Business Transaction Protocol

3 4

1

2

5

6 7

10 11 An OASIS Committee Specification

CURRENT STATUS : internal committee draft

Version 1.0 [0.9.0.1] DD Mmm 2001 *16 November 2001*]

Working draft 0.1 (pre-London)	14 June 2001
Working draft 0.2 (London)	18 June 2001
Working draft 0.3a (circulated)	12 July 2001
Working draft 0.3b (not circulated)	17 July 2001
Working draft 0.3c (circulated)	20 July 2001
Working draft 0.4 (circulated; incorporates PRF material)	25 July 2001
Working draft 0.5 (uncirculated)	8 August 2001
Working draft 0.6 (State tables)	31 August 2001
Working draft 0.7 (revised abs msgs) – (not circulated)	28 September 2001
Working draft 0.72 (completed abs msg revsn 0.7)	3 October 2001
Working draft 0.80 – full scope, PRF handback	18 October 2001
Working Draft 0.9	24 October 2001
Working Draft 0.90.1 – minor editorials issues applied	16 November 2001

Copyright and related notices

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

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

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

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

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

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

53 54	Acknowledgements
55 56	Employees of the following companies participated in the finalization of this specification as members of the OASIS Business Transactions Technical Committee:
57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82	BEA Systems, Inc. Bowstreet, Inc. Choreology Ltd. Entrust, Inc. Hewlett-Packard Co. Interwoven Inc. IONA Technologies PLC SeeBeyond Inc. Sun Microsystems Computer Corp. Talking Blocks Inc. The primary authors and editors of the main body of the specification were: Alex Ceponkus (alex@ceponkus.org) Peter Furniss (peter.furniss@choreology.com) Alastair Green (alastair.green@choreology.com) Additional contributions to its writing were made by Sanjay Dalal (sanjay.dalal@bea.com) Mark Little (mark_little@hp.com) We thank Pal Takacsi-Nagy of BEA Systems Inc for his efforts in chairing the Technical Committee, and Karl Best of OASIS for his guidance on the organization of the Committee's work.
83 84	
85 86 87	In memory of Ed Felt
88 89 90 91 92 93 94 95 96	Ed Felt of BEA Systems Inc. was an active and highly valued contributor to the work of the OASIS Business Transactions Technical Committee. His many years of design and implementation experience with the Tuxedo system, Weblogic's Java transactions, and Weblogic Integration's Conversation Management Protocol were brought to bear in his comments on and proposals for this specification. He was killed in the crash of the hijacked United Airlines flight 93 near to Pittsburgh, on 11 September 2001.
97	

OASIS BTPDraft Specification 0.90.1, 16 November 2001

•	pographical and Linguistic Conventions and Style
101 102	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:
103 104 105 106	Cancel Participant Application Message
	The first occurrence of a word defined in the Glossary is given in bold, thus:
109 110	Coordinator
113	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:
116 117 118 119	BEGIN CONTEXT RESIGN
	The values of elements within a BTP protocol message are indicated thus:
122 123	BEGIN/atom
	BTP protocol messages that are related semantically are joined by an ampersand:
126 127	BEGIN/atom & CONTEXT
128 129 130	BTP protocol messages that are transmitted together in a compound are joined by a + sign:
131 132	ENROL + VOTE
	XML schemata and instances are given in Courier:
135 136	
	Illustrative fragments of code in other languages, such as Java, are given in Lucida Console:
139 140 141	<pre>int main (String[] args) { }</pre>
144	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:

147	An Inferior must send one of RESIGN, PREPARED or CANCELLED to its
148	Superior.
149	
150	

Contents

Contents	
Copyright and related notices	2
• • • • • • • • • • • • • • • • • • • •	
Part 1. Purpose and Features of BTP	9
Part 2. Normative Specification of BTP	14
CONTEXT	29
CONTEXT_REPLY	30
DDED A DE	26
	Copyright and related notices Acknowledgements Typographical and Linguistic Conventions and Style Contents. Part 1. Purpose and Features of BTP

197	PREPARED	38
198	CONFIRM	39
199	CONFIRMED	40
200	CANCEL	42
201	CANCELLED	
202	CONFIRM ONE PHASE	
203	HAZARD	
204	CONTRADICTION	
205	SUPERIOR STATE	
206	INFERIOR_STATE	
207	REQUEST CONFIRM	
208	REQUEST_STATUSES	
209	INFERIOR_STATUSES	
210	REQUEST_STATUS	
211	STATUS	
212	REDIRECT	
213	FAULT	
214	Standard qualifiers	
215	Transaction timelimit	
216	Inferior timeout	
217	Minimum inferior timeout	
218	Inferior name	
219	State Tables.	
220	Explanation of the state tables	
221	Status queries	
222	Decision events	
223	Disruptions – failure events	
224	Invalid cells and assumptions of the communication mechanism	
225	Meaning of state table events.	
226	Persistent information	
227	Failure Recovery	
228	Types of failure	
229	Persistent information	
230	Redirection	
231	Terminator:Decider failures	
232	XML representation of Message Set.	
233	Addresses.	
234	Qualifiers.	
235	Identifiers.	
236	Message References.	
237	Messages	
238	CONTEXT	
239	CONTEXT-REPLY	
240	BEGIN	
241	BEGUN	
242	ENROL	
243	ENROLLED	
244 244	RESIGN	

