations associated with
2425 the Inferior. CANCELLED is sent from Inferior to Superior in the following cases:

- 2426 1. before (and instead of) sending PREPARED, to indicate the Inferior is unable to
2427 apply the operations in full and is cancelling all of them;
- 2428 2. in reply to CANCEL, regardless of whether PREPARED has been sent;
- 2429 3. after sending PREPARED and then making and applying an autonomous
2430 decision to cancel.
- 2431 4. in reply to CONFIRM_ONE_PHASE if the Inferior decides to cancel the
2432 associated operations

2433 As is specified in the state tables, cases 1, 2 and 3 are not always distinct in some circumstances
2434 of recovery and resending of messages.

Parameter

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

2435

2436 **superior-identifier** the “superior-identifier” as on the CONTEXT message.

2437 **inferior-identifier** the inferior identifier as on the earlier ENROL message.

2438 **qualifiers** standardised or other qualifiers.

2439 **target-address** the address to which the CANCELLED is sent. This will be the Superior
2440 address as on the CONTEXT message.

2441 **sender-address** the address from which the CANCELLED is sent. This is an address of
2442 the Inferior.

2443 Types of FAULT possible (sent to “sender-address”)

2444 **General**

2445 **InvalidSuperior** – if “superior-identifier” is unknown

2446 **InvalidInferior** – if no ENROL has been received for this “inferior-identifier”, or if
2447 RESIGN has been received from this Inferior

2448 **WrongState** – if CONFIRM has been sent

2449 *Note – A CANCELLED message arriving before a CANCEL message is sent, or after a*
2450 *CONFIRM has been sent will occur when the Inferior has taken an autonomous*
2451 *decision and is not regarded as occurring in the wrong state. (The latter will cause a*
2452 *CONTRADICTION message to be sent.)*

2453 **CONFIRM_ONE_PHASE**

2454 Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In this
2455 case the two-phase exchange is not performed between the Superior and Inferior and the outcome
2456 decision for the operations associated with the Inferior is determined by the Inferior.

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

2457

2458 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this
2459 Inferior.

2460 **report hazard** Defines whether the superior wishes to be informed if a mixed condition
2461 occurs for the operations associated with the Inferior. If “report-hazard” is “true”, the
2462 Inferior will reply with HAZARD if a mixed condition occurs, or if the Inferior cannot
2463 determine that a mixed condition has not occurred. If “report-hazard” is false, the
2464 Inferior will report only its own decision, regardless of whether that decision was
2465 correctly and consistently applied. Default is false.

2466 **qualifiers** standardised or other qualifiers.

2467 **target-address** the address to which the CONFIRM_ONE_PHASE message is sent This
2468 will be the “inferior-address” on the ENROL message.

2469 **sender-address** the address from which the CONFIRM_ONE_PHASE is sent. This is an
2470 address of the Superior.

2471 CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom PREPARED
2472 has been received (subject to the requirement that there is only one enrolled Inferior).

2473 Types of FAULT possible (sent to “sender-address”)

2474 ***General***

2475 ***InvalidInferior*** – if “inferior-identifier” is unknown

2476 ***WrongState*** – if a PREPARE has already been sent to this Inferior

2477 **HAZARD**

2478 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly and
2479 consistently cancel or confirm the operations in accord with the decision , or when the Inferior is
2480 unable to determine that a “mixed” condition has not occurred.

2481 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there is a
2482 mixed condition within its associated operations or is unable to determine that there is not a
2483 mixed condition.

2484 *Note - If the Inferior makes its own autonomous decision then it signals that decision with*
 2485 *CONFIRMED or CANCELLED and waits to receive a confirmatory CONFIRM or*
 2486 *CANCEL, or a CONTRADICTION if the autonomous decision by the Inferior was*
 2487 *the opposite of that made by the Superior.*

2488

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

2489

2490 **superior-identifier** The “superior-identifier” as on the ENROL message

2491 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message

2492 **level** indicates, with value “mixed” that a mixed condition has definitely occurred; or, with
 2493 value “possible” that it is unable to determine whether a mixed condition has occurred
 2494 or not.

2495 **qualifiers** standardised or other qualifiers.

2496 **target-address** the address to which the HAZARD is sent. This will be the superior
 2497 address from the ENROL message.

2498 **sender-address** the address from which the HAZARD is sent. This is an address of the
 2499 Inferior.

2500 Types of FAULT possible (sent to “sender-address”)

2501 ***General***

2502 ***InvalidSuperior*** – if “superior-identifier” is unknown

2503 ***InvalidInferior*** – if no ENROL has been received for this “inferior-identifier”, or if
 2504 RESIGN has been received from this Inferior

2505 The form HAZARD/mixed refers to a HAZARD message with “level” = “mixed”, the form
 2506 HAZARD/possible refers to a HAZARD message with “level” = “possible”.

2507 **CONTRADICTION**

2508 Sent by the Superior to an Inferior that has taken an autonomous decision contrary to the decision
2509 for the atom. This is detected by the Superior when the ‘wrong’ one of CONFIRMED or
2510 CANCELLED is received. CONTRADICTION is also sent in response to a HAZARD message.

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

2511

2512 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this
2513 Inferior.

2514 **qualifiers** standardised or other qualifiers.

2515 **target-address** the address to which the CONTRADICTION message is sent. This will be
2516 the “inferior-address” from the ENROL message.

2517 **sender-address** the address from which the CONTRADICTION is sent. This is an
2518 address of the Superior.

2519 Types of FAULT possible (sent to “sender-address”)

2520 ***General***

2521 ***InvalidInferior*** – if “inferior-identifier” is unknown

2522 ***WrongState*** – if neither CONFIRMED or CANCELLED has been sent by this Inferior

2523 **SUPERIOR_STATE**

2524 Sent by a Superior as a query to an Inferior when

2525 1. in the active state

2526 2. there is uncertainty what state the Inferior has reached (due to recovery from
2527 previous failure or other reason).

2528 Also sent by the Superior to the Inferior in response to a received INFERIOR_STATE, in
2529 particular states.

Parameter	Type
inferior-identifier	Identifier

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

2530

2531 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for this
2532 Inferior.

2533 **status** states the current state of the Superior, in terms of its relation to this Inferior only.

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

2534

2535 **response-requested** true, if SUPERIOR_STATE is sent as a query at the Superior’s
2536 initiative; false, if SUPERIOR_STATE is sent in reply to a received
2537 INFERIOR_STATE or other message. Can only be true if status is active or prepared-
2538 received. Default is “false”

2539 **qualifiers** standardised or other qualifiers.

2540 **target-address** the address to which the SUPERIOR_STATE message is sent. This will
2541 be the “inferior-address” from the ENROL message.

2542 **sender-address** the address from which the SUPERIOR_STATE is sent. This is an
 2543 address of the Superior.

2544 The Inferior, on receiving SUPERIOR_STATE with “response-requested = true, should reply in a
 2545 timely manner by (depending on its state) repeating the previous message it sent or by sending
 2546 INFERIOR_STATE with the appropriate status value.

2547 A status of unknown shall only be sent if it has been determined for certain that the Superior has
 2548 no knowledge of the Inferior, or (equivalently) it can be determined that the relationship with the
 2549 Inferior was cancelled. If there could be persistent information corresponding to the Superior, but
 2550 it is not accessible from the entity receiving an INFERIOR_STATE/*y (or other) message
 2551 targeted to the Superior or that entity cannot determine whether any such persistent information
 2552 exists or not, the response shall be Inaccessible.

2553 SUPERIOR_STATE/unknown is also used as a response to messages, other than
 2554 INFERIOR_STATE/*y that are received when the Inferior is not known (and it is known there is
 2555 no state information for it).

2556 The form SUPERIOR_STATE/abcd refers to a SUPERIOR_STATE message status having a
 2557 value equivalent to “abcd” (for active, prepared-received, unknown and inaccessible) and with
 2558 “response-requested” = “false”. SUPERIOR_STATE/abcd/y refers to a similar message, but with
 2559 “response-requested” = “true”. The form SUPERIOR_STATE/*y refers to a
 2560 SUPERIOR_STATE message with “response-requested” = “true” and any value for status.

2561 **INFERIOR_STATE**

2562 Sent by an Inferior as a query when in the active state to a Superior, when (due recovery from
 2563 previous failure or other reason) there is uncertainty what state the Superior has reached.

2564 Also sent by the Inferior to the Superior in response to a received SUPERIOR_STATE, in
 2565 particular states.

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

2566

2567 **superior-identifier** The “superior-identifier” as used on the ENROL message

2568 **inferior-identifier** The “inferior-identifier” as on the ENROL message

2569 **status** states the current state of the Inferior for the atomic business transaction, which
 2570 corresponds to the last message sent to the Superior by (or in the case of ENROL for)
 2571 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

2572

2573 **response-requested** “true” if INFERIOR_STATE is sent as a query at the Superior’s
 2574 initiative; “false” if INFERIOR_STATE is sent in reply to a received
 2575 SUPERIOR_STATE or other message. Can only be “true” if “status” is “active” or
 2576 “prepared-received”. Default is “false”

2577 **qualifiers** standardised or other qualifiers.

2578 **target-address** the address to which the INFERIOR_STATE is sent. This will be the
 2579 “target-address” as used the original ENROL message.

2580 **sender-address** the address from which the INFERIOR_STATE is sent. This is an
 2581 address of the Inferior.

2582 The Superior, on receiving INFERIOR_STATE with “response-requested” = “true”, should reply
 2583 in a timely manner by (depending on its state) repeating the previous message it sent or by
 2584 sending SUPERIOR_STATE with the appropriate status value.

2585 A status of “unknown” shall only be sent if it has been determined for certain that the Inferior has
 2586 no knowledge of a relationship with the Superior. If there could be persistent information
 2587 corresponding to the Superior, but it is not accessible from the entity receiving an
 2588 SUPERIOR_STATE/*y (or other) message targetted on the Inferior or the entity cannot
 2589 determine whether any such persistent information exists, the response shall be “inaccessible”.

2590 INFERIOR_STATE/unknown is also used as a response to messages, other than
 2591 SUPERIOR_STATE/*y that are received when the Inferior is not known (and it is known there
 2592 is no state information for it).

2593 A SUPERIOR_STATE/INFERIOR_STATE exchange that determines that one or both sides are
 2594 in the active state does not require that the Inferior be cancelled (unlike some other two-phase

2595 commit protocols). The relationship between Superior and Inferior, and related application
 2596 elements may be continued, with new application messages carrying the same CONTEXT.
 2597 Similarly, if the Inferior is prepared but the Superior is active, there is no required impact on the
 2598 progression of the relationship between them.

2599 The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a
 2600 value equivalent to “abcd” (for active, unknown and inaccessible) and with “response-requested”
 2601 = “false”. INFERIOR_STATE/abcd/y refers to a similar message, but with “response-requested”
 2602 = “true”. The form INFERIOR_STATE/*y refers to a INFERIOR_STATE message with
 2603 “response-requested” = “true” and any value for status.

2604 REDIRECT

2605 Sent when the address previously given for a Superior or Inferior is no longer valid and the
 2606 relevant state information is now accessible with a different address (but the same superior or
 2607 “inferior-identifier”).

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

2608

2609 **superior-identifier** The “superior-identifier” as on the CONTEXT message and used on an
 2610 ENROL message. (present only if the REDIRECT is sent from the Inferior).

2611 **inferior-identifier** The “inferior-identifier” as on the ENROL message

2612 **old-address** The previous address of the sender of REDIRECT. A match is considered to
 2613 apply if any of the “old-address” values match one that is already known.

2614 **new-address** The (set of alternatives) “new-address” values to be used for messages sent
 2615 to this entity.

2616 **qualifiers** standardised or other qualifiers.

2617 **target-address** the address to which the REDIRECT is sent. This is the address of the
 2618 opposite side (superior/inferior) as given in a CONTEXT or ENROL message

2619 If the actor whose address is changed is an Inferior, the “new-address” value replaces the
 2620 “inferior-address” as present in the ENROL.

2621 If the actor whose address is changed is a Superior, the “new-address” value replaces the Superior
2622 address as present in the CONTEXT message (or as present in any other mechanism used to
2623 establish the Superior:Inferior relationship).

2624 Messages used in control relationships

2625 BEGIN

2626 A request to a Factory to create a new Business Transaction. This may either be a new top-level
2627 transaction, in which case the Composer or Coordinator will be the Decider, or the new Business
2628 Transaction may be immediately made the Inferior within an existing Business Transaction (thus
2629 creating a sub-Composer or sub-Coordinator).

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

2630

2631 **transaction-type** identifies whether a new Cohesion or new Atom is to be created; this
2632 value will be the “superior-type” in the new CONTEXT

2633 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction timelimit”
2634 may be present on BEGIN, to set the timelimit for the new business transaction and
2635 will be copied to the new CONTEXT. The standard qualifier “Inferior name” may be
2636 present if there is a CONTEXT related to the BEGIN.

2637 **target-address** the address of the entity to which the BEGIN is sent. How this address is
2638 acquired and the nature of the entity are outside the scope of this specification.

2639 **reply-address** the address to which the replying BEGUN and related CONTEXT message
2640 should be sent.

2641 A new top-level Business Transaction is created if there is no CONTEXT related to the BEGIN.
2642 A Business Transaction that is to be Inferior in an existing Business Transaction is created if the
2643 CONTEXT message for the existing Business Transaction is related to the BEGIN. In this case,
2644 the Factory is responsible for enrolling the new Composer or Coordinator as an Inferior of the
2645 Superior identified in that CONTEXT.

2646 *Note – This specification does not provide a standardised means to determine which of*
2647 *the Inferiors of a sub-Composer are in its confirm set. This is considered part of the*
2648 *application:inferior relationship.*

2649 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction-type” having the
2650 corresponding value.

2651 Types of FAULT possible (sent to “reply-address”)

2652 **General**

2653 **Redirect** – if the Factory now has a different address

2654 **WrongState** - only issued if there is a related CONTEXT, and the Superior identified by
 2655 the CONTEXT is in the wrong state to enrol new Inferiors

2656 **BEGUN**

2657 BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT for
 2658 the new business transaction.

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

2659

2660 **decider-address** for a top-most transaction (no CONTEXT related to the BEGIN), this is
 2661 the address to which PREPARE_INFERIORS, CONFIRM_TRANSACTION,
 2662 CANCEL_TRANSACTION, CANCEL_INFERIORS and
 2663 REQUEST_INFERIOR_STATUSES messages are to be sent; if a CONTEXT was
 2664 related to the BEGIN this parameter is absent

2665 **inferior-address** for a non-top-most transaction (a CONTEXT was related to the BEGIN),
 2666 this is the “inferior-address” used in the enrolment with the Superior identified by the
 2667 CONTEXT related to the BEGIN. The parameter is optional (implementor’s choice) if
 2668 this is not a top-most transaction; it shall be absent if this is a top-most transaction.

2669 **transaction-identifier** if this is a top-most transaction, this is an globally-unambiguous
 2670 identifier for the new Decider (Composer or Coordinator). If this is not a top-most
 2671 transaction, the transaction-identifier shall be the inferior-identifier used in the
 2672 enrolment with the Superior identified by the CONTEXT related to the BEGIN.

2673 *Note – The “transaction-identifier” may be identical to the “superior-identifier” in*
 2674 *the CONTEXT that is related to the BEGUN*

2675 **qualifiers** standardised or other qualifiers.

2676 **target-address** the address to which the BEGUN is sent. This will be the “reply-address”
 2677 from the BEGIN.

2678 At implementation option, the “decider-address” and/or “inferior-address” and the “superior-
 2679 address” in the related CONTEXT may be the same or may be different. There is no general
 2680 requirement that they even use the same bindings. Any may also be the same as the “target-

2681 address” of the BEGIN message (the identifier on messages will ensure they are applied to the
2682 appropriate Composer or Coordinator).

2683 No FAULT messages are issued on receiving BEGUN.

2684 PREPARE_INFERIORS

2685 Sent from a Terminator to a Decider, but only if it is a Cohesion Composer, to tell it to prepare all
2686 or some of its inferiors, by sending PREPARE to any that have not already sent PREPARED,
2687 RESIGN or CANCELLED to the Decider (Composer) on its relationships as Superior. If the
2688 inferiors-list parameter is absent, the request applies to all the inferiors; if the parameter is
2689 present, it applies only to the identified inferiors of the Decider (Composer).

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

2690

2691 **transaction identifier** identifies the Decider and will be the transaction-identifier from the
2692 BEGUN message.

2693 **inferiors-list** defines which of the Inferiors of this Decider preparation is requested for,
2694 using the “inferior-identifiers” as on the ENROL received by the Decider (in its role as
2695 Superior). If this parameter is absent, the PREPARE applies to all Inferiors.

2696 **qualifiers** standardised or other qualifiers.

2697 **target-address** the address to which the PREPARE_INFERIORS message is sent. This
2698 will be the decider-address from the BEGUN message.

2699 **reply-address** the address of the Terminator sending the PREPARE_INFERIORS
2700 message.

2701 For all Inferiors identified in the inferiors-list parameter (all Inferiors if the parameter is absent),
2702 from which none of PREPARED, CANCELLED or RESIGNED has been received, the Decider
2703 shall issue PREPARE. It will reply to the Terminator, using the “reply-address” on the
2704 PREPARE_INFERIORS message, sending an INFERIOR_STATUSES message giving the status
2705 of the Inferiors identified on the inferiors-list parameter (all of them if the parameter was absent).

2706 If one or more of the “inferior-identifier”s in the “inferior-list” is unknown (does not correspond
2707 to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. It is an implementation
2708 option whether CANCEL is sent to any of the Inferiors that are validly identified in the “inferiors-
2709 list”.

2710 Types of FAULT possible (sent to Superior address)

2711 **General**

2712 **InvalidDecider** – if Decider address is unknown

2713 **Redirect** – if the Decider now has a different “decider-address”

2714 **UnknownTransaction** – if the transaction-identifier is unknown

2715 **InvalidInferior** – if one or more inferior-~~handles~~identifiers on the inferiors-list is unknown

2716 **WrongState** – if a CONFIRM_TRANSACTION or CANCEL_TRANSACTION has
2717 already been received by this Composer.

2718 The form PREPARE_INFERIORS/all refers to a PREPARE_INFERIORS message where the
2719 “inferiors-list” parameter is absent. The form PREPARE_INFERIORS/specific refers to a
2720 PREPARE_INFERIORS message where the “inferiors-list” parameter is present.

2721 CONFIRM_TRANSACTION

2722 Sent from a Terminator to a Decider to request confirmation of the business transaction. If the
2723 business transaction is a Cohesion, the confirm-set is specified by the “inferiors-list” parameter.

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

2724

2725 **transaction-identifier** identifies the Decider. This will be the transaction-identifier from
2726 the BEGUN message.

2727 **inferiors-list** defines which Inferiors enrolled with the Decider, if it is a Cohesion
2728 Composer, are to be confirmed, using the “inferior-identifiers” as on the ENROL
2729 received by the Decider (in its role as Superior). Shall be absent if the Decider is an
2730 Atom Coordinator.

2731 **report-hazard** Defines whether the Terminator wishes to be informed of hazard events and
2732 contradictory decisions within the business transaction. If “report-hazard” is “true”, the
2733 receiver will wait until responses (CONFIRMED, CANCELLED or HAZARD) have
2734 been received from all of its inferiors, ensuring that any hazard events are reported. If
2735 “report-hazard” is “false”, the Decider will reply with

2736 TRANSACTION_CONFIRMED or TRANSACTION_CANCELLED as soon as the
2737 decision for the transaction is known.

2738 **qualifiers** standardised or other qualifiers.

2739 **target-address** the address to which the CONFIRM_TRANSACTION message is sent.
2740 This will be the “decider-address” on the BEGUN message.

2741 **reply-address** the address of the Terminator sending the CONFIRM_TRANSACTION
2742 message.

2743 If the “inferiors-list” parameter is present, the Inferiors identified shall be the “confirm-set” of the
2744 Cohesion. If the parameter is absent and the business transaction is a Cohesion, the “confirm-set”
2745 shall be all remaining Inferiors. If the business transaction is an Atom, the “confirm-set” is
2746 automatically all the Inferiors.

2747 Any Inferiors from which RESIGN is received are not counted in the confirm-set.

2748 If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED has
2749 not been received, PREPARE shall be issued to that Inferior.

2750 *NOTE -- If PREPARE has been sent but PREPARED not yet received from an Inferior in*
2751 *the confirm-set, it is an implementation option whether and when to re-send*
2752 *PREPARE. The Superior implementation may choose to re-send PREPARE if there*
2753 *are indications that the earlier PREPARE was not delivered.*

2754 A confirm decision may be made only if PREPARED has been received from all Inferiors in the
2755 “confirm-set”. The making of the decision shall be persistent (and if it is not possible to persist
2756 the decision, it is not made). If there is only one remaining Inferior in the “confirm set” and
2757 PREPARE has not been sent to it, CONFIRM_ONE_PHASE may be sent to it.

2758 All remaining Inferiors that are not in the confirm set shall be cancelled.

2759 If a confirm decision is made and “report-hazard” was “false”, a
2760 TRANSACTION_CONFIRMED message shall be sent to the “reply-address”.

2761 If a cancel decision is made and “report-hazard” was “false”, a TRANSACTION_CANCELLED
2762 message shall be sent to the “reply-address”.

2763 If "report-hazard" was "true", TRANSACTION_CONFIRMED shall be sent to the "reply-
2764 address" after CONFIRMED has been received from each Inferior in the confirm-set and
2765 CANCELLED or RESIGN from each and any Inferior not in the confirm-set.

2766 If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.
2767 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in the
2768 confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent to the
2769 “reply-address”.

2770 If one or more of the "inferior-identifier"s in the "inferior-list" is unknown (does not correspond
2771 to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. The Decider shall not make a
2772 confirm decision and shall not send CONFIRM to any Inferior.

2773 Types of FAULT possible (sent to “reply-address”)

2774 **General**

2775 **InvalidDecider** – if Decider address is unknown

2776 **Redirect** – if the Decider now has a different “decider-address”

2777 **UnknownTransaction** – if the transaction-identifier is unknown

2778 **InvalidInferior** – if one or more “inferior ~~handles~~ identifiers” in the inferiors-list is
2779 unknown

2780 **WrongState** – if a CANCEL_TRANSACTION has already been received .

2781 The form CONFIRM_TRANSACTION/all refers to a CONFIRM_TRANSACTION message
2782 where the “inferiors-list” parameter is absent. The form CONFIRM_TRANSACTION/specific
2783 refers to a CONFIRM_TRANSACTION message where the “inferiors-list” parameter is present.

2784 **TRANSACTION_CONFIRMED**

2785 A Decider sends TRANSACTION_CONFIRMED to a Terminator in reply to
2786 CONFIRM_TRANSACTION if all of the confirm-set confirms (and, for a Cohesion, all other
2787 Inferiors cancel) without reporting hazards, or if the Decider made a confirm decision and the
2788 CONFIRM_TRANSACTION had a “report-hazards” value of “false”.

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

2789

2790 **transaction-identifier** the “transaction-identifier” as on the BEGUN message (i.e. the
2791 identifier of the Decider as a whole).

2792 **qualifiers** standardised or other qualifiers.

2793 **target-address** the address to which the TRANSACTION_CONFIRMED is sent., this
2794 will be the “reply-address” from the CONFIRM_TRANSACTION message

2795 Types of FAULT possible (sent to “decider-address”)

2796 **General**

2797 **InvalidTerminator** – if Terminator address is unknown

2798 **UnknownTransaction** – if the transaction-identifier is unknown

2799 **CANCEL_TRANSACTION**

2800 Sent by a Terminator to a Decider at any time before CONFIRM_TRANSACTION has been sent.

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

2801

2802 **transaction-identifier** identifies the Decider and will be the transaction-identifier from the
2803 BEGUN message.

2804 **report-hazard** Defines whether the Terminator wishes to be informed of hazard events and
2805 contradictory decisions within the business transaction. If "report-hazard" is "true", the
2806 receiver will wait until responses (CONFIRMED, CANCELLED or HAZARD) have
2807 been received from all of its inferiors, ensuring that any hazard events are reported. If
2808 "report-hazard" is "false", the Decider will reply with
2809 TRANSACTION_CANCELLED immediately.

2810 **qualifiers** standardised or other qualifiers.

2811 **target-address** the address to which the CANCEL_TRANSACTION message is sent.
2812 This will be the decider-address from the BEGUN message.

2813 **reply-address** the address of the Terminator sending the CANCEL_TRANSACTION
2814 message.

2815 The business transaction is cancelled – this is propagated to any remaining Inferiors by issuing
2816 CANCEL to them. No more Inferiors will be permitted to enrol.

2817 If "report-hazard" was "false", a TRANSACTION_CANCELLED message shall be sent to the
2818 "reply-address".

2819 If "report-hazard" was "true" and any HAZARD or CONFIRMED message was received, an
2820 INFERIOR_STATUSES reporting the status for all Inferiors shall be sent to the "reply-address".

2821 If "report-hazard" was "true", TRANSACTION_CANCELLED shall be sent to the "reply-
2822 address" after CANCELLED or RESIGN has been received from each Inferior.

2823 Types of FAULT possible (sent to Superior address)

2824 **General**

2825 **InvalidDecider** – if Decider address is unknown

2826 **Redirect** – if the Decider now has a different "decider-address"

2827 **UnknownTransaction** – if the transaction-identifier is unknown

2828 **WrongState** – if a CONFIRM_TRANSACTION has been received by this Composer.

2829 **CANCEL_INFERIORS**

2830 Sent by a Terminator to a Decider, but only if is a Cohesion Composer, at any time before

2831 CONFIRM_TRANSACTION or CANCEL_TRANSACTION has been sent.

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

2832

2833 **transaction-identifier** identifies the Decider and will be the transaction-identifier from the

2834 BEGUN message.

2835 **inferiors-list** defines which of the Inferiors of this Decider are to be cancelled, using the

2836 "inferior-identifiers" as on the ENROL received by the Decider (in its role as

2837 Superior).

2838 **qualifiers** standardised or other qualifiers.

2839 **target-address** the address to which the CANCEL_TRANSACTION message is sent.

2840 This will be the decider-address from the BEGUN message.

2841 **reply-address** the address of the Terminator sending the CANCEL_TRANSACTION

2842 message.

2843 Only the Inferiors identified in the inferiors-list are to be cancelled. Any other inferiors are

2844 unaffected by a CANCEL_INFERIORS. Further Inferiors may be enrolled.

2845 *Note – A CANCEL_INFERIORS for all of the currently enrolled Inferiors will leave the*

2846 *cohesion 'empty', but permitted to continue with new Inferiors, if any enrol.*

2847 If one or more of the "inferior-identifier"s in the "inferior-list" is unknown (does not correspond

2848 to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. It is an implementation

2849 option whether CANCEL is sent to any of the Inferiors that are validly identified in the "inferiors-

2850 list".

2851 Types of FAULT possible (sent to Superior address)

2852 **General**

2853 **InvalidDecider** – if Decider address is unknown

2854 **Redirect** – if the Decider now has a different “decider-address”

2855 **UnknownTransaction** – if the transaction-identifier is unknown

2856 **InvalidInferior** – if one or more inferior-~~handle~~-identifiers on the inferiors-list is unknown

2857 **WrongState** – if a CONFIRM_TRANSACTION or CANCEL_TRANSACTION has been
2858 received by this Composer.

2859 TRANSACTION_CANCELLED

2860 A Decider sends TRANSACTION_CANCELLED to a Terminator in reply to
2861 CANCEL_TRANSACTION or in reply to CONFIRM_TRANSACTION if the Decider decided
2862 to cancel. In both cases, TRANSACTION_CANCELLED is used only if all Inferiors cancelled
2863 without reporting hazards or the CANCEL_TRANSACTION or CONFIRM_TRANSACTION
2864 had a “report-hazard” value of “false.”

Parameter

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

2865

2866 **transaction-identifier** the “transaction-identifier” as on the BEGUN message (i.e. the
2867 identifier of the Decider as a whole).

2868 **qualifiers** standardised or other qualifiers.

2869 **target-address** the address to which the TRANSACTION_CANCELLED is sent. This
2870 will be the “reply-address” from the CANCEL_TRANSACTION or
2871 CONFIRM_TRANSACTION message.

2872 Types of FAULT possible (sent to “decider-address”)

2873 **General**

2874 **InvalidTerminator** – if Terminator address is unknown

2875 **UnknownTransaction** – if the transaction-identifier is unknown

2876 REQUEST_INFERIOR_STATUSES

2877 Sent to a Decider to ask it to report the status of its Inferiors with an INFERIOR_STATUSES
 2878 message. It can also be sent to any actor with a “superior-address” or “inferior-address”, asking it
 2879 about the status of that transaction tree nodes Inferiors, if there are any. In this latter case, the
 2880 receiver may reject the request with a FAULT(StatusRefused). If it is prepared to reply, but has
 2881 no Inferiors, it replies with an INFERIOR_STATUSES with an empty “status-list” parameter.

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

2882

2883 **target-identifier** identifies the transaction (or transaction tree node). When the message is
 2884 used to a Decider, this will be the transaction-identifier from the BEGUN message.
 2885 Otherwise it will be the superior-identifier from a CONTEXT or an inferior-identifier
 2886 from an ENROL message.

2887 **inferiors-list** defines which inferiors enrolled with the target are to be included in the
 2888 INFERIOR_STATUSES, using the “inferior-identifiers” as on the ENROL received
 2889 by the Decider (in its role as Superior). If the list is absent, the status of all enrolled
 2890 Inferiors will be reported.

2891 **qualifiers** standardised or other qualifiers.

2892 **target-address** the address to which the REQUEST_STATUS message is sent. When
 2893 used to a Decider, this will be the “decider-address” from the BEGUN message.
 2894 Otherwise it may be a “superior-address” from a CONTEXT or “inferior-address”
 2895 from an ENROL message.

2896 **reply-address** the address to which the replying INFERIOR_STATUSES is to be sent

2897 Types of FAULT possible (sent to reply-address)

2898 **General**

2899 **Redirect** – if the intended target now has a different address

2900 **StatusRefused** – if the receiver is not prepared to report its status to the sender of this
 2901 message. This “fault-type” shall not be issued when a Decider receives
 2902 REQUEST_STATUSES from the Terminator.

2903 **UnknownTransaction** – if the transaction-identifier is unknown

2904 The form REQUEST_INFERIOR_STATUSES/all refers to a REQUEST_STATUS with the
2905 inferiors-list absent. The form REQUEST_INFERIOR_STATUS/specific refers to a
2906 REQUEST_INFERIOR_STATUS with the inferiors-list present.

2907 INFERIOR_STATUSES

2908 Sent by a Decider to report the status of all or some of its inferiors in response to a
2909 REQUEST_INFERIOR_STATUSES, PREPARE_INFERIORS, CANCEL_INFERIORS,
2910 CANCEL_TRANSACTION with “report-hazard” value of “true” and
2911 CONFIRM_TRANSACTION with “report-hazard” value of “true”. It is also used by any actor in
2912 response to a received REQUEST_INFERIOR_STATUSES to report the status of inferiors, if
2913 there are any.

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

2914

2915 **responders-identifier** the target-identifier used on the
2916 REQUEST_INFERIOR_STATUSES.

2917 **status-list** contains a number of Status-items, each reporting the status of one of the
2918 inferiors of the Decider. The fields of a Status-item are

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

2919

2920 The status value reports the current status of the particular inferior, as known to the Decider
2921 (Composer or Coordinator). Values are:

status value	Meaning
<i>active</i>	The Inferior is enrolled
<i>resigned</i>	RESIGNED has been received from the Inferior

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

2922

2923 **general-qualifiers** standardised or other qualifiers applying to the
2924 INFERIOR_STATUSES as a whole. Each Status-item contains a “qualifiers” field
2925 containing qualifiers applying to (and received from) the particular Inferior.

2926 **target-address** the address to which the INFERIOR_STATUSES is sent. This will be the
2927 “reply-address” on the received message

2928 If the inferiors-list parameter was present on the received message, only the inferiors identified by
2929 that parameter shall have their status reported in status-list of this message. If the inferiors-list
2930 parameter was absent, the status of all enrolled inferiors shall be reported, except that an inferior
2931 that had been reported as *cancelled* or *resigned* on a previous INFERIOR_STATUSES message
2932 **may** be omitted (sender’s option).

2933 Types of FAULT possible (sent to “decider-address”)

2934 **General**

2935 **InvalidTerminator** – if Terminator address is unknown

2936 **UnknownTransaction** – if the transaction-identifier is unknown

2937 **Groups – combinations of related messages**

2938 The following combinations of messages form related groups, for which the meaning of the group
2939 is not just the aggregate of the meanings of the messages. The “&” notation is used to indicate
2940 relatedness. Messages appearing in parentheses in the names of groups in this section indicate
2941 messages that may or may not be present. The notation A & B / & C in a group name in this
2942 section indicates a group that contains A and B or A and C or A, B and C, possibly with any of
2943 those appearing more than once.

2944 **CONTEXT & application message**

2945 **Meaning:** the transmission of the application message is deemed to be part of the
2946 business transaction identified by the CONTEXT. The exact effect of this for application
2947 work implied by the transmission of the message is determined by the application – in
2948 many cases, it will mean the effects of the application message are to be subject to the
2949 outcome delivered to an enrolled Inferior, thus requiring the enrolment of a new Inferior
2950 if no appropriate Inferior is enrolled or if the CONTEXT is for cohesion.

2951 **target-address:** the “target-address” is that of the application message. It is not required
2952 that the application address be a BTP address (in particular, there is no BTP-defined
2953 “additional information” field – the application protocol (and its binding) may or may not
2954 have a similar construct).

2955 There may be multiple application messages related to a single CONTEXT message. All
2956 the application messages so related are deemed to be part of the business transaction
2957 identified by the CONTEXT. This specification does not imply any further relatedness
2958 among the application messages themselves (though the application might).

2959 The actor that sends the group shall retain knowledge of the Superior address in the
2960 CONTEXT. If the CONTEXT is a CONTEXT/atom, the actor shall also keep track of
2961 transmitted CONTEXTs for which no CONTEXT_REPLY has been received.

2962 If the CONTEXT is a CONTEXT/atom, the actor receiving the CONTEXT shall ensure
2963 that a CONTEXT_REPLY message is sent back to the “reply-address” of the CONTEXT
2964 with the appropriate completion status.

2965 *Note – The representation of the relation between CONTEXT and one or more*
2966 *application messages depends on the binding to the carrier protocol. It is not*
2967 *necessary that the CONTEXT and application messages be closely associated “on*
2968 *the wire” (or even sent on the same connection) – some kind of referencing*
2969 *mechanism may be used.*

2970 **CONTEXT_REPLY & ENROL**

2971 **Meaning:** the enrolment of the Inferior identified in the ENROL is to be performed with
2972 the Superior identified in the CONTEXT message this CONTEXT_REPLY is replying
2973 to. If the “completion-status” of CONTEXT_REPLY is “related”, failure of this
2974 enrolment shall prevent the confirmation of the business transaction.

2975 **target-address:** the “target-address” is that of the CONTEXT_REPLY. This will be the
2976 “reply-address” of the CONTEXT message (in many cases, including request/reply
2977 application exchanges, this address will usually be implicit).

2978 The “target-address” of the ENROL message is omitted.

2979 The actor receiving the related group will use the retained Superior address from the
2980 CONTEXT sent earlier to forward the ENROL. When doing so, it changes the ENROL to
2981 ask for a response (if it was an ENROL/no-rsp-req) and supplies its own address as the
2982 “reply-address”, remembering the original “reply-address” if there was one.

2983 If ENROLLED is received and the original received ENROL was ENROL/rsp-req, the
2984 ENROLLED is forwarded back to the original “reply-address”.

2985 If this attempt fails (i.e. ENROLLED is not received), and the “completion-status” of the
2986 CONTEXT_REPLY was “related”, the actor is required to ensure that the Superior does
2987 not proceed to confirmation. How this is achieved is an implementation option, but must
2988 take account of the possibility that direct communication with the Superior may fail. (One
2989 method is to prevent CONFIRM_TRANSACTION being sent to the Superior (in its role
2990 as Decider); another is to enrol as another Inferior before sending the original CONTEXT
2991 out with an application message). If the Superior is a sub-coordinator or sub-composer,
2992 an enrolment failure must ensure the sub-coordinator does not send PREPARED to its
2993 own Superior.

2994 If the actor receiving the related group is also the Superior (i.e. it has the same binding
2995 address), the explicit forwarding of the ENROL is not required, but the resultant effect –
2996 that if enrolment fails the Superior does not confirm or issue PREPARED – shall be the
2997 same.

2998 A CONTEXT_REPLY & ENROL group may contain multiple ENROL messages, for
2999 several Inferiors. Each ENROL shall be forwarded and an ENROLLED reply received
3000 before the Superior is allowed to confirm if the “completion-status” in the
3001 CONTEXT_REPLY was “related”.

3002 When the group is constructed, if the CONTEXT had “superior-type” value of “atom”,
3003 the “completion-status” of the CONTEXT_REPLY shall be “related”. If the “superior-
3004 type” was “cohesive”, the “completion-status” shall be “incomplete” or “related” (as
3005 required by the application). If the value is “incomplete”, the actor receiving the group
3006 shall forward the ENROLs, but is not required to prevent confirmation (though it may do
3007 so).

3008 **CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED**

3009 This combination is characterised by a related CONTEXT_REPLY and either or both of
3010 PREPARED and CANCELLED, with or without ENROL.

3011 **Meaning:** If ENROL is present, the meaning and required processing is the same as for
3012 CONTEXT_REPLY & ENROL. The PREPARED or CANCELLED message(s) are
3013 forwarded to the Superior identified in the CONTEXT message this CONTEXT_REPLY
3014 is replying to.

3015 *Note – the combination of CONTEXT_REPLY & ENROL & CANCELLED may be used*
3016 *to force cancellation of an atom*

3017 **target-address:** the “target-address” is that of the CONTEXT_REPLY. This will be the
3018 “reply-address” of the CONTEXT message (in many cases, including request/reply
3019 application exchanges, this address will usually be implicit).

3020 The “target-address” of the PREPARED and CANCELLED message is omitted – they
3021 will be sent to the Superior identified in the earlier CONTEXT message.

3022 The actor receiving the group forwards the PREPARED or CANCELLED message to the
3023 Superior in as for an ENROL, using the retained Superior address from the CONTEXT
3024 sent earlier, except there is no reply required from the Superior.

3025 If (as is usual) an ENROL and PREPARED or CANCELLED message are for the same
3026 Inferior, the ENROL shall be sent first, but the actor need not wait for the ENROLLED to
3027 come back before sending the PREPARED or CANCELLED (so an
3028 ENROL+PREPARED bundle from this actor to the Superior could be used).

3029 The group can contain multiple ENROL, PREPARED and CANCELLED messages.
3030 Each PREPARED and CANCELLED message will be for a different Inferior.. There is
3031 no constraint on the order of their forwarding, except that ENROL and PREPARED or
3032 CANCELLED for the same Inferior shall be delivered to the Superior in the order
3033 ENROL first, followed by the other message for that Inferior.

3034 **CONTEXT_REPLY & ENROL & application message (& PREPARED)**

3035 This combination is characterised by a related CONTEXT_REPLY, ENROL and an application
3036 message. PREPARED may or may not be present in the related group.

3037 **Meaning:** the relation between the BTP messages is as for the preceding groups, The
3038 transmission of the application message (and application effects implied by its
3039 transmission) has been associated with the Inferior identified by the ENROL and will be
3040 subject to the outcome delivered to that Inferior.

3041 **target-address:** the “target-address” of the group is the “target-address” of the
3042 CONTEXT_REPLY which shall also be the “target-address” of the application message.
3043 The ENROL and PREPARED messages do not contain their “target-address” parameters.

3044 The processing of ENROL and PREPARED messages is the same as for the previous
3045 groups.

3046 This group can be used when participation in business transaction (normally a cohesion),
3047 is initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with
3048 some associated application semantic, performs some work for the transaction and sends
3049 an application message with a related ENROL. The CONTEXT_REPLY allows the
3050 addressing of the application (and the CONTEXT_REPLY) to be distinct from that of the
3051 Superior.

3052 The actor receiving the group may associate the “inferior-identifier” received on the
3053 ENROL with the application message in a manner that is visible to the application
3054 receiving the message (e.g. for subsequent use in Terminator:Decider exchanges).

3055 **BEGUN & CONTEXT**

3056 **Meaning:** the CONTEXT is that for the new business transaction, containing the
3057 Superior address.

3058 **target-address:** the “target-address” is that of the BEGUN message – this will be the
3059 “reply-address” of the earlier BEGIN message.

3060 **BEGIN & CONTEXT**

3061 **Meaning:** the new business transaction is to be an Inferior (sub-coordinator or sub-
3062 composer) of the Superior identified by the CONTEXT. The Factory (receiver of the
3063 BEGIN) will perform the enrolment.

3064 **target-address:** the “target-address” is that of the BEGIN – this will be the address of the
3065 Factory.

3066 **Standard qualifiers**

3067 The following qualifiers are expected to be of general use to many applications and environments.
3068 The URI “urn:oasis:names:tc:BTP:1.0:qualifiers” is used in the Qualifier group
3069 value for the qualifiers defined here.

3070 **Transaction timelimit**

3071 The transaction timelimit allows the Superior (or an application element initiating the business
3072 transaction) to indicate the expected length of the active phase, and thus give an indication to the
3073 Inferior of when it would be appropriate to initiate cancellation if the active phase appears to
3074 continue too long. The time limit ends (the clock stops) when the Inferior decides to be prepared
3075 and issues PREPARED to the Superior.

3076 It should be noted that the expiry of the time limit does not change the permissible actions of the
3077 Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is **permitted** to
3078 initiate cancellation for internal reasons. The timelimit gives an indication to the entity of when it
3079 will be useful to exercise this right.

3080 The qualifier is propagated on a CONTEXT message.

3081 The “Qualifier name” shall be “transaction-timelimit”.

3082 The “Content” shall contain the following field:

Content field	Type
Timelimit	Integer

3083

3084 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
3085 time of transmission of the containing CONTEXT, of the active phase of the business
3086 transaction.

3087 **Inferior timeout**

3088 This qualifier allows an Inferior to limit the duration of its “promise”, when sending PREPARED,
3089 that it will maintain the ability to confirm or cancel the effects of all associated operations.
3090 Without this qualifier, an Inferior is expected to retain the ability to confirm or cancel
3091 indefinitely. If the timeout does expire, the Inferior is released from its promise and can apply the
3092 decision indicated in the qualifier.

3093 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply a
3094 confirm or cancel decision before the CONFIRM or CANCEL is received and before this timeout
3095 expires (or if this qualifier is not used). Such a decision is termed a heuristic decision, and (as
3096 with other transaction mechanisms), is considered to be an exceptional event. As with heuristic
3097 decisions, the taking of an autonomous decision by a Inferior **subsequent** to the expiry of this
3098 timeout, is liable to cause contradictory decisions across the business transaction. BTP ensures
3099 that at least the occurrence of such a contradiction will be (eventually) reported to the Superior of
3100 the business transaction. BTP treats “true” heuristic decisions and autonomous decisions after
3101 timeout the same way – in fact, the expiry in this timeout does not cause a qualitative (state table)
3102 change in what can happen, but rather a step change in the probability that it will.

3103 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
3104 intended decision, only that it is at liberty to do so. An implementation may choose to only apply
3105 the decision if there is contention for the underlying resource, for example. Nevertheless,
3106 Superiors are recommended to avoid relying on this and ensure decisions for the business
3107 transaction are made before these timeouts expire (and allow a margin of error for network
3108 latency etc.).

3109 The qualifier may be present on a PREPARED message. If the PREPARED message has the
3110 “default-is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall have
3111 the value “cancel”.

3112 The “Qualifier name” shall be “inferior-timeout”.

3113 The “Content” shall contain the following fields:

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

3114

3115 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
3116 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the effects
3117 of the associated operations, as ordered by the receiving Superior.

3118 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an
3119 autonomous decision is made.

3120 **Minimum inferior timeout**

3121 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the
3122 Inferior. If a Superior knows that the decision for the business transaction will not be determined
3123 for some period, it can require that Inferiors do not send PREPARED messages with Inferior
3124 timeouts that would expire before then. An Inferior that is unable or unwilling to send a
3125 PREPARED message with a longer (or no) timeout **should** cancel, and reply with CANCELLED.

3126 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If present on
3127 more than one, and with different values of the MinimumTimeout field, the value on
3128 ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall prevail over
3129 either of the others.

3130 The "Qualifier name" shall be "minimum-inferior-timeout".

3131 The "Content" shall contain the following field:

Content field	Type
MinimumTimeout	Integer

3132

3133 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be
3134 acceptable in the Inferior timeout qualifier on an answering PREPARED message.

3135 **Inferior name**

3136 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on
3137 INFERIOR_STATUSES and thus allow the Terminator to determine which Inferior (of the
3138 Composer or Coordinator) is related to which application work. This is in addition to the
3139 "inferior-identifier" field. The name can be human-readable and can also be used in fault tracing,
3140 debugging and auditing.

3141 The name is never used by the BTP actors themselves to identify each other or to direct messages.
3142 (The BTP actors use the addresses and the identifiers in the message parameters for those
3143 purposes.)

3144 This specification makes no requirement that the names are unambiguous within any scope
 3145 (unlike the globally unambiguous “inferior-identifier” on ENROLLED and BEGUN). Other
 3146 specifications, including those defining use of BTP with a particular application may place
 3147 requirements on the use and form of the names. (This may include reference to information
 3148 passed in application messages or in other, non-standardised, qualifiers.)

3149 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item in
 3150 INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if present,
 3151 the same qualifier value **should** be included in the consequent ENROL. If
 3152 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an inferior-
 3153 name qualifier, the same qualifier value **should** be included in the Status-item.

3154 The “Qualifier -name” shall be “inferior-name”

3155 The “Content” shall contain the following fields:

Content field	Type
inferior-name	String

3156

3157 **Inferior name** the name assigned to the enrolling Inferior.

3158 State Tables

3159 The state tables deal with the state transitions of the Superior and Inferior roles and which
 3160 message can be sent and received in each state. The state tables directly cover only a single, bi-
 3161 lateral Superior:Inferior relationship. The interactions between, for example, multiple Inferiors of
 3162 a single Superior that will apply the same decision to all or some (of them), are dealt with in the
 3163 definitions of the “decision” events which also specify when changes are made to persistent state
 3164 information (see below).

3165 There are two state tables, one for Superior, one for Inferior. States are identified by a letter-digit
 3166 pair, with upper-case letters for the superior, lower-case for the inferior. The same letter is used to
 3167 group states which have the same, or similar, persistent state, with the digit indicating volatile
 3168 state changes or minor variations. Corresponding upper and lower-case letters are used to identify
 3169 (approximately) corresponding Superior and Inferior states.

3170 The Inferior table includes events occurring both at the Inferior as such and at the associated
 3171 Enroller, as the Enroller’s actions are constrained by and constrain the Inferior role itself.

3172 In the state tables, each side is either waiting to make a decision or can send a message. For some
 3173 states, the message to be sent is a repetition of a regular message; for other states, the
 3174 INFERIOR_STATE or SUPERIOR_STATE message can be sent, requesting a response.
 3175 Normally, on entry to a state that allows the sending of any message other than one of the
 3176 *_STATE messages, the implementation will send that message – failure to do so will cause the
 3177 relationship to lock up. The message can be resent if the implementation determines that the
 3178 original message (or the next message sent in reply) may have been lost.

3179 **Status queries**

3180 In BTP the messages SUPERIOR_STATE and INFERIOR_STATE are available to prompt the
3181 peer to report its current state by repeating the previous message (when this is allowed) or by
3182 sending the other *_STATE message. The “reply_requested” parameter of these messages
3183 distinguishes between their use as a prompt and as a reply. An implementation receiving a
3184 *_STATE message with “reply_requested” as “true” is not required to reply immediately – it may
3185 choose to delay any reply until a decision event occurs and then send the appropriate new
3186 message (e.g. on receiving INFERIOR_STATE/prepared/y while in state E1, a superior is
3187 permitted to delay until it has performed “decide to confirm” or “decide to cancel”). However,
3188 this may cause the other side to repeatedly send interrogatory *_STATE messages.

3189 Note that a Superior (or some entity standing in for a now-extinct Superior) uses
3190 SUPERIOR_STATE/unknown to reply to messages received from an Inferior where the
3191 Superior:Inferior relationship is in an unknown (using state “Y1”). The *_STATE messages with
3192 a “state” value “inaccessible” can be used as a reply when **any** message is received and the
3193 implementation is temporarily unable to determine whether the relationship is known or what the
3194 state is. Receipt of the *_STATE/inaccessible messages is not shown in the tables and has no
3195 effect on the state at the receiving side (though it may cause the implementation to resend its own
3196 message after some interval of its own choosing).

3197 **Decision events**

3198 The persistent state changes (equivalent to logging in a regular transaction system) and some
3199 other events are modelled as “decision events” (e.g. “decide to confirm”, “decide to be
3200 prepared”). The exact nature of the real events and changes in an implementation that are
3201 modelled by these events depends on the position of the Superior or Inferior within the business
3202 transaction and on features of the implementation (e.g. making of a persistent record of the
3203 decision means that the information will survive at least some failures that otherwise lose state
3204 information, but the level of survival depends on the purpose of the implementation). Table 3 and
3205 Table 4 define the decision events.

3206 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
3207 PREPARE indicates that the application exchange (to associate operations with the Inferior) is
3208 complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier state
3209 corresponding to an incomplete application exchange. However, implementations are not required
3210 to make the sending of PREPARE persistent in terms of recovery – a Superior that experiences
3211 failure after sending PREPARE may, on recovery, have no information about the transaction, in
3212 which case it is considered to be in the completed state (Z), which will imply the cancellation of
3213 the Inferior and its associated operations.

3214 Where a Superior is an Intermediate (i.e. is itself an Inferior to another Superior entity), in a
3215 transaction tree, its “decide to confirm” and “decide to cancel” decisions will in fact be the receipt
3216 of a CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
3217 persistent information (which would combine both superior and inferior information, pointing
3218 both up and down the tree).

3219 **Disruptions – failure events**

3220 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or may)
3221 cause a change of state. The disruption events in the state tables model different extents of loss of
3222 state information. An implementation is **not** required to exhibit all the possible disruption events,
3223 but it is not allowed to exhibit state transitions that do not correspond to a possible disruption.
3224 The different levels of disruption describe legitimate states for the endpoint to be in after it has
3225 been restored to normal functioning. The absence of a destination state for the disruption events
3226 means that such a transition is not legitimate – thus, for example, an Inferior that has decided to
3227 be prepared will always recover to the same state, by virtue of the information persisted in the
3228 “decide to be prepared” event.

3229 In addition to the disruption events in the tables, there is an implicit “disruption 0” event, which
3230 involves possible interruption of service and loss of messages in transit, but no change of state
3231 (either because no state information was lost, or because recovery from persistent information
3232 restores the implementation to the same state). The “disruption 0” event would typically be an
3233 appropriate abstraction for a communication failure.

3234 **Invalid cells and assumptions of the communication mechanism**

3235 The empty cells in state table represent events that cannot happen. For events corresponding to
3236 sending a message or any of the decision events, this prohibition is absolute – e.g. a conformant
3237 implementation in the Superior active state “B1” will not send CONFIRM. For events
3238 corresponding to receiving a message, the interpretation depends on the properties of the
3239 underlying communications mechanism.

3240 For all communication mechanisms, it is assumed that

3241 a) the two directions of the Superior:Inferior communication are not synchronised –
3242 that is messages travelling in opposite directions can cross each other to any
3243 degree; any number of messages may be in transit in either direction; and

3244 b) messages may be lost arbitrarily

3245 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered at
3246 all, are delivered to the receiver in the order they were sent) , then receipt of a message in a state
3247 where the corresponding cell is empty indicates that the far-side has sent a message out of order –
3248 a FAULT message with the “fault-type” “WrongState” can be returned.

3249 If the communication mechanisms cannot guarantee ordered delivery, then messages received
3250 where the corresponding cell is empty should be ignored. Assuming the far-side is conformant,
3251 these messages can assumed to be “stale” and have been overtaken by messages sent later but
3252 already delivered. (If the far-side is non-conformant, there is a problem anyway).

3253 **Meaning of state table events**

3254 The tables in this section define the events (rows) in the state tables. Table 2 defines the events
3255 corresponding to sending or receiving BTP messages and the disruption events. Table 3 describes
3256 the decision events for an Inferior, Table 4 those for a Superior.

3257 The decision events for a Superior, defined in Table 4 cannot be specified without reference to
3258 other Inferiors to which it is Superior and to its relation with the application or other entity that
3259 (acting ultimately on behalf of the application) drives it.

3260 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and which
3261 are to be treated as an atomic decision unit with (and thus including) the Inferior on this
3262 relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior-type” of
3263 “atom”, this will be all Inferiors established with same Superior address and “superior-identifier”
3264 except those from which RESIGN has been received. If the CONTEXT had “superior-type” of
3265 “cohesion”, the “remaining Inferiors” excludes any that it has been determined will be cancelled,
3266 as well as any that have resigned – in other words it includes only those for which a confirm
3267 decision is still possible or has been made. The determination of exactly which Inferiors are
3268 “remaining Inferiors” in a cohesion is determined, in some way, by the application. The term
3269 “Other remaining Inferiors” excludes this Inferior on this relationship. A Superior with a single
3270 Inferior will have no “other remaining Inferiors”.

3271 In order to ensure that the confirmation decision is delivered to all remaining Inferiors, despite
3272 failures, the Superior must persistently record which these Inferiors are (i.e. their addresses and
3273 identifiers). It must also either record that the decision is confirm, or ensure that the confirm
3274 decision (if there is one) is persistently recorded somewhere else, and that it will be told about it.
3275 This latter would apply if the Superior were also BTP Inferior to another entity which persisted a
3276 confirm decision (or recursively deferred it still higher). However, since there is no requirement
3277 that the Superior be also a BTP Inferior to any other entity, the behaviour of asking another entity
3278 to make (and persist) the confirm decision is termed "offering confirmation" - the Superior offers
3279 the possible confirmation of itself, and its remaining Inferiors to some other entity. If that entity
3280 (or something higher up) then does make and persist a confirm decision, the Superior is
3281 "instructed to confirm" (which is equivalent BTP CONFIRM).

3282 The application, or an entity acting indirectly on behalf of the application, may request a Superior
3283 to prepare an Inferior (or all Inferiors). This typically implies that there will be no more
3284 operations associated with the Inferior. Following a request to prepare all remaining Inferiors, the
3285 Superior may offer confirmation to the entity that requested the prepare. (If the Superior is also a
3286 BTP Inferior, its superior can be considered an entity acting on behalf of the application.)

3287 The application, or an entity acting indirectly on behalf of the application, may also request
3288 confirmation. This means the Superior is to attempt to make and persist a confirm decision itself,
3289 rather than offer confirmation.

3290 **Table 2 : send, receive and disruption events**

Event name	Meaning
send/receive ENROL/rsp-req	send/receive ENROL with response-requested = true
send/receive ENROL/no-rsp-req	send/receive ENROL with response-requested = false
send/receive RESIGN/rsp-req	send/receive RESIGN with response-requested = true
send/receive RESIGN/no-rsp-req	send/receive RESIGN with response-requested = false
send/receive PREPARED	send/receive PREPARED, with default-cancel = false

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

3291

3292

Table 3 : Decision events for Inferior

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

Event name	Meaning
decide to cancel autonomously	<ul style="list-style-type: none"> Decision to cancel autonomously has been made persistent the effects of associated operations will be cancelled regardless of failures
apply ordered confirmation	<ul style="list-style-type: none"> Effects of all associated operations have been confirmed; Persistent information is effectively removed
remove persistent information	<ul style="list-style-type: none"> Persistent information is effectively removed;
detect problem	<ul style="list-style-type: none"> For at least some of the associated operations, EITHER <ul style="list-style-type: none"> they cannot be consistently cancelled or consistently confirmed; OR it cannot be determined whether they will be cancelled or confirmed AND, information about this is not persistent
detect and record problem	<ul style="list-style-type: none"> As for the first condition of "detect problem" information recording this has been persisted (to the degree considered appropriate), or the detection itself is persistent. (i.e. will be re-detected on recovery)

3293

3294

Table 4: Decision events for a Superior

Event name	Meaning
decide to confirm one-phase	<ul style="list-style-type: none"> All associated application messages to be sent to the service have been sent; There are no other remaining Inferiors If an atom, all enrolments that would create other Inferiors have completed (no outstanding CONTEXT_REPLYS) The Superior has been requested to confirm
decide to prepare	<ul style="list-style-type: none"> All associated application messages to be sent to the service have been sent; The Superior has been requested to prepare this Inferior
decide to confirm	<ul style="list-style-type: none"> Either <ul style="list-style-type: none"> PREPARED or PREPARED/cancel has been received from all other remaining Inferiors; AND

Event name	Meaning
	<ul style="list-style-type: none"> o Superior has been requested to confirm; AND o persistent information records the confirm decision and identifies all remaining Inferiors; • Or o persistent information records an offer of confirmation and has been instructed to confirm
decide to cancel	<ul style="list-style-type: none"> • Superior has not offered confirmation; OR • Superior has offered confirmation and has been instructed to cancel; OR • Superior has offered confirmation but has made an autonomous cancellation decision
remove confirm information	<ul style="list-style-type: none"> • Persistent information has been effectively removed;
record contradiction	<ul style="list-style-type: none"> • Information recording the contradiction has been persisted (to the degree considered appropriate)

3295

3296 Persistent information

3297 Persisted information (especially prepared information at an Inferior, confirm information at a
3298 Superior) may include qualifications of the state carried in Qualifiers of the corresponding
3299 message (e.g. inferior timeouts in prepared information). It may also include application-specific
3300 information (especially in Inferiors) to allow the future confirmation or cancellation of the
3301 associated operations. In some cases it will also include information allowing an application
3302 message sent with a BTP message (e.g. PREPARED) to be repeated.

3303 The “effective” removal of persistent information allows for the possibility that the information is
3304 retained (perhaps for audit and tracing purposes) but some change to the persistent information
3305 (as a whole) means that if there is a failure after such change, on recovery, the persistent
3306 information does not cause the endpoint to return the state it would have recovered to before the
3307 change.

3308 In all cases, the degree to which information described as “persistent” will survive failure is a
3309 configuration and implementation option. An implementation **should** describe the level of failure
3310 that it is capable of surviving. For applications manipulating information that is itself volatile (e.g.
3311 network configurations), there is no requirement to make the BTP state information more
3312 persistent than the application information.

3313 The degree of persistence of the recording of a hazard (problem) at an Inferior and recording of a
3314 detected contradiction at a Superior may be different from that applying to the persistent prepared
3315 and confirm information. Implementations and configuration may choose to pass hazard and
3316 contradiction information via management mechanisms rather than through BTP. Such passing of
3317 information to a management mechanism could be treated as “record problem” or “record
3318 contradiction”.

Table 5 : Superior states

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

Table 6 : Inferior states

State	summary
i1	aware of CONTEXT
a1	enrolling
b1	enrolled
c1	resigning
d1	preparing
e1	prepared
e2	prepared,default to cancel
f1	confirming
f2	confirming after default cancel
g1	CANCEL received in prepared state
g2	CANCEL received in prepared/cancel state
h1	Autonomously confirmed
h2	autonomously confirmed, superior confirmed
j1	autonomously cancelled
j2	autonomously cancelled, superior cancelled
k1	autonomously cancelled, contradicted
k2	autonomously cancelled, CONTRADICTION received
l1	autonomously confirmed, contradicted
l2	autonomously confirmed, CONTRADICTION received
m1	confirmation applied
n1	cancelling
p1	hazard detected, not recorded
p2	hazard detected in prepared state, not recorded
q1	hazard recorded
s1	CONFIRM_ONE_PHASE received after prepared state
s2	CONFIRM_ONE_PHASE received
s3	CONFIRM_ONE_PHASE received, confirming
s4	CONFIRM_ONE_PHASE received, cancelling
s5	CONFIRM_ONE_PHASE received, hazard detected
s6	CONFIRM_ONE_PHASE received, hazard recorded
x1	completed, presuming abort
x2	completed, presuming abort after prepared/cancel
y1	completed, queried

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

3322

3323 Superior state table

3324 Table 7: Superior state table – normal forward progression

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

3325

3326

Table 8: Superior state table – cancellation and contradiction

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

Table 9: Superior state table – hazard and request confirm

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

3330

Table 10: Superior state table – query after completion and completed states

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

3331

3332

3332 Inferior state table

3333 Table 11: Inferior state table – normal forward progression

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

3334

3335

3335

Table 12: Inferior state table – cancellation and contradiction

	g1	g2	h1	h2	j 1	j 2	k1	k2	l 1	l 2
send ENROL/rsp-req send ENROL/no-rsp-req send RESI GN/rsp-req send RESI GN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIR MED/auto send CONFIR MED/response send CANCELLED send HAZARD										
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown										
recei ve ENROLLED recei ve RESI GNED recei ve PREPARE recei ve CONFIR M_ONE_PHASE recei ve CONFIR M recei ve CANCEL recei ve CONTRADI CTI ON			h1 h1 s3 h2 h2 l 1 l 2		j 1 j 1 s4 k1 j 2 j 2 k2		k1 k2 k2		l 1 l 2 l 2	
recei ve SUP_STATE/active/y recei ve SUP_STATE/active recei ve SUP_STATE/prepared-rcvd/y recei ve SUP_STATE/prepared-rcvd recei ve SUP_STATE/unknown			h1 h1 h1 h1 l 1		j 1 j 1 j 1 j 1 j 2 j 2		k2 k2		l 1	
deci de to resi gn deci de to be prepared deci de to be prepared/cancel deci de to confi rm autonomously deci de to cancel autonomously appl y ordered confi rmati on remove persi stent i nformati on detect probl em detect and record probl em										
	n1 p2	n1 p2	m1		z		z		z	
di srupti on I di srupti on II di srupti on III	e1 e2		h1		j 1		j 1 k1 j 1		h1 l 1 h1	

3336

3337

3337

Table 13: Inferior state table – confirm, cancel ordered and hazard recording

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

3338

3339

Table 14: Inferior state table – request confirm states

	s1	s2	s3	s4	s5	s6
send ENROL/rsp-req send ENROL/no-rsp-req send RESI GN/rsp-req send RESI GN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIR MED/auto send CONFIR MED/response send CANCELLED send HAZARD			z	z	z	z
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown						
receive ENROLLED receive RESI GNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADI CTI ON	s1	s2	s3	s4	s5	s6
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown	x1	z	z	z	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		s3 s4 s2 s6				
disruption I disruption II disruption III	e1	z		z	z	

3341

Table 15: Inferior state table – completed states (including presume-abort and queried)

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

3342

3343

Persistent information

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

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

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

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

- "superior-address"(as on CONTEXT as updated by REDIRECT)

- "superior-identifier" (as on CONTEXT)

- "default-is-cancel" value (as on PREPARED)

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

- "inferior-address" (as on ENROL, as updated by REDIRECT)

- "inferior-identifier" (as on ENROL)

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

XML representation of Message Set

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

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

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

3378 In addition to an XML schema, this specification uses an informal syntax to describe the structure
3379 of the BTP messages. The syntax appears as an XML instance, but the values contain data types
3380 instead of values. The following symbols are appended to some of the XML constructs: ? (zero
3381 or one), * (zero or more), + (one or more.) The absence of one of these symbols corresponds to
3382 "one and only one."

3383 The Delivery parameters are shown in the XML with a darker background.

3384 Addresses

3385 As described in the "Abstract Message and Associated Contracts – Addresses" section, a BTP
3386 address comprises three parts, and for a "target-address" only the "additional information" field is
3387 inside the BTP messages. For all BTP messages whose abstract form includes a "target-address"
3388 parameter, the corresponding XML representation includes a "target-additional-information"
3389 element. This element may be omitted if it would be empty.

3390 For other addresses, all three fields are represent, as in:

```
3391 <btp:some-address>  
3392   <btp:binding-name>...carrier binding URI...</btp:binding-name>  
3393   <btp:binding-address>...carrier specific  
3394   address...</btp:binding-address>  
3395   <btp:additional-information>...optional additional addressing  
3396   information...</btp:additional-information> ?  
3397 </btp:some-address>
```

3399 A "published" address can be a set of <some-address>, which are alternatives which can be
3400 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to same
3401 endpoint, or backup endpoints. In the former, the receiver of the message has the choice of which
3402 address to use (depending on which binding is preferable.) In the case where multiple addresses
3403 are used for redundancy, a priority attribute can be specified to help the receiver choose among
3404 the addresses- the address with the highest priority should be used, other things being equal. The
3405 priority is used as a hint and does not enforce any behaviour in the receiver of the message.
3406 Default priority is a value of 1.

3407 Qualifiers

3408 The "Qualifier name" is used as the element name, within the namespace of the "Qualifier
3409 group".

3410 Examples:

```
3411 <btpq:inferior-timeout  
3412   xmlns:btpq="urn:oasis:names:tc:BTP:1.0:qualifiers"  
3413   xmlns:btp="urn:oasis:names:tc:BTP:1.0:core"  
3414   btp:must-be-understood="false"  
3415   btp:to-be-propagated="false">1800</btpq:inferior-timeout>  
3416 <auth:username  
3417   xmlns:auth="http://www.example.com/ns/auth"  
3418   xmlns:btp="urn:oasis:names:tc:BTP:1.0:core"
```

3419 btp:must-be-understood="true"
3420 btp:to-be-propagated="true">jtauber</auth:username>
3421
3422 Attributes must-be-understood **has default value “true”** and to-be-propagated has default value
3423 “false”.

3424 **Identifiers**

3425 Identifiers shall be URIs "

3426 *Note – Identifiers need to be globally unambiguous. Apart from their generation, .the*
3427 *only operation the BTP implementations have to perform on identifiers is to match*
3428 *them.*

3429 **Message References**

3430 Each BTP message has an optional id attribute to give it a unique identifier. An application can
3431 make use of those identifiers, but no processing is enforced.

3432 **Messages**

3433 **CONTEXT**

```
3434       <btp:context id?>  
3435        <btp:superior-address> +  
3436        ...address...  
3437       </btp:superior-address>  
3438       <btp:superior-identifier>...URI...</btp:superior-identifier>  
3439       <btp:superior-type>cohesion|atom</btp:superior-type>  
3440       <btp:qualifiers> ?  
3441        ...qualifiers...  
3442       </btp:qualifiers>  
3443       <btp:reply-address> ?  
3444        ...address...  
3445       </btp:reply-address>  
3446       </btp:context>
```

3447 **CONTEXT_REPLY**

```
3448       <btp:context-reply id?>  
3449        <btp:superior-identifier>...URI...</btp:superior-identifier>  
3450        <btp:completion-  
3451        status>completed|incomplete|related|repudiated</btp:completion-  
3452        status>  
3453        <btp:qualifiers> ?  
3454        ...qualifiers...  
3455        </btp:qualifiers>  
3456        <btp:target-additional-information> ?  
3457        ...additional address information...  
3458        </btp:target-additional-information>  
3459       </btp:context-reply>
```

3460 REQUEST_STATUS

```
3461 <btpr:request-status id?>
3462   <btpr:target-identifier>...URI...</btpr:target-identifier>
3463   <btpr:qualifiers> ?
3464   ...qualifiers...
3465 </btpr:qualifiers>
3466   <btpr:target-additional-information> ?
3467   ...additional address information...
3468 </btpr:target-additional-information>
3469   <btpr:reply-address> ?
3470   ...address...
3471 </btpr:reply-address>
3472 </btpr:request-status>
```

3473 STATUS

```
3474 <btpr:status id?>
3475   <btpr:responders-identifier>...URI...</btpr:responders-identifier>
3476   <btpr:status-value>created|enrolling|active|resigning|
3477     resigned|preparing|prepared|
3478     confirming|confirmed|cancelling|cancelled|
3479     cancel-contradiction|confirm-contradiction|
3480     hazard|contradicted|unknown|inaccessible</btpr:status-
3481 value>
3482   <btpr:qualifiers> ?
3483   ...qualifiers...
3484 </btpr:qualifiers>
3485   <btpr:target-additional-information> ?
3486   ...additional address information...
3487 </btpr:target-additional-information>
3488 </btpr:status>
```

3489 FAULT

```
3490 <btpr:fault id?>
3491   <btpr:superior-identifier>...URI...</btpr:superior-identifier> ?
3492   <btpr:inferior-identifier>...URI...</btpr:inferior-identifier> ?
3493   <btpr:fault-type>...fault type name...</btpr:fault-type>
3494   <btpr:fault-data>...fault data...</btpr:fault-data> ?
3495   <btpr:fault-text>...string data ...</btpr:fault-data> ?
3496   <btpr:qualifiers> ?
3497   ...qualifiers...
3498 </btpr:qualifiers>
3499   <btpr:target-additional-information> ?
3500   ...additional address information...
3501 </btpr:target-additional-information>
3502 </btpr:fault>
3503
```

3504 The following fault type names are represented by simple strings, corresponding to the entries
3505 defined in the abstract message set:

- 3506 • communication-failure
- 3507 • duplicate-inferior
- 3508 • general
- 3509 • invalid-decider
- 3510 • invalid-inferior
- 3511 • invalid-superior
- 3512 • status-refused
- 3513 • invalid-terminator
- 3514 • unknown-parameter
- 3515 • unknown-transaction
- 3516 • unsupported-qualifier
- 3517 • wrong-state
- 3518 • redirect
- 3519

3520 Revisions of this specification may add other fault type names, which shall be simple strings of
 3521 letters, numbers and hyphens. If other specifications define fault type names to be used with BTP,
 3522 the names shall be URIs.

3523 Fault data can take on various forms:

3524 Identifier:

```
3525           <btp: fault-data>...URI...</btp: fault-data>
```

3527 Inferior Identity:

```
3528           <btp: fault-data>
3529            <btp: inferior-address> +
3530            ...address...
3531           </btp: inferior-address>
3532           <btp: inferior-identifier>...URI...</btp: inferior-identifier>
3533           </btp: fault-data>
```

3535 **ENROL**

```
3536           <btp: enrol id?>
3537            <btp: superior-identifier>...URI...</btp: superior-identifier>
3538            <btp: response-requested>true| false</btp: response-requested>
3539            <btp: inferior-address> +
3540            ...address...
3541            </btp: inferior-address>
3542            <btp: inferior-identifier>...URI...</btp: inferior-identifier>
3543            <btp: qualifiers> ?
3544            ...qualifiers...
```



```

3545     </btp:qualifiers>
3546     <btp:target-additional-information> ?
3547         ...additional address information...
3548     </btp:target-additional-information>
3549     <btp:reply-address> ?
3550         ...address...
3551     </btp:reply-address>
3552 </btp:enrol>

```

3553 ENROLLED

```

3554 <btp:enrolled id?>
3555     <btp:sender-address> ?
3556         ...address...
3557     </btp:sender-address>
3558     <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3559     <btp:qualifiers> ?
3560         ...qualifiers...
3561     </btp:qualifiers>
3562     <btp:target-additional-information> ?
3563         ...additional address information...
3564     </btp:target-additional-information>
3565 </btp:enrolled>

```

3566 RESIGN

```

3567 <btp:resign id?>
3568     <btp:superior-identifier>...URI...</btp:superior-identifier>
3569     <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3570     <btp:response-requested>true|false</btp:response-requested>
3571     <btp:qualifiers> ?
3572         ...qualifiers...
3573     </btp:qualifiers>
3574     <btp:target-additional-information> ?
3575         ...additional address information...
3576     </btp:target-additional-information>
3577     <btp:sender-address> ?
3578         ...address...
3579     </btp:sender-address>
3580 </btp:resign>

```

3581 RESIGNED

```

3582 <btp:resigned id?>
3583     <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3584     <btp:qualifiers> ?
3585         ...qualifiers...
3586     </btp:qualifiers>
3587     <btp:target-additional-information> ?
3588         ...additional address information...
3589     </btp:target-additional-information>
3590     <btp:sender-address> ?
3591         ...address...
3592     </btp:sender-address>

```

3593 </btp:resigned>

3594 **PREPARE**

```
3595       <btp:prepare id?>
3596        <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3597        <btp:qualifiers> ?
3598        ...qualifiers...
3599        </btp:qualifiers>
3600        <btp:target-additional-information> ?
3601        ...additional address information...
3602        </btp:target-additional-information>
3603        <btp:sender-address> ?
3604        ...address...
3605        </btp:sender-address>
3606       </btp:prepare>
```

3607 **PREPARED**

```
3608       <btp:prepared id?>
3609        <btp:superior-identifier>...URI...</btp:superior-identifier>
3610        <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3611        <btp:default-is-cancel>true|false</btp:default-is-cancel>
3612        <btp:qualifiers> ?
3613        ...qualifiers...
3614        </btp:qualifiers>
3615        <btp:target-additional-information> ?
3616        ...additional address information...
3617        </btp:target-additional-information>
3618        <btp:sender-address> ?
3619        ...address...
3620        </btp:sender-address>
3621       </btp:prepared>
```

3622 **CONFIRM**

```
3623       <btp:confirm id?>
3624        <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3625        <btp:qualifiers> ?
3626        ...qualifiers...
3627        </btp:qualifiers>
3628        <btp:target-additional-information> ?
3629        ...additional address information...
3630        </btp:target-additional-information>
3631        <btp:sender-address> ?
3632        ...address...
3633        </btp:sender-address>
3634       </btp:confirm>
```

3635 **CONFIRMED**

```
3636       <btp:confirmed id?>
3637        <btp:superior-identifier>...URI...</btp:superior-identifier>
3638        <btp:inferior-identifier>...URI...</btp:inferior-identifier>
```

```

3639     <btp:confirmed-received>true|false</btp:confirmed-received>
3640     <btp:qualifiers> ?
3641         ...qualifiers...
3642     </btp:qualifiers>
3643     <btp:target-additional-information> ?
3644         ...additional address information...
3645     </btp:target-additional-information>
3646     <btp:sender-address> ?
3647         ...address...
3648     </btp:sender-address>
3649 </btp:confirmed>

```

3650 CANCEL

```

3651 <btp:cancel id?>
3652     <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3653     <btp:reply-address> ?
3654     ...address...
3655     </btp:reply-address>
3656     <btp:qualifiers> ?
3657         ...qualifiers...
3658     </btp:qualifiers>
3659     <btp:target-additional-information> ?
3660         ...additional address information...
3661     </btp:target-additional-information>
3662     <btp:sender-address> ?
3663         ...address...
3664     </btp:sender-address>
3665 </btp:cancel>

```

3666 CANCELLED

```

3667 <btp:cancelled id?>
3668     <btp:superior-identifier>...URI...</btp:superior-identifier>
3669     <btp:inferior-identifier>...URI...</btp:inferior-identifier> ?
3670     <btp:qualifiers> ?
3671         ...qualifiers...
3672     </btp:qualifiers>
3673     <btp:target-additional-information> ?
3674         ...additional address information...
3675     </btp:target-additional-information>
3676     <btp:sender-address> ?
3677         ...address...
3678     </btp:sender-address>
3679 </btp:cancelled>

```

3680 CONFIRM_ONE_PHASE

```

3681 <btp:confirm-one-phase id?>
3682     <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3683     <btp:report-hazard>true|false</btp:report-hazard>
3684     <btp:qualifiers> ?
3685         ...qualifiers...
3686     </btp:qualifiers>

```

```

3687 <btp:target-additional-information> ?
3688   ...additional address information...
3689 </btp:target-additional-information>
3690 <btp:sender-address> ?
3691   ...address...
3692 </btp:sender-address>
3693 </btp:confirm-one-phase>

```

3694 HAZARD

```

3695 <btp:hazard id?>
3696   <btp:superior-identifier>...URI...</btp:superior-identifier>
3697   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3698   <btp:level>mixed|possible</btp:level>
3699   <btp:qualifiers> ?
3700     ...qualifiers...
3701 </btp:qualifiers>
3702 <btp:target-additional-information> ?
3703   ...additional address information...
3704 </btp:target-additional-information>
3705 <btp:sender-address> ?
3706   ...address...
3707 </btp:sender-address>
3708 </btp:hazard>

```

3709 CONTRADICTION

```

3710 <btp:contradiction id?>
3711   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3712   <btp:qualifiers> ?
3713     ...qualifiers...
3714 </btp:qualifiers>
3715 <btp:target-additional-information> ?
3716   ...additional address information...
3717 </btp:target-additional-information>
3718 <btp:sender-address> ?
3719   ...address...
3720 </btp:sender-address>
3721 </btp:contradiction>

```

3722 SUPERIOR_STATE

```

3723 <btp:superior-state id?>
3724   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3725   <btp:status>active|prepared-
3726 received|inaccessible|unknown</btp:status>
3727   <btp:response-requested>true|false</btp:response-requested>
3728   <btp:qualifiers> ?
3729     ...qualifiers...
3730 </btp:qualifiers>
3731 <btp:target-additional-information> ?
3732   ...additional address information...
3733 </btp:target-additional-information>
3734 <btp:sender-address> ?

```

```

3735     ...address...
3736     </btp:sender-address>
3737 </btp:superior-state>

```

3738 INFERIOR_STATE

```

3739 <btp:inferior-state id?>
3740   <btp:superior-identifier>...URI...</btp:superior-identifier>
3741   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3742   <btp:status>active|inaccessible|unknown</btp:status>
3743   <btp:response-requested>true|false</btp:response-requested>
3744   <btp:qualifiers> ?
3745     ...qualifiers...
3746   </btp:qualifiers>
3747   <btp:target-additional-information> ?
3748     ...additional address information...
3749   </btp:target-additional-information>
3750   <btp:sender-address> ?
3751     ...address...
3752   </btp:sender-address>
3753 </btp:inferior-state>

```

3754 REDIRECT

```

3755 <btp:redirect id?>
3756   <btp:superior-identifier>...URI...</btp:superior-identifier> ?
3757   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3758   <btp:old-address> +
3759     ...address...
3760   </btp:old-address>
3761   <btp:new-address> +
3762     ...address...
3763   </btp:new-address>
3764   <btp:qualifiers> ?
3765     ...qualifiers...
3766   </btp:qualifiers>
3767   <btp:target-additional-information> ?
3768     ...additional address information...
3769   </btp:target-additional-information>
3770 </btp:redirect>

```

3771 BEGIN

```

3772 <btp:begin id?>
3773   <btp:transaction-type>cohesion|atom</btp:transaction-type>
3774   <btp:qualifiers> ?
3775     ...qualifiers...
3776   </btp:qualifiers>
3777   <btp:target-additional-information> ?
3778     ...additional address information...
3779   </btp:target-additional-information>
3780   <btp:reply-address> ?
3781     ...address...
3782   </btp:reply-address>

```

3783 </btp:begin>

3784 **BEGUN**

```
3785 <btp:begin id?>
3786   <btp:decider-address> *
3787     ...address...
3788   </btp:decider-address>
3789   <btp:inferior-address> *
3790     ...address...
3791   </btp:inferior-address>
3792   <btp:transaction-identifier>...URI...</btp:transaction-
3793 identifier>
3794   <btp:qualifiers> ?
3795     ...qualifiers...
3796   </btp:qualifiers>
3797   <btp:target-additional-information> ?
3798     ...additional address information...
3799   </btp:target-additional-information>
3800 </btp:begin>
```

3801 **PREPARE_INFERIORS**

```
3802 <btp:prepare-inferiors id?>
3803   <btp:transaction-identifier>...URI...</btp:transaction-
3804 identifier>
3805   <btp:inferiors-list> ?
3806     <btp:inferior-handleidentifier>...URI...</btp:inferior-
3807 handleidentifier> +
3808   </btp:inferiors-list>
3809   <btp:qualifiers> ?
3810     ...qualifiers...
3811   </btp:qualifiers>
3812   <btp:target-additional-information> ?
3813     ...additional address information...
3814   </btp:target-additional-information>
3815   <btp:reply-address> ?
3816     ...address...
3817   </btp:reply-address>
3818 </btp:prepare-inferiors>
```

3819 **CONFIRM_TRANSACTION**

```
3820 <btp:confirm-transaction id?>
3821   <btp:transaction-identifier>...URI...</btp:transaction-
3822 identifier>
3823   <btp:inferiors-list> ?
3824     <btp:inferior-identifierhandle>...URI...</btp:inferior-
3825 identifierhandle> +
3826   </btp:inferiors-list>
3827   <btp:report-hazard>true|false</btp:report-hazard>
3828   <btp:qualifiers> ?
3829     ...qualifiers...
3830   </btp:qualifiers>
```

```

3831     <btp:target-additional-information> ?
3832     ...additional address information...
3833     </btp:target-additional-information>
3834     <btp:reply-address> ?
3835     ...address...
3836     </btp:reply-address>
3837 </btp: confirm_transaction>

```

3838 TRANSACTION_CONFIRMED

```

3839 <btp:transaction-confirmed id?>
3840   <btp:transaction-identifier>...URI...</btp:transaction-
3841   identifier>
3842   <btp:qualifiers> ?
3843   ...qualifiers...
3844   </btp:qualifiers>
3845   <btp:target-additional-information> ?
3846   ...additional address information...
3847   </btp:target-additional-information>
3848 </btp:transaction-confirmed>

```

3849 CANCEL_TRANSACTION

```

3850 <btp:cancel-transaction id?>
3851   <btp:transaction-identifier>...URI...</btp:transaction-
3852   identifier>
3853   <btp:report-hazard>true|false</btp:report-hazard>
3854   <btp:qualifiers> ?
3855   ...qualifiers...
3856   </btp:qualifiers>
3857   <btp:target-additional-information> ?
3858   ...additional address information...
3859   </btp:target-additional-information>
3860   <btp:reply-address> ?
3861   ...address...
3862   </btp:reply-address>
3863 </btp:cancel-transaction>

```

3864 CANCEL_INFERIORS

```

3865 <btp:cancel-inferiors id?>
3866   <btp:transaction-identifier>...URI...</btp:transaction-
3867   identifier> ?
3868   <btp:inferiors-list>
3869   <btp:inferior-identifierhandle>...URI...</btp:inferior-
3870   identifierhandle> +
3871   </btp:inferiors-list>
3872   <btp:qualifiers> ?
3873   ...qualifiers...
3874   </btp:qualifiers>
3875   <btp:target-additional-information> ?
3876   ...additional address information...
3877   </btp:target-additional-information>
3878   <btp:reply-address> ?

```

```

3879     ...address...
3880     </btp:reply-address>
3881 </btp:cancel-inferiors>

```

3882 TRANSACTION_CANCELLED

```

3883 <btp:transaction-cancelled id?>
3884   <btp:transaction-identifier>...URI...</btp:transaction-
3885   identifier>
3886   <btp:qualifiers> ?
3887   ...qualifiers...
3888 </btp:qualifiers>
3889   <btp:target-additional-information> ?
3890   ...additional address information...
3891 </btp:target-additional-information>
3892 </btp:transaction-cancelled>

```

3893 REQUEST_INFERIOR_STATUSES

```

3894 <btp:request-inferior-statuses id?>
3895   <btp:target-identifier>...URI...</btp:target-identifier>
3896   <btp:inferiors-list> ?
3897   <btp:inferior-handleidentifier>...URI...</btp:inferior-
3898   handleidentifier> +
3899   </btp:inferiors-list>
3900   <btp:qualifiers> ?
3901   ...qualifiers...
3902 </btp:qualifiers>
3903   <btp:target-additional-information> ?
3904   ...additional address information...
3905 </btp:target-additional-information>
3906   <btp:reply-address> ?
3907   ...address...
3908 </btp:reply-address>
3909 </btp:request-inferior-statuses>

```

3910 INFERIOR_STATUSES

```

3911 <btp:inferior-statuses id?>
3912   <btp:responders-identifier>...URI...</btp:responders-identifier>
3913   <btp:status-list>
3914     <btp:status-item> +
3915     <btp:inferior-handleidentifier>...URI...</btp:inferior-
3916     handleidentifier>
3917     <btp:status>active|resigned|preparing|prepared|
3918     autonomously-confirmed|autonomously-cancelled|
3919     confirming|confirmed|cancelling|cancelled|
3920     cancel-contradiction|confirm-contradiction|
3921     hazard|invalid</btp:status>
3922     <btp:qualifiers> ?
3923     ...qualifiers...
3924   </btp:qualifiers>
3925   </btp:status-item>
3926 </btp:status-list>

```



```

3927     <btqp:qualifiers> ?
3928         ...qualifiers...
3929     </btqp:qualifiers>
3930     <btqp:target-additional-information> ?
3931         ...additional address information...
3932     </btqp:target-additional-information>
3933 </btqp:inferior-statuses>

```

3934 **Standard qualifiers**

3935 The informal syntax for these messages assumes the namespace prefix “btqp” is associated with
 3936 the URI “urn:oasis:names:tc:BTP:1.0:qualifiers”.

3937 **Transaction timelimit**

```

3938     <btqp:transaction-timelimit>
3939         <btqp:timelimit>
3940             ...time in seconds...
3941         </btqp:timelimit>
3942     </btqp:transaction-timelimit>

```

3943 **Inferior timeout**

```

3944     <btqp:inferior-timeout>
3945         <btqp:timeout>
3946             ...time in seconds...
3947         </btqp:timeout>
3948         <btqp:intended-decision>confirm|cancel</btqp:intended-decision>
3949     </btqp:inferior-timeout>

```

3950 **Minimum inferior timeout**

```

3951     <btqp:minimum-inferior-timeout>
3952         <btqp:minimum-timeout>
3953             ...time in seconds...
3954         </btqp:minimum-timeout>
3955     </btqp:minimum-inferior-timeout>

```

3956 **Inferior name**

```

3957     <btqp:inferior-name>
3958         <btqp:inferior-name>
3959             ...string...
3960         </btqp:inferior-name>
3961     </btqp:inferior-name>

```

3962 **Compounding of Messages**

3963 Relating BTP to one another, in a “group” is represented by containing them within the
 3964 btqp:related-group element, with the related messages as child elements. The processing for the
 3965 group is defined in the section “Groups – combinations of related messages”. For example

```

3966     <btp:related-group>
3967         <btp:context-reply>
3968             ...<completion-status>related</completion-status> ...
3969         </btp:context-reply>
3970         <btp:enrol>...</btp:enrol>
3971         <btp:prepared>...</btp:prepared>
3972     </btp:related-group>

```

3973 If the rules for the group state that the “target-address” of the abstract message is omitted, the
3974 corresponding target-address-information element shall be absent in the message in the related-
3975 group. The carrier protocol binding specifies how a relation between application and BTP
3976 messages is represented.

3977 Bundling (semantically insignificant combination) of BTP messages and related groups is
3978 indicated with the "btp:messages" element, with the bundled messages and related groups as child
3979 elements. For example (confirming one and cancelling another inferiors of a cohesion):

3980

```

3981     <btp:messages>
3982         <btp:confirm>...</btp:confirm>
3983         <btp:cancel>...</btp:cancel>
3984     </btp:messages>
3985

```

3986 XML Schemas

3987 XML schema for BTP messages

```

3988 <?xml version="1.0"?>
3989 <schema
3990     xmlns="http://www.w3.org/2001/XMLSchema"
3991     targetNamespace="urn:oasis:names:tc:BTP:1.0:core"
3992     xmlns:btp="urn:oasis:names:tc:BTP:1.0:core"
3993     elementFormDefault="qualified">
3994
3995     <!-- Qualifiers -->
3996     <complexType name="qualifier-type">
3997         <simpleContent>
3998             <extension base="string">
3999                 <attribute name="must-be-understood" type="boolean"/>
4000                 <attribute name="to-be-propagated" type="boolean"/>
4001             </extension>
4002         </simpleContent>
4003     </complexType>
4004
4005     <element name="qualifier" type="btp:qualifier-type" abstract="true"/>
4006
4007     <element name="qualifiers">
4008         <complexType>
4009             <sequence>
4010                 <element ref="btp:qualifier" maxOccurs="unbounded"/>
4011             </sequence>
4012         </complexType>

```

```

4013     </element>
4014     <!-- example qualifier:
4015         <element name="some-qualifier" type="btp:qualifier-type"
4016 substitutionGroup="btp:qualifier"/>
4017     -->
4018
4019     <!-- Message set data types -->
4020     <simpleType name="identifier">
4021         <restriction base="anyURI" />
4022     </simpleType>
4023     <simpleType name="additional-information">
4024         <restriction base="string" />
4025     </simpleType>
4026     <complexType name="address">
4027         <sequence>
4028             <element name="binding-name" type="anyURI"/>
4029             <element name="binding-address" type="string"/>
4030             <element name="additional-information" type="btp:additional-
4031 information" minOccurs="0" />
4032         </sequence>
4033     </complexType>
4034     <simpleType name="superior-type">
4035         <restriction base="string">
4036             <enumeration value="cohesion"/>
4037             <enumeration value="atom"/>
4038         </restriction>
4039     </simpleType>
4040     <simpleType name="transaction-type">
4041         <restriction base="string">
4042             <enumeration value="cohesion"/>
4043             <enumeration value="atom"/>
4044         </restriction>
4045     </simpleType>
4046
4047     <!-- Compounding -->
4048     <element name="messages">
4049         <complexType>
4050             <sequence>
4051                 <element ref="btp:message" minOccurs="0"
4052 maxOccurs="unbounded"/>
4053             </sequence>
4054         </complexType>
4055     </element>
4056     <element name="related-group" substitutionGroup="btp:message">
4057         <complexType>
4058             <sequence>
4059                 <element ref="btp:message" minOccurs="0"
4060 maxOccurs="unbounded"/>
4061             </sequence>
4062         </complexType>
4063     </element>
4064
4065     <!-- Message set -->
4066     <element name="message" abstract="true" />
4067     <element name="context" substitutionGroup="btp:message">

```

```

4068         <complexType>
4069             <sequence>
4070                 <element name="superior-address" type="btp:address"
4071 maxOccurs="unbounded"/>
4072                 <element name="superior-identifier" type="btp:identifier"/>
4073                 <element name="superior-type" type="btp:superior-type"/>
4074                 <element ref="btp:qualifiers" minOccurs="0"/>
4075                 <element name="reply-address" type="btp:address"
4076 minOccurs="0"/>
4077             </sequence>
4078             <attribute name="id" type="ID" use="optional"/>
4079         </complexType>
4080     </element>
4081     <element name="context-reply" substitutionGroup="btp:message">
4082         <complexType>
4083             <sequence>
4084                 <element name="superior-identifier" type="btp:identifier"/>
4085                 <element name="completion-status">
4086                     <simpleType>
4087                         <restriction base="string">
4088                             <enumeration value="completed"/>
4089                             <enumeration value="incomplete"/>
4090                             <enumeration value="related"/>
4091                             <enumeration value="repudiated"/>
4092                         </restriction>
4093                     </simpleType>
4094                 </element>
4095                 <element ref="btp:qualifiers" minOccurs="0"/>
4096                 <element name="target-additional-information"
4097 type="btp:additional-information" minOccurs="0"/>
4098             </sequence>
4099             <attribute name="id" type="ID"/>
4100         </complexType>
4101     </element>
4102     <element name="request-status" substitutionGroup="btp:message">
4103         <complexType>
4104             <sequence>
4105                 <element name="target-identifier" type="btp:identifier"/>
4106                 <element ref="btp:qualifiers" minOccurs="0"/>
4107                 <element name="target-additional-information"
4108 type="btp:additional-information" minOccurs="0"/>
4109                 <element name="reply-address" type="btp:address"
4110 minOccurs="0"/>
4111             </sequence>
4112             <attribute name="id" type="ID"/>
4113         </complexType>
4114     </element>
4115     <element name="status" substitutionGroup="btp:message">
4116         <complexType>
4117             <sequence>
4118                 <element name="responders-identifier"
4119 type="btp:identifier"/>
4120                 <element name="status-value">
4121                     <simpleType>
4122                         <restriction base="string">

```

```

4123         <enumeration value="created"/>
4124         <enumeration value="enrolling"/>
4125         <enumeration value="active"/>
4126         <enumeration value="resigning"/>
4127         <enumeration value="resigned"/>
4128         <enumeration value="preparing"/>
4129         <enumeration value="prepared"/>
4130         <enumeration value="confirming"/>
4131         <enumeration value="confirmed"/>
4132         <enumeration value="cancelling"/>
4133         <enumeration value="cancelled"/>
4134         <enumeration value="cancel-contradiction"/>
4135         <enumeration value="confirm-contradiction"/>
4136         <enumeration value="hazard"/>
4137         <enumeration value="contradicted"/>
4138         <enumeration value="unknown"/>
4139         <enumeration value="inaccessible"/>
4140     </restriction>
4141 </simpleType>
4142 </element>
4143 <element ref="btp:qualifiers" minOccurs="0"/>
4144 <element name="target-additional-information"
4145 type="btp:additional-information" minOccurs="0"/>
4146 </sequence>
4147 <attribute name="id" type="ID"/>
4148 </complexType>
4149 </element>
4150
4151 <element name="fault" substitutionGroup="btp:message">
4152 <complexType>
4153 <sequence>
4154 <element name="superior-identifier" type="btp:identifier"
4155 minOccurs="0"/>
4156 <element name="inferior-identifier" type="btp:identifier"
4157 minOccurs="0"/>
4158 <element name="fault-type">
4159 <simpleType>
4160 <restriction base="string">
4161 <enumeration value="communication-failure"/>
4162 <enumeration value="duplicate-inferior"/>
4163 <enumeration value="general"/>
4164 <enumeration value="invalid-decider"/>
4165 <enumeration value="invalid-inferior"/>
4166 <enumeration value="invalid-superior"/>
4167 <enumeration value="status-refused"/>
4168 <enumeration value="invalid-terminator"/>
4169 <enumeration value="unknown-parameter"/>
4170 <enumeration value="unknown-transaction"/>
4171 <enumeration value="unsupported-qualifier"/>
4172 <enumeration value="wrong-state"/>
4173 </restriction>
4174 </simpleType>
4175 </element>
4176 <element name="fault-data" type="anyType" minOccurs="0"/>
4177 <element ref="btp:qualifiers" minOccurs="0"/>

```

```

4178         <element name="target-additional-information"
4179 type="btp:additional-information" minOccurs="0"/>
4180     </sequence>
4181     <attribute name="id" type="ID"/>
4182 </complexType>
4183 </element>
4184 <element name="enrol" substitutionGroup="btp:message">
4185     <complexType>
4186         <sequence>
4187             <element name="superior-identifier" type="btp:identifier"/>
4188             <element name="response-requested" type="boolean"/>
4189             <element name="reply-address" type="btp:address"
4190 minOccurs="0"/>
4191             <element name="inferior-address" type="btp:address"
4192 minOccurs="1" maxOccurs="unbounded"/>
4193             <element name="inferior-identifier" type="btp:identifier"/>
4194             <element ref="btp:qualifiers" minOccurs="0"/>
4195             <element name="target-additional-information"
4196 type="btp:additional-information" minOccurs="0"/>
4197         </sequence>
4198         <attribute name="id" type="ID"/>
4199     </complexType>
4200 </element>
4201
4202 <element name="enrolled" substitutionGroup="btp:message">
4203     <complexType>
4204         <sequence>
4205             <element name="inferior-identifier" type="btp:identifier"/>
4206             <element ref="btp:qualifiers" minOccurs="0"/>
4207             <element name="target-additional-information"
4208 type="btp:additional-information" minOccurs="0"/>
4209         </sequence>
4210         <attribute name="id" type="ID"/>
4211     </complexType>
4212 </element>
4213 <element name="resign" substitutionGroup="btp:message">
4214     <complexType>
4215         <sequence>
4216             <element name="superior-identifier" type="btp:identifier"/>
4217             <element name="inferior-identifier" type="btp:identifier"/>
4218             <element name="response-requested" type="boolean"/>
4219             <element ref="btp:qualifiers" minOccurs="0"/>
4220             <element name="target-additional-information"
4221 type="btp:additional-information" minOccurs="0"/>
4222         </sequence>
4223         <attribute name="id" type="ID"/>
4224     </complexType>
4225 </element>
4226
4227 <element name="resigned" substitutionGroup="btp:message">
4228     <complexType>
4229         <sequence>
4230             <element name="inferior-identifier" type="btp:identifier"/>
4231             <element ref="btp:qualifiers" minOccurs="0"/>

```

```

4232         <element name="target-additional-information"
4233 type="btp:additional-information" minOccurs="0"/>
4234     </sequence>
4235     <attribute name="id" type="ID"/>
4236 </complexType>
4237 </element>
4238
4239 <element name="prepare" substitutionGroup="btp:message">
4240 <complexType>
4241 <sequence>
4242     <element name="inferior-identifier" type="btp:identifier"/>
4243     <element ref="btp:qualifiers" minOccurs="0"/>
4244     <element name="target-additional-information"
4245 type="btp:additional-information" minOccurs="0"/>
4246 </sequence>
4247 <attribute name="id" type="ID"/>
4248 </complexType>
4249 </element>
4250 <element name="prepared" substitutionGroup="btp:message">
4251 <complexType>
4252 <sequence>
4253     <element name="superior-identifier" type="btp:identifier"/>
4254     <element name="inferior-identifier" type="btp:identifier"/>
4255     <element name="default-is-cancel" type="boolean"/>
4256     <element ref="btp:qualifiers" minOccurs="0"/>
4257     <element name="target-additional-information"
4258 type="btp:additional-information" minOccurs="0"/>
4259 </sequence>
4260 <attribute name="id" type="ID"/>
4261 </complexType>
4262 </element>
4263
4264 <element name="confirm" substitutionGroup="btp:message">
4265 <complexType>
4266 <sequence>
4267     <element name="inferior-identifier" type="btp:identifier"/>
4268     <element ref="btp:qualifiers" minOccurs="0"/>
4269     <element name="target-additional-information"
4270 type="btp:additional-information" minOccurs="0"/>
4271 </sequence>
4272 <attribute name="id" type="ID"/>
4273 </complexType>
4274 </element>
4275
4276 <element name="confirmed" substitutionGroup="btp:message">
4277 <complexType>
4278 <sequence>
4279     <element name="superior-identifier" type="btp:identifier"/>
4280     <element name="inferior-identifier" type="btp:identifier"/>
4281     <element name="confirmed-received" type="boolean"/>
4282     <element ref="btp:qualifiers" minOccurs="0"/>
4283     <element name="target-additional-information"
4284 type="btp:additional-information" minOccurs="0"/>
4285 </sequence>
4286 <attribute name="id" type="ID"/>

```

```

4287     </complexType>
4288 </element>
4289 <element name="cancel" substitutionGroup="btp:message">
4290     <complexType>
4291         <sequence>
4292             <element name="inferior-identifier" type="btp:identifier"/>
4293             <element name="reply-address" type="btp:address"
4294 minOccurs="0"/>
4295             <element ref="btp:qualifiers" minOccurs="0"/>
4296             <element name="target-additional-information"
4297 type="btp:additional-information" minOccurs="0"/>
4298         </sequence>
4299         <attribute name="id" type="ID"/>
4300     </complexType>
4301 </element>
4302 <element name="cancelled" substitutionGroup="btp:message">
4303     <complexType>
4304         <sequence>
4305             <element name="superior-identifier" type="btp:identifier"/>
4306             <element name="inferior-identifier" type="btp:identifier"
4307 minOccurs="0"/>
4308             <element ref="btp:qualifiers" minOccurs="0"/>
4309             <element name="target-additional-information"
4310 type="btp:additional-information" minOccurs="0"/>
4311         </sequence>
4312         <attribute name="id" type="ID"/>
4313     </complexType>
4314 </element>
4315
4316 <element name="confirm-one-phase" substitutionGroup="btp:message">
4317     <complexType>
4318         <sequence>
4319             <element name="inferior-identifier" type="btp:identifier"/>
4320             <element name="report-hazard" type="boolean"/>
4321             <element ref="btp:qualifiers" minOccurs="0"/>
4322             <element name="target-additional-information"
4323 type="btp:additional-information" minOccurs="0"/>
4324         </sequence>
4325         <attribute name="id" type="ID"/>
4326     </complexType>
4327 </element>
4328 <element name="hazard" substitutionGroup="btp:message">
4329     <complexType>
4330         <sequence>
4331             <element name="superior-identifier" type="btp:identifier"/>
4332             <element name="inferior-identifier" type="btp:identifier"/>
4333             <element name="level">
4334                 <simpleType>
4335                     <restriction base="string">
4336                         <enumeration value="mixed"/>
4337                         <enumeration value="possible"/>
4338                     </restriction>
4339                 </simpleType>
4340             </element>
4341             <element ref="btp:qualifiers" minOccurs="0"/>

```



```

4342         <element name="target-additional-information"
4343 type="btp:additional-information" minOccurs="0"/>
4344     </sequence>
4345     <attribute name="id" type="ID"/>
4346 </complexType>
4347 </element>
4348 <element name="contradiction" substitutionGroup="btp:message">
4349     <complexType>
4350         <sequence>
4351             <element name="inferior-identifier" type="btp:identifier"/>
4352             <element ref="btp:qualifiers" minOccurs="0"/>
4353             <element name="target-additional-information"
4354 type="btp:additional-information" minOccurs="0"/>
4355         </sequence>
4356         <attribute name="id" type="ID"/>
4357     </complexType>
4358 </element>
4359
4360 <element name="superior-state" substitutionGroup="btp:message">
4361     <complexType>
4362         <sequence>
4363             <element name="inferior-identifier" type="btp:identifier"/>
4364             <element name="status">
4365                 <simpleType>
4366                     <restriction base="string">
4367                         <enumeration value="active"/>
4368                         <enumeration value="prepared-received"/>
4369                         <enumeration value="inaccessible"/>
4370                         <enumeration value="unknown"/>
4371                     </restriction>
4372                 </simpleType>
4373             </element>
4374             <element name="response-requested" type="boolean"/>
4375             <element ref="btp:qualifiers" minOccurs="0"/>
4376             <element name="target-additional-information"
4377 type="btp:additional-information" minOccurs="0"/>
4378         </sequence>
4379         <attribute name="id" type="ID"/>
4380     </complexType>
4381 </element>
4382 <element name="inferior-state" substitutionGroup="btp:message">
4383     <complexType>
4384         <sequence>
4385             <element name="superior-identifier" type="btp:identifier"/>
4386             <element name="inferior-identifier" type="btp:identifier"/>
4387             <element name="status">
4388                 <simpleType>
4389                     <restriction base="string">
4390                         <enumeration value="active"/>
4391                         <enumeration value="inaccessible"/>
4392                         <enumeration value="unknown"/>
4393                     </restriction>
4394                 </simpleType>
4395             </element>
4396             <element name="response-requested" type="boolean"/>

```

```

4397         <element ref="btp:qualifiers" minOccurs="0"/>
4398         <element name="target-additional-information"
4399 type="btp:additional-information" minOccurs="0"/>
4400     </sequence>
4401     <attribute name="id" type="ID"/>
4402 </complexType>
4403 </element>
4404 <element name="redirect" substitutionGroup="btp:message">
4405     <complexType>
4406         <sequence>
4407             <element name="superior-identifier" type="btp:identifier"
4408 minOccurs="0"/>
4409             <element name="inferior-identifier" type="btp:identifier"
4410 />
4411             <element name="old-address" type="btp:address"
4412 minOccurs="unbounded"/>
4413             <element name="new-address" type="btp:address"
4414 minOccurs="unbounded"/>
4415             <element ref="btp:qualifiers" minOccurs="0"/>
4416             <element name="target-additional-information"
4417 type="btp:additional-information" minOccurs="0"/>
4418         </sequence>
4419         <attribute name="id" type="ID"/>
4420     </complexType>
4421 </element>
4422
4423 <element name="begin" substitutionGroup="btp:message">
4424     <complexType>
4425         <sequence>
4426             <element name="transaction-type" type="btp:superior-type"/>
4427             <element ref="btp:qualifiers" minOccurs="0"/>
4428             <element name="target-additional-information"
4429 type="btp:additional-information" minOccurs="0"/>
4430             <element name="reply-address" type="btp:address"
4431 minOccurs="0"/>
4432         </sequence>
4433         <attribute name="id" type="ID"/>
4434     </complexType>
4435 </element>
4436 <element name="begun" substitutionGroup="btp:message">
4437     <complexType>
4438         <sequence>
4439             <element name="decider-address" type="btp:address"
4440 minOccurs="0" maxOccurs="unbounded"/>
4441             <element name="transaction-identifier"
4442 type="btp:identifier" minOccurs="0"/>
4443             <element name="inferior-handleidentifier"
4444 type="btp:identifier" minOccurs="0"/>
4445             <element name="inferior-address" type="btp:address"
4446 minOccurs="0" maxOccurs="unbounded"/>
4447             <element ref="btp:qualifiers" minOccurs="0"/>
4448             <element name="target-additional-information"
4449 type="btp:additional-information" minOccurs="0"/>
4450         </sequence>
4451         <attribute name="id" type="ID"/>

```

```

4452         </complexType>
4453     </element>
4454     <element name="prepare-inferiors" substitutionGroup="btp:message">
4455         <complexType>
4456             <sequence>
4457                 <element name="transaction-identifier"
4458 type="btp:identifier"/>
4459                 <element name="inferiors-list" minOccurs="0">
4460                     <complexType>
4461                         <sequence>
4462                             <element name="inferior-handleidentifier"
4463 type="btp:identifier" maxOccurs="unbounded"/>
4464                         </sequence>
4465                     </complexType>
4466                 </element>
4467                 <element ref="btp:qualifiers" minOccurs="0"/>
4468                 <element name="target-additional-information"
4469 type="btp:additional-information" minOccurs="0"/>
4470                 <element name="reply-address" type="btp:address"
4471 minOccurs="0"/>
4472             </sequence>
4473             <attribute name="id" type="ID"/>
4474         </complexType>
4475     </element>
4476     <element name="confirm-transaction" substitutionGroup="btp:message">
4477         <complexType>
4478             <sequence>
4479                 <element name="transaction-identifier"
4480 type="btp:identifier"/>
4481                 <element name="inferiors-list" minOccurs="0">
4482                     <complexType>
4483                         <sequence>
4484                             <element name="inferior-handleidentifier"
4485 type="btp:identifier" maxOccurs="unbounded"/>
4486                         </sequence>
4487                     </complexType>
4488                 </element>
4489                 <element name="report-hazard" type="boolean"/>
4490                 <element ref="btp:qualifiers" minOccurs="0"/>
4491                 <element name="target-additional-information"
4492 type="btp:additional-information" minOccurs="0"/>
4493                 <element name="reply-address" type="btp:address"
4494 minOccurs="0"/>
4495             </sequence>
4496             <attribute name="id" type="ID"/>
4497         </complexType>
4498     </element>
4499     <element name="transaction-confirmed" substitutionGroup="btp:message">
4500         <complexType>
4501             <sequence>
4502                 <element name="transaction-identifier"
4503 type="btp:identifier"/>
4504                 <element ref="btp:qualifiers" minOccurs="0"/>
4505                 <element name="target-additional-information"
4506 type="btp:additional-information" minOccurs="0"/>

```

```

4507         </sequence>
4508         <attribute name="id" type="ID"/>
4509     </complexType>
4510 </element>
4511 <element name="cancel-transaction" substitutionGroup="btp:message">
4512     <complexType>
4513         <sequence>
4514             <element name="transaction-identifier"
4515 type="btp:identifier"/>
4516             <element name="report-hazard" type="boolean"/>
4517             <element ref="btp:qualifiers" minOccurs="0"/>
4518             <element name="target-additional-information"
4519 type="btp:additional-information" minOccurs="0"/>
4520             <element name="reply-address" type="btp:address"
4521 minOccurs="0"/>
4522         </sequence>
4523         <attribute name="id" type="ID"/>
4524     </complexType>
4525 </element>
4526
4527 <element name="cancel-inferiors" substitutionGroup="btp:message">
4528     <complexType>
4529         <sequence>
4530             <element name="transaction-identifier"
4531 type="btp:identifier" minOccurs="0"/>
4532             <element name="inferiors-list">
4533                 <complexType>
4534                     <sequence>
4535                         <element name="inferior-handleidentifier"
4536 type="btp:identifier" maxOccurs="unbounded"/>
4537                     </sequence>
4538                 </complexType>
4539             </element>
4540             <element ref="btp:qualifiers" minOccurs="0"/>
4541             <element name="target-additional-information"
4542 type="btp:additional-information" minOccurs="0"/>
4543             <element name="reply-address" type="btp:address"
4544 minOccurs="0"/>
4545         </sequence>
4546         <attribute name="id" type="ID"/>
4547     </complexType>
4548 </element>
4549 <element name="transaction-cancelled" substitutionGroup="btp:message">
4550     <complexType>
4551         <sequence>
4552             <element name="transaction-identifier"
4553 type="btp:identifier"/>
4554             <element ref="btp:qualifiers" minOccurs="0"/>
4555             <element name="target-additional-information"
4556 type="btp:additional-information" minOccurs="0"/>
4557         </sequence>
4558         <attribute name="id" type="ID"/>
4559     </complexType>
4560 </element>
4561

```

```

4562     <element name="request-inferior-statuses"
4563 substitutionGroup="btp:message">
4564     <complexType>
4565         <sequence>
4566             <element name="target-identifier" type="btp:identifier"/>
4567             <element name="inferiors-list" minOccurs="0">
4568                 <complexType>
4569                     <sequence>
4570                         <element name="inferior-handle"
4571 type="btp:identifier" maxOccurs="unbounded"/>
4572                     </sequence>
4573                 </complexType>
4574             </element>
4575             <element ref="btp:qualifiers" minOccurs="0"/>
4576             <element name="target-additional-information"
4577 type="btp:additional-information" minOccurs="0"/>
4578             <element name="reply-address" type="btp:address"
4579 minOccurs="0"/>
4580         </sequence>
4581         <attribute name="id" type="ID"/>
4582     </complexType>
4583 </element>
4584
4585     <element name="inferior-statuses" substitutionGroup="btp:message">
4586     <complexType>
4587         <sequence>
4588             <element name="responders-identifier"
4589 type="btp:identifier"/>
4590             <element name="status-list">
4591                 <complexType>
4592                     <sequence>
4593                         <element name="status-item" maxOccurs="unbounded">
4594                             <complexType>
4595                                 <sequence>
4596                                     <element name="inferior-handle-identifier"
4597 type="btp:identifier"/>
4598                                 <element name="status">
4599                                     <simpleType>
4600                                         <restriction base="string">
4601                                             <enumeration value="active"/>
4602                                             <enumeration value="resigned"/>
4603                                             <enumeration value="preparing"/>
4604                                             <enumeration value="prepared"/>
4605                                             <enumeration value="autonomously-confirmed"/>
4606                                             <enumeration value="autonomously-cancelled"/>
4607                                             <enumeration value="confirming"/>
4608                                             <enumeration value="confirmed"/>
4609                                             <enumeration value="cancelling"/>
4610                                             <enumeration value="cancelled"/>
4611                                             <enumeration value="cancel-contradiction"/>
4612                                             <enumeration value="confirm-contradiction"/>
4613                                             <enumeration value="hazard"/>
4614                                             <enumeration value="invalid"/>
4615                                         </restriction>
4616                                     </simpleType>

```

```

4617         </element>
4618         <element ref="btp:qualifiers" minOccurs="0"/>
4619     </sequence>
4620 </complexType>
4621 </element>
4622 </sequence>
4623 </complexType>
4624 </element>
4625     <element ref="btp:qualifiers" minOccurs="0"/>
4626     <element name="target-additional-information"
4627 type="btp:additional-information" minOccurs="0"/>
4628 </sequence>
4629     <attribute name="id" type="ID"/>
4630 </complexType>
4631 </element>
4632
4633 </schema>

```

XML schema for standard qualifiers

```

4635 <?xml version="1.0"?>
4636 <schema
4637     xmlns="http://www.w3.org/2001/XMLSchema"
4638     targetNamespace="urn:oasis:names:tc:BTP:1.0:qualifiers"
4639     xmlns:btpq="urn:oasis:names:tc:BTP:1.0:qualifiers"
4640     xmlns:btp="urn:oasis:names:tc:BTP:1.0:core"
4641     elementFormDefault="qualified">
4642
4643     <element name="transaction-timelimit"
4644 substitutionGroup="btp:qualifier">
4645         <complexType>
4646             <complexContent>
4647                 <extension base="btp:qualifier-type">
4648                     <sequence>
4649                         <element name="timelimit"
4650 type="nonNegativeInteger"/>
4651                     </sequence>
4652                 </extension>
4653             </complexContent>
4654         </complexType>
4655     </element>
4656     <element name="inferior-timeout" substitutionGroup="btp:qualifier">
4657         <complexType>
4658             <complexContent>
4659                 <extension base="btp:qualifier-type">
4660                     <sequence>
4661                         <element name="timelimit"
4662 type="nonNegativeInteger"/>
4663                         <element name="intended-decision">
4664                             <simpleType>
4665                                 <restriction base="string">
4666                                     <enumeration value="confirm"/>
4667                                     <enumeration value="cancel"/>
4668                                 </restriction>
4669                             </simpleType>

```

```

4670         </element>
4671     </sequence>
4672 </extension>
4673 </complexContent>
4674 </complexType>
4675 </element>
4676 <element name="minimum-inferior-timeout"
4677 substitutionGroup="btp:qualifier">
4678     <complexType>
4679         <complexContent>
4680             <extension base="btp:qualifier-type">
4681                 <sequence>
4682                     <element name="minimum-timeout"
4683 type="nonNegativeInteger"/>
4684                 </sequence>
4685             </extension>
4686         </complexContent>
4687     </complexType>
4688 </element>
4689 <element name="inferior-name" substitutionGroup="btp:qualifier">
4690     <complexType>
4691         <complexContent>
4692             <extension base="btp:qualifier-type">
4693                 <sequence>
4694                     <element name="inferior-name" type="string"/>
4695                 </sequence>
4696             </extension>
4697         </complexContent>
4698     </complexType>
4699 </element>
4700 </schema>

```

Carrier Protocol Bindings

The notion of bindings is introduced to act as the glue between the BTP 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.

Carrier Protocol Binding Proforma

A BTP carrier binding specification should provide the following information:

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

4718 elsewhere shall have binding names that are URIs. Bindings specified in this document use
4719 numbers to identify the version of the binding, not the version(s) of the carrier protocol.

4720 **Binding address format:** This section states the format of the “binding address” field of a BTP
4721 address for this binding. For many bindings, this will be a URL of some kind; for other bindings
4722 it may be some other form

4723 **BTP message representation:** This section will define how BTP messages are represented. For
4724 many bindings, the BTP message syntax will be as specified in the XML schema defined in this
4725 document, and the normal string encoding of that XML will be used.

4726 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that are
4727 not related to application messages are sent in either direction between Superior and Inferior. (i.e.
4728 those messages sent directly between BTP actors). This mapping need not be symmetric (i.e.
4729 Superior to Inferior may differ to some degree to Inferior to Superior). The mapping may define
4730 particular rules for particular BTP messages, or messages with particular parameter values (e.g.
4731 the FAULT message with “fault-type” “CommunicationFailure” will typically not be sent as a
4732 BTP message). The mapping states any constraints or requirements on which BTP may or must
4733 be bundled together by compounding.

4734 **Mapping for BTP messages related to application messages:** This section will define how
4735 BTP messages that are related to application messages are sent. A binding specification may defer
4736 details of this to a particular application (e.g. a mapping specification could just say “the
4737 CONTEXT may be carried as a parameter of an application invocation”). Alternatively, the
4738 binding may specify a general method that represents the relationship between application and
4739 BTP messages.

4740 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly but
4741 are treated as implicit in carrier-protocol mechanisms, application messages or other BTP
4742 messages. This may depend on particular parameter values of the BTP messages or the
4743 application messages.

4744 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
4745 protocol and of BTP shall be defined. This may include definition of which carrier protocol
4746 exceptions are equivalent to a FAULT/communication-failure message.

4747 **Relationship to other bindings:** Any relationship to other bindings is defined in this section. If
4748 BTP addresses with different bindings are to be considered to match (for purposes of identifying the
4749 peer Superior/Inferior and redirection), this should be specified here.

4750 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are imposed
4751 by use of this binding should be listed. This would include limitations on which messages can be
4752 sent, which event sequences are supported and restrictions on parameter values. Such limitations
4753 may reduce the usefulness of an implementation, but may be appropriate in certain environments.

4754 **Other:** Other features of the binding, especially any that will potentially affect interoperation
4755 should be specified here. This may include restrictions or requirements on the use or support of
4756 optional carrier parameters or mechanisms or use of standard or other qualifiers.

4757 **Bindings for request/response carrier protocols**

4758 BTP does not generally follow a request/response pattern. In particular, on the outcome
4759 relationship either side may initiate a message – this is an essential part of the presume-abort
4760 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
4761 messages, especially in the control relationship, that do have a request/response pattern. Many
4762 (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The specification of
4763 a binding specification to a request/response carrier protocol needs to state what rules apply –
4764 which messages can be carried by requests, which by responses. The simplest rule is to send all
4765 BTP messages on requests, and let the carrier responses travel back empty. This would be
4766 inefficient in use of network resources, and possibly inconvenient when used for the BTP
4767 request/response pairs.

4768 This section defines a set of rules that allow more efficient use of the carrier, while allowing the
4769 initiator of a BTP request/response pair to ensure the BTP response is sent back on the carrier
4770 response. These rules are specified in this section to enable binding specifications to reference
4771 them, without requiring each binding specification to repeat similar information. These rules also
4772 allow the receiver of a message between Superior and Inferior (in either direction) on a carrier
4773 protocol request to send any reply message on the carrier response – the “sender-address” field is
4774 implicitly considered to be that of the sender of the carrier request.

4775 A binding to a request/response carrier is not required to use these rules. It may define other rules.

4776 **Request/response exploitation rules**

4777 These rules allow implementations to use the request and response of the carrier protocol
4778 efficiently, and, when a BTP request/response exchange occurs, to either treat the
4779 request/response exchanges of the carrier protocol and of BTP independently, if both sides wish,
4780 or allow either side to map them closely.

4781 Under these rules, an implementation sending a BTP request (i.e. a message, other than
4782 CONTEXT, which has “reply-address” as a parameter in the abstract message definition), can
4783 ensure that it and the reply map to a carrier request/response by supplying no value for the “reply-
4784 address”. An implementation receiving such a request is required to send the BTP response on the
4785 carrier response.

4786 Conversely, if an implementation does supply a “reply-address” value on the request, the receiver
4787 has the option of sending the BTP response back on the carrier response, or sending it on a new
4788 carrier request.

4789 Within the outcome relationship, apart from ENROL, there is no “reply-address”, and the parties
4790 normally know each other’s “superior-address” and “inferior-address”. However, these messages
4791 have a “sender-address”, which is used when the receiver does not have knowledge of the peer. In
4792 this case, the “sender-address” is treated as the “reply-address” of the other messages – if the field
4793 is absent in a message on a carrier request, the “sender-address” is implicitly that of the request
4794 sender. Any message for the peer (including the three messages mentioned, FAULT but also any
4795 other valid message in the Superior:Inferior relationship) may be sent on the carrier response.
4796 Apart from this, both sides are permitted to treat the carrier request/response exchanges as
4797 opportunities for sending messages to the appropriate destination.

4798 The rules:

- 4799 a) A BTP actor **may** bundle one or more BTP messages and related groups that
4800 have the same binding address for their target in a single btp:messages and
4801 transmit this btp:messages element on a carrier protocol request. There is no
4802 restriction on which combinations of messages and groups may be so bundled,
4803 other than that they have the same binding address, and that this binding address
4804 is usable as the destination of a carrier protocol request.
- 4805 b) A BTP actor that has received a carrier protocol request to which it has not yet
4806 responded, and which has one or more BTP messages and groups whose binding
4807 address for the target matches the origin of the carrier request **may** bundle such
4808 BTP messages in a single btp:messages element and transmit that on the carrier
4809 protocol response.
- 4810 c) A BTP actor that has received, on a carrier protocol request, one or more BTP
4811 messages or related groups that require a BTP response and for which no “reply-
4812 address” was supplied, **must** bundle the responding BTP message and groups in a
4813 btp:messages element and transmit this element on the carrier protocol response
4814 to the request that carried the BTP request.
- 4815 d) A BTP actor that has received, on a carrier protocol request, one or more BTP
4816 messages or related groups that, as abstract messages, have a “sender-address”
4817 parameter but no “reply-address” was supplied and does not have knowledge of
4818 the peer address, **must** bundle the responding BTP message and groups in a
4819 btp:messages element and transmit this element on the carrier protocol response
4820 to the request that carried the BTP request. If the actor does have knowledge of
4821 the peer address it **may** send one or messages for the peer in the carrier protocol
4822 response, regardless of whether the binding address of the peer matches the
4823 address of the carrier protocol requestor.
- 4824 e) Where only one message or group is to be sent, it shall be contained within a
4825 btp:messages element, as a bundle of one element.
- 4826 f) A BTP actor that receives a carrier protocol request carrying BTP messages that
4827 do have a “reply-address”, or which initiate processing that produces BTP
4828 messages whose target binding address matches the origin of the request, **may**
4829 freely choose whether to use the carrier protocol response for the replies, or to
4830 send back an “empty carrier protocol response”, and send the BTP replies in a
4831 separately initiated carrier protocol request. The characteristics of an “empty
4832 carrier protocol response” shall be stated in the particular binding specification.
- 4833 g) A BTP actor that sends BTP messages on a carrier protocol request **must** be able
4834 to accept returning BTP messages on the corresponding carrier protocol response
4835 and, if the actor has offered an address on which it will receive carrier requests,
4836 must be able to accept “replying” BTP messages on a separate carrier protocol
4837 request.

4838 SOAP Binding

4839 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
4840 specification, using the SOAP literal messaging style conventions. If no application message is
4841 sent at the same time, the BTP messages are contained within the SOAP Body element. If
4842 application messages are sent, the BTP messages are contained in the SOAP Header element.

4843 **Binding name:** soap-http-1

4844 **Binding address format:** shall be a URL, of type HTTP.

4845 **BTP message representation:** The string representation of the XML, as specified in the XML
4846 schema defined in this document shall be used. The BTP XML messages are embedded in the
4847 SOAP message without the use of any specific encoding rules (literal style SOAP message);
4848 hence the encodingStyle attribute need not be set or can be set to an empty string.

4849 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be
4850 used.

4851 BTP messages sent on an HTTP request or HTTP response which is not carrying an application
4852 message, the messages are contained in a single btp:messages element which is the immediate
4853 child element of the SOAP Body element.

4854 An “empty carrier protocol response” sent after receiving an HTTP request containing a
4855 btp:messages element in the SOAP Body ~~and~~ when the implementation ~~BTP-actor~~ chooses just to
4856 reply at the lower level (and when the request/response exploitation rules allow an empty carrier
4857 protocol response), shall be any of:

- 4858 a) an empty HTTP response
- 4859 b) an HTTP response containing an empty SOAP Envelope
- 4860 c) an HTTP response containing a SOAP Envelope containing a single, empty
4861 btp:messages element.

4862 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they have
4863 no effect on the BTP sequence (other than indicating that the earlier sending did not cause a
4864 communication failure.)

4865 If an application message is being sent at the same time, the mapping for related messages shall
4866 be used, as if the BTP messages were related to the application message. (There is no ambiguity
4867 in whether the BTP messages are related, because only CONTEXT and ENROL can be related to
4868 an application message.)

4869 **Mapping for BTP messages related to application messages:** All BTP messages sent with an
4870 application message, whether related to the application message or not, shall be sent in a single
4871 btp:messages element in the SOAP Header. There shall be precisely one btp:messages element in
4872 the SOAP Header.

4873 The “request/response exploitation” rules shall apply to the BTP messages carried in the SOAP
4874 Header, as if they had been carried in a SOAP Body, unrelated to an application message, sent to
4875 the same binding address.

4876 *Note – The application protocol itself (which is using the SOAP Body) may use the SOAP*
4877 *RPC or document approach – this is determined by the application.*

4878 Only CONTEXT and ENROL messages are related (&) to application messages. If there is only
4879 one CONTEXT or one ENROL message present in the SOAP Header, it is assumed to be related
4880 to the whole of the application message in the SOAP Body. If there are multiple CONTEXT or
4881 ENROL messages, any relation of these BTP messages shall be indicated by application specific
4882 means.

4883 *Note 1 – An application protocol could use references to the ID values of the*
4884 *BTP messages to indicate relation between BTP CONTEXT or ENROL*
4885 *messages and the application message.*

4886 *Note 2 -- However indicated, what the relatedness means, or even whether it has*
4887 *any significance at all, is a matter for the application.*

4888 **Implicit messages:** A SOAP FAULT, or other communication failure received in response to a
4889 SOAP request that had a CONTEXT in the SOAP Header shall be treated as if a
4890 CONTEXT_REPLY/repudiated had been received. See also the discussion under “other” about
4891 the SOAP mustUnderstand attribute.

4892 **Faults:** A SOAP FAULT or other communication failure shall be treated as
4893 FAULT/communication-failure.

4894 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4895 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
4896 http-1” if the binding address and additional information fields match.

4897 **Limitations on BTP use:** None

4898 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
4899 attribute. The SOAPAction HTTP header is left to be application specific when there are
4900 application messages in the SOAP Body, as an already existing web service that is being
4901 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
4902 header shall contain no value when the SOAP message carries only BTP messages in the SOAP
4903 Body.

4904 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP
4905 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to determine
4906 whether any enrolments are necessary and replies with CONTEXT_REPLY as appropriate. The
4907 sender of the CONTEXT (and related application message) can use this to ensure that the
4908 application work is performed as part of the business transaction, assuming the receiver’s SOAP
4909 implementation supports the mustUnderstand attribute. If mustUnderstand is false, a receiver can
4910 ignore the CONTEXT (if BTP is not supported there), and no CONTEXT_REPLY will be
4911 returned. It is a local option on the sender (client) side whether the absence of a
4912 CONTEXT_REPLY is assumed to be equivalent to aCONTEXT_REPLY/ok (and the business
4913 transaction allowed to proceed to confirmation).

4914 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to
4915 enforce these requirements.

4916 Example scenario using SOAP binding

4917 The example below shows an application request with CONTEXT message sent from
4918 client.example.com (which includes the Superior) to services.example.com (Service).

```
4919
4920 <soap:Envelope
4921     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
4922     soap:encodingStyle="">
4923   <soap:Header>
4924     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core">
4925       <btp:context superior-type="atom">
4926         <btp:superior-address>
4927           <btp:binding>soap-http-1</btp:binding>
4928           <btp:binding-
4929 address>http://client.example.com/soaphandler</btp:binding-
4930 address>
4931           <btp:additional-information>btpengine</btp:additional-
4932 information>
4933         </btp:superior-address>
4934         <btp:superior-
4935 identifier>http://example.com/1001</btp:superior-identifier>
4936         <btp:qualifiers>
4937           <btpq:transaction-timelimit
4938 xmlns:btpq="urn:oasis:names:tc:BTP:1.0:qualifiers"><btpq:timelimit
4939 >1800</btpq:timelimit></btpq:transaction-timelimit>
4940           </btp:qualifiers>
4941         </btp:context>
4942       </btp:messages>
4943     </soap:Header>
4944     <soap:Body>
4945       <ns1:orderGoods
4946 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4947         <custID>ABC8329045</custID>
4948         <itemID>224352</itemID>
4949         <quantity>5</quantity>
4950       </ns1:orderGoods>
4951     </soap:Body>
4952   </soap:Envelope>
4953
```

4954 The example below shows CONTEXT_REPLY and a related ENROL message sent from
4955 services.example.com to client.example.com, in reply to the previous message. There is no
4956 application response, so the BTP messages are in the SOAP Body. The ENROL message does not
4957 contain the target-additional-information, since the grouping rules for CONTEXT_REPLY &
4958 ENROL omit the “target-address” (the receiver of this example remembers the superior address
4959 from the original CONTEXT)

```
4960 <soap:Envelope
4961     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
```

```

4962     soap:encodingStyle="">
4963     <soap:Header>
4964     </soap:Header>
4965     <soap:Body>
4966         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core">
4967             <btp:related-group>
4968                 <btp:context-reply>
4969                     <btp:target-additional-information>btpengine</btp:target-
4970 additional-information>
4971                 <btp:superior-
4972 identifier>http://example.com/1001</btp:superior-identifier>
4973                 <completion-status>related</completion-status>
4974                 </btp:context-reply>
4975                 <btp:enrol response-requested="false">
4976                     <btp:target-additional-
4977 information>btpengine</btp:target-additional-information>
4978                     <btp:superior-
4979 identifier>http://example.com/1001</btp:superior-identifier>
4980                     <btp:inferior-address>
4981                         <btp:binding>soap-http-1</btp:binding>
4982                         <btp:binding-address>
4983                             http://services.example.com/soaphandler
4984                         </btp:binding-address>
4985                     </btp:inferior-address>
4986                     <btp:inferior-identifier>
4987                         http://example.com/AAAB
4988                     </btp:inferior-identifier>
4989                 </btp:enrol>
4990             </btp:related-group>
4991         </btp:messages>
4992     </soap:Body>
4993 </soap:Envelope>
4994

```

4995 SOAP + Attachments Binding

4996 This binding describes how BTP messages will be carried using SOAP as in the [SOAP Messages](#)
4997 [with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-http-1. The two
4998 bindings only differ when application messages are sent.

4999 **Binding name:** soap-attachments-http-1

5000 **Binding address format:** as for soap-http-1

5001 **BTP message representation:** As for soap-http-1

5002 **Mapping for BTP messages (unrelated):** As for “soap-http-1”, except the SOAP Envelope
5003 containing the SOAP Body containing the BTP messages shall be in a MIME body part, as
5004 specified in [SOAP Messages with Attachments](#) specification. If an application message is being
5005 sent at the same time, the mapping for related messages for this binding shall be used, as if the
5006 BTP messages were related to the application message(s).

5007 **Mapping for BTP messages related to application messages:** MIME packaging shall be used.
5008 One of the MIME multipart/related parts shall contain a SOAP Envelope, whose SOAP Headers
5009 element shall contain precisely one btp:messages element, containing any BTP messages. Any
5010 BTP CONTEXT in the btp:messages is considered to be related to the application message(s) in
5011 the SOAP Body, and to also any of the MIME parts referenced from the SOAP Body (using the
5012 "href" attribute).

5013 **Implicit messages:** As for soap-http-1.

5014 **Faults:** As for soap-http-1.

5015 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
5016 string "soap-http-1" is considered to match one that has the binding string "soap-attachements-
5017 http-1" if the binding address and additional information fields match.

5018 **Limitations on BTP use:** None

5019 **Other:** As for soap-http-1

5020 *Example using SOAP + Attachments binding*

```
5021 Content-Type: Multipart/Related; boundary=MIME_boundary;  
5022 type=text/xml;  
5023     start="someID"  
5024 --MIME_boundary  
5025 Content-Type: text/xml; charset=UTF-8  
5026 Content-ID: someID  
5027 <?xml version='1.0' ?>  
5028 <soap:Envelope  
5029     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
5030     soap:encodingStyle=" ">  
5031     <soap:Header>  
5032         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core">  
5033             <btp:context superior-type="atom">  
5034                 <btp:superior-address>  
5035                     <btp:binding>soap-http-1</btp:binding>  
5036                     <btp:binding-address>  
5037                         http://client.example.com/soaphandler  
5038                     </btp:binding-address>  
5039                 </btp:superior-address>  
5040                 <btp:superior-  
5041 identifier>http://example.com/1001</btp:superior-identifier>  
5042             </btp:context>  
5043         </btp:messages>  
5044     </soap:Header>  
5045     <soap:Body>  
5046         <orderGoods href="cid:anotherID"/>  
5047     </soap:Body>  
5048 </soap:Envelope>  
5049 --MIME_boundary  
5050 Content-Type: text/xml  
5051 Content-ID: anotherID
```

```

5052     <ns1:orderGoods
5053 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
5054     <custID>ABC8329045</custID>
5055     <itemID>224352</itemID>
5056     <quantity>5</quantity>
5057 </ns1:orderGoods>
5058
5059 --MIME_boundary--

```

5060 Conformance

5061 A BTP implementation need not implement all aspects of the protocol to be useful. The level of
5062 conformance of an implementation is defined by which roles it can support using the specified
5063 messages and carrier protocol bindings for interoperation with other implementations.

5064 [An implementation may implement some roles and relationships in accordance with this](#)
5065 [specification, while providing the \(approximate\) functionality of other roles in some other](#)
5066 [manner. \(For example, an implementation might provide an equivalent of the control](#)
5067 [relationships using a language-specific API, but support roles involved in the outcome](#)
5068 [relationships using standard BTP messages.\) Such an implementation is conformant in respect of](#)
5069 [the roles it does implement in accordance with this specification.](#)

5070 ~~An implementation may implement the functionality of some roles in a non-interoperable way—~~
5071 ~~usually combining pairs of roles, such as Terminator and Decider. Such an implementation is~~
5072 ~~conformant in respect of the roles it does implement in accordance with this specification.~~

5073 An implementation can state which aspects of the BTP specification it conforms to in terms of
5074 which Roles it supports. Since most Roles cannot usefully be supported in isolation, the following
5075 Role Groups can be used to describe implementation capabilities:.

Role Group	Roles
Initiator/Terminator	Initiator Terminator
Cohesive Hub	Factory Composer (as Decider and Superior) Coordinator (as Decider and Superior) Sub-composer Sub-coordinator
Atomic Hub	Factory Coordinator Sub-coordinator
Cohesive Superior	Composer (as Superior only) Sub-Composer Coordinator (as Superior only) Sub-coordinator

Role Group	Roles
Atomic Superior	Coordinator (as Superior only) Sub-coordinator
Participant	Inferior Enroller

5076
5077 The Role Groups occupy different positions within a business transaction tree and thus require
5078 presence of implementations supporting other Role Groups:

5079 Initiator/Terminator uses control relationship to Atomic Hub or Cohesive Hub to initiate
5080 and control Atoms or Cohesions. Initiator/Terminator would typically be a library linked
5081 with application software.

5082 Atomic Hub and Cohesive Hub would often be standalone servers.

5083 Cohesive Superior and Atomic Superior would provide the equivalent of
5084 Initiator/Terminator functionality by internal or proprietary means.

5085 Cohesive Hubs, Atomic Hubs, Cohesive Superior and Atomic Superior use outcome
5086 relationships to Participants and to each other.

5087 Participants will establish outcome relationships to implementations of any of the other
5088 Role Groups except Initiator/Terminator. A Participant “covers” a resource or application
5089 work of some kind. It should be noted that a Participant is unaffected by whether it is
5090 enrolled in an Atom or Cohesion – it gets only a single outcome.

5091 An implementation may support one or more Role Groups. The following combinations are
5092 defined as commonly expected conformance profiles, although other combinations or selections
5093 are equally possible.

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

5094

5095 BTP has several features, such as optional parameters, that allow alternative implementation
5096 architectures. Implementations should pay particular attention to avoid assuming their peers have
5097 made the same implementation options as they have (e.g. an implementation that always sends
5098 ENROL with the same inferior address and with the “reply-address” absent (because the Inferior
5099 in all transactions are dealt with by the same addressable entity), must not assume that the same is
5100 true of received ENROLs)

5101

5101

Part 3. Glossary

5102

Actor

An entity that executes procedures, a software agent. (See also BTP Actor)

Address

An identifier for an endpoint.

Application

An actor, which uses the Business Transaction Protocol (in the context of this specification).

Also, a group of such actors, which may be distributed, that perform a common purpose.

(When used in phrases such as “determined by the Application”, it is not relevant to BTP whether this is determined by the owner of a single system or is explicitly part of the contract that defines the distributed collaborative application. When it is necessary to distinguish the responsibilities of a single party, the term “Application element” is used.)

Application element

An actor that communicates, using application protocols, with other application elements, as part of an overall distributed application. A single system may contain more than one application element.

Application Endpoint

An endpoint of an application message.

Application Message

A message produced by an application element and consumed by an application element.

Application Operation

An operation, which is started when an application message arrives.

Appropriate

In accordance with a pertinent contract or specification.

Atom

A set of participants, which are the direct inferiors of a node (which may have only one member), all of which will receive instructions that will result in a homogeneous outcome. That is they will be issued instructions to all confirm or all cancel. (Transitively, a set of operations whose effect is capable of counter effect.)

Atomic Business Transaction

A complete business transaction that follows the atom rules for every node in the transaction tree over space and time, so that all the participants in the transaction will receive instructions that will result in a homogeneous outcome. That is they will be issued instructions to all confirm or all cancel. (Transitively, a set of operations whose effect is capable of counter effect.)

Become prepared

Ensure that of a set of procedures is capable of being successfully instructed to cancel or to confirm.

BTP Actor

A software entity, or agent, that is able to take part in Business Transaction Protocol exchanges i.e. that sends or receives BTP messages. A BTP Actor may be capable of only playing a single role, or of playing several different roles concurrently and / or sequentially. A BTP Actor may be involved in one, or more, transactions, concurrently and / or sequentially.

BTP element

A BTP actor that supports an application element (or elements) but is not itself concerned with application messages or semantics.

(Business) Application Protocol

The messages, their meanings and their permitted sequences used to effect a change in the state of a business relationship.

(Business) application system

A system that contains one, or more, business applications, and resources such as volatile and persistent storage for business state information. It may also contain other things such as an operating system and BTP elements.

Business relationship agreement

The contract and / or set of agreements that govern and constrain a business relationship between two, or more, parties.

Business relationship

A *business relationship* is any distributed state held by the parties, which is subject to contractual constraints agreed by those parties.

Business Transaction Protocol (BTP)

The messages, their meanings and their permitted sequences defined in this specification. Its purpose is to provide the interactions (or signalling) required to coordinate the effects of application protocol to achieve a business transaction.

BTP-Address

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

Business transaction

A set of state changes that occur, or are desired, in computer systems controlled by some set of parties, and these changes are related in some application defined manner. A *business transaction* is subject to, and a part of, a *business relationship*. (BTP assumes that the parties involved in a *business transaction* have distinct and autonomous application systems, which do not require knowledge of each others’ implementation or internal state representations in volatile or persistent storage. Access to such loosely coupled systems is assumed to occur only through service interfaces.)

Cancel

Process a counter effect for the current effect of a set of procedures. There are a number of different ways that this may be achieved in practice.

Carrier Protocol

A protocol, which defines how the transmission of BTP messages occur.

**Carrier Protocol Address
(CPA)**

The address of an endpoint for a particular carrier protocol.

Client

An actor, which sends application messages to services.

Cohesion

A set of participants, which are the direct inferiors of a node that may receive instructions that may result in different outcomes for each participant. That is they will be issued instructions to confirm or cancel according to the application logic. Participants may resign or be instructed to cancel until the confirm set is fixed. Once the confirm set for a cohesion is fixed, then all participants in the confirm set are treated atomically. That is they will all be instructed to confirm unless one, or more, cancel in which case all will be instructed to cancel. All participants not in the confirm set will be instructed to cancel.

Cohesive Business Transaction	A complete business transaction for which at least one node over space and time follows the cohesion rules. The other nodes in the transaction tree of a cohesive business transaction may follow either the cohesion rules or the atom rules.
Confirm	Ensure that the effect of a set of procedures is completed. There are a number of different ways that this may be achieved in practice.
Context	Information pertinent to a single transaction, or branch of a transaction.
Contract	Any rule, agreement or promise which constrains an actor's behaviour and is known to any other actor, and upon which any other knowing actor may rely.
Control relationship	The application element:BTP element relationships that create the nodes of the transaction tree (Initiator:Factory) and drive the completion (Terminator:Decider).
Coordinator	A BTP actor, which is the top 'node' of a transaction and decides the outcome of its immediate branches according to the atom rules defined in this specification. It has a lifetime, which is coincident with that of the atom. A coordinator can issue instructions to prepare, cancel and confirm. These instructions take the form of BTP messages. A coordinator is identified by its transaction-identifier. A coordinator must also have a BTP Address to which participants can send BTP messages.
Counter effect	An appropriate effect intended to counteract a prior effect.
Counter effect contract	The contract, which governs the relationship between the effect and the counter effect of a procedure. In the absence of any other overriding contracts the counter effect contract is the promise that the Counter effect will attempt so far as is possible to reverse or cancel the Effect such that an observer (on completion of the Counter effect) is unaware that the Effect ever occurred, but this attempt cannot be guaranteed to succeed.

Decider	<p>The top node of a transaction tree, a composer or a coordinator (so called because the Terminator can only request confirmation – the Decider makes the final determination). The term can always be interpreted as “Composer or Coordinator”.</p> <p>It is the role at the other end of a control relationship to a Terminator.</p>
Delivery parameter	<p>A parameter of an abstract message that is concerned with the transmission of the message to its target or the transmission of an immediate reply.. Distinguished from Payload parameter.</p>
Effect	<p>The changes induced by the incomplete or complete processing of a set of procedures by an actor, which are observable by another contemporary or future actor, and which are made in conformance with a contract known to any such observer. This contract must state the counter effect of the effect, and this is known as a counter effect contract. An effect is Completed when the change inducing processing of the set of procedures is finished.</p>
Endpoint	<p>A sender or receiver.</p>
Enroller	<p>The BTP Actor role that informs a superior of the existence of an inferior.</p>
Factory	<p>The BTP Actor role that creates transaction contexts and deciders.</p>
Inappropriate	<p>In violation of a pertinent contract or specification.</p>
Ineffectual	<p>Describes a set of procedures, which has no effect.</p>
Inferior	<p>The end of end of a BTP node to BTP node relationship governed by the outcome protocol that is topologically further from the top of the transaction tree.</p>
Inferior-Address	<p>The address used to communicate with an actor playing the role of an Inferior.</p>
Inferior-identifier	<p>A globally unambiguous identification of a particular Inferior within a single transaction (represented as an URI or equivalent).</p>
Initiator	<p>The BTP Actor role (an application element) that starts a transaction.</p>

Intermediate	A node that is a sub-composer or a sub-coordinator. An alternative term to interposed.
Interposed	A node that is a sub-composer or a sub-coordinator. An alternative term to intermediate.
Message	A datum, which is produced and then consumed.
Node	A logical entity that is associated with a single transaction. A node is a composer, a coordinator, a sub-coordinator, a sub-composer, or a participant.
Operation	A procedure, which is started by a receiver when a message arrives at it.
Outcome	A decision to either cancel or confirm.
Outcome relationship	The Superior:Inferior relationship (i.e. between BTP actors within the transaction tree) and the Enroller:Superior relationship used in establishing it.
Participant	A participant is part of an application system that also contains one, or more, applications, which manipulate resources. It is a role of a BTP Actor that is (or is equivalent to) a set of procedures, which is capable of receiving instructions from another BTP Actor to prepare, cancel and confirm. These signals are used by the application(s) to determine whether to effect (confirm) or counter effect (cancel) the results of application operations. A participant must also have a BTP Address, to which these instructions will be delivered, in the form of BTP messages. A participant is identified by an inferior-identifier.
Payload parameter	A parameter of an abstract message that is will be received and processed or retained by the receiving BTP actor. The various identifier parameters are considered Payload parameters . Distinguished from Delivery parameter.
Peer	The other party in a two-party relationship, as in Superior to Inferior, or Sender to Receiver.
Provisional Effect	The changes induced by the incomplete or complete processing of a set of procedures by an actor, which are subject to later completion or counter-effecting. The provisional effect may or may not be observable by other actors.
Receiver	The consumer of a message.

Relationship parties	The legal entities that enter into an agreement that forms the basis of the relationship.
Responders-identifier	An identifier carried in a BTP message that can be interpreted as transaction-identifier, a superior-identifier, or an inferior-identifier according to the nature of the role in a BTP actor that is responding to a received message.
Role	The participation of a software agent in a particular relationship in a particular business transaction. The software agent performing a role is termed an Actor .
Sender	The producer of a message.
Service	An actor (an application element), which on receipt of application messages, may start an appropriate application operation. For example, a process that advertises an interface allowing defined RPCs (remote procedure calls) to be invoked by a remote client.
Status requestor	The BTP Actor role that requests the status of another BTP actor.
Sub-composer	An actor, which is not the top 'node' of a transaction. It receives an outcome from its superior and decides the outcome of its immediate branches according to the cohesive rules defined in this specification. It has a lifetime, which is coincident with that of the cohesion. A sub-composer can issue instructions to prepare, cancel and confirm on individual branches. These instructions take the form of BTP messages. A sub-composer must also have at least one BTP Address to which lower nodes can send BTP messages.
Sub-coordinator	An actor, which is not the top 'node' of a transaction. It receives an outcome from its superior and propagates the outcome to its immediate branches according to the atom rules defined in this specification. It has a lifetime, which is coincident with that of this atom. A sub-coordinator can issue instructions to prepare, cancel and confirm. These instructions take the form of BTP messages. A sub-coordinator must also have at least one BTP Address to which lower nodes can send BTP messages.

Superior	<p>The BTP role that will accept enrolments of Inferiors and subsequently inform the Inferior of the Outcome applicable to it.</p> <p>A Superior will be one of Composer, Coordinator, Sub-composer, or Sub-coordinator.</p> <p>A Superior is considered to be a Superior even if it currently has no enrolled Inferiors.</p>
Superior-address	<p>The set of BTP-addresses used to communicate with an actor playing the role of a Superior.</p>
Superior-identifier	<p>A globally unambiguous identifier of a particular Superior within a particular transaction (represented as an URI or equivalent).</p>
Target-identifier	<p>An identifier carried in a BTP message that can be interpreted as transaction-identifier, a superior-identifier, or an inferior identifier according to the nature of the role in a BTP actor that receives this identifier.</p>
Terminator	<p>A BTP role performed by an Application element communicating with a Decider to control the completion of the Business Transaction. Frequently will be identical to the Initiator, but distinguished because the control of the Business Transaction can be passed between Application elements.</p>
Transaction	<p>A complete unit of work as defined by an application. A transaction starts when a part of the distributed transaction first initiates some work that is to be a part of a new transaction. The transaction tree may grow and shrink over time and (logical) space. A transaction completes when all the participants in a transaction have completed (that is have replied to their confirm or cancel instruction).</p>
Transaction tree	<p>A pattern of BTP nodes that provides the coordination of a distributed application transaction. There is single top node (a Decider) that interacts with the initiating application (which is a part of a distributed application). The Decider node has one, or more outcome relationships with other BTP nodes (sub-composer, sub-coordinator, or participant nodes). Any intermediate nodes (Sub-composer or Sub-coordinator nodes) have exactly one relationship up the tree in which they act as Inferior, and one, or more, relationships down the tree in which they act as Superior. Participants are leaves of the tree. That is they have exactly one relationship up the tree in which they act as Inferior and no down tree relationships.</p>

Transaction-identifier

A globally unambiguous identifier for a particular a Decider(represented as an URI or equivalent). A Decider is the top 'node' of the transaction and thus this identifier also unambiguously identifies the transaction. Often identical to the Superior-identifier of the Decider in its role as Superior, though the protocol does not require this.

Transmission

The passage of a message from a sender to a receiver.

5103