245	RESIGNED	90
246	PREPARE	
247	PREPARED	91
248	CONFIRM	91
249	CONFIRMED	91
250	CANCEL	92
251	CANCELLED	92
252	HAZARD	93
253	CONTRADICTION	93
254	SUPERIOR_STATE	93
255	INFERIOR_STATE	94
256	CONFIRM_ONE_PHASE	94
257	REQUEST_CONFIRM	94
258	REQUEST_STATUSES	
259	INFERIOR_STATUSES	95
260	REQUEST_STATUS	
261	STATUS	96
262	REDIRECT	
263	FAULT	
264	Standard qualifiers	
265	Transaction timelimit	
266	Inferior timeout	
267	Minimum inferior timeout	
268	Compounding of Messages	
269	Carrier Protocol Bindings	
270	Carrier Protocol Binding Proforma	
271	SOAP Binding	
272	Example scenario using SOAP binding	
273	SOAP + Attachments Binding	
274	XML Schema for SOAP Bindings	
275	Conformance	118
276	Part 3. Appendices	120
277	A. Glossary	
278		

Part 1. Purpose and Features of BTP

Introduction

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

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

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

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

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

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

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

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 334 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 • 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

Overview of the Business Transaction Protocol

A Business Transaction is a consistent change in the state of a business relationship between two or more parties. BTP provides means to allow the 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 some set of parties, and that these changes are related in some business-defined manner.

Typically business-defined messages ("application messages") are exchanged between the parties to the transaction, which result in the performance of some set of operations. These operations create provisional or tentative state changes (the transaction's effect). The provisional changes of each party must either be confirmed (given final effect), or must be cancelled (counter-effected). Those parties which are confirmed create an atomic unit, within which the business transaction has a consistent final effect.

The meaning of "effect", "final effect" and "counter-effect" is specific to each business transaction and to each party's role within it. A party may log intended changes (as its effect) and only process them as visible state changes on confirmation (its final effect). Or it may make visible state changes and store the information needed to cancel (its effect), and then simply delete the information needed for cancellation (its final effect). A 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.

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

determine whether it is able both to cancel (counter-effect) and to confirm (give final effect to) its effect

□ report its ability or inability to cancel-or-confirm (its preparedness) to a central coordinating entity

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

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

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

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

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

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

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

It should be noted that this BTP specification recognises that:

an Inferior may itself be a Superior to other BTP Inferiors; this occurs when some of the operations associated with the Inferior involve other application elements whose operations are to be subject to the confirm/cancel instruction sent to the Inferior. The specification treats any lower Inferiors as part of the associated operations;

the requirement on an Inferior to be able to confirm or cancel does not include any specific mechanism to determine the isolation of the effects of operations; the requirement is only that the Inferior is able to confirm or cancel the operations, as their effects are known to the Superior and the application directly in contact with the Superior. Thus the confirm-or-cancel requirement may be achieved by performing all the operations and remembering a compensating counter operation (that will be triggered by a cancel order); or by remembering the operations (having checked they

455 456 457	are valid) and performing them only if a confirm order is received; or by forbidding any other access to data changed by the operations and releasing them in their unchanged state (if cancelled) or their changed state (if confirmed); or by various
458	combinations of these. In addition, a cancellation may not return data to their original
459	state, but only to a state accepted by the application as appropriate to a cancelled
460	operation.
461	
462	
463	
464	
465	
466	
467	

Part 2. Normative Specification of BTP

Actors, Roles and Relationships

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

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

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

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

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

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

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

Relationships

There are two primary relationships in BTP.

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

510		
511 512 513		Between BTP actors within the tree, where one (the Superior) will inform the other (the Inferior) what the outcome decision is.
514 515 516	busines	orimary relationships are involved in arriving at a decision on the outcome of a stransaction, and propagating that decision to all parties to the transaction. Taking the at is followed when a business transaction is confirmed:
517 518	1.	The Terminator determines that the business transaction should confirm, if it can; or (for a Cohesion), which parts should confirm
519 520	2.	The Terminator asks the Decider to apply the desired outcome to the tree, if it can guarantee the consistency of the confirm decision
521 522	3.	The Decider, which is Superior to one or more Inferiors, asks its Inferiors if they can agree to a confirm decision (for a Cohesion, this may not be all the Inferiors)
523 524	4.	If any of those Inferiors are also Superiors, they ask their Inferiors and so on down the tree
525	5.	Inferiors that are not Superiors report if they can agree to a confirm to their Superior
526 527	6.	Inferiors that are also Superiors report their agreement only if they received such agreement from their Inferiors, and can agree themselves
528 529 530 531	7.	Eventually agreement (or not) is reported to the Decider. If all have agreed, the Decider makes and persists the confirm decision (hence the term "Decider" – it decides, everything else just asked); if any have disagreed, or if the confirm decision cannot be persisted, a cancel decision is made
532	8.	The Decider, as Superior tells its Inferiors of the outcome
533	9.	Inferiors that are also Superiors tell their Inferiors, recursively down the tree
534 535 536	10.	The Decider replies to the Terminator's request to confirm, reporting the outcome decision
537 538 539		re other relationships that are secondary to Terminator:Decider, Superior:Inferior, involved in the establishment of the primary relationships.
540 541 542 543	Inferior	o primary relationships are linked in that a Decider is a Superior to one or more s. There are also similarities in the semantics of some of the exchanges (messages) the relationships. However they differ in that
544 545 546 547	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
548 549 550	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

552 553 554 555 556	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).
557 558 559 560 561	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.
562 563	Roles involved in the Superior:Inferior relationship
564 565	Superior
566 567 568 569 570 571 572 573 574	Accepts enrolments from Inferiors, 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.
576 577 578	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.
579 580 581 582 583	A Superior may be <i>Atomic</i> or <i>Cohesive</i> ; 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".
584 585 586	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.
587 588	A Superior receives
589 590	ENROL
591 592	to enrol a new Inferior, establishing a new Superior:Inferior relationship.
593 594	A Superior sends
595 596	ENROLLED

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

597598599

A Superior sends

600	
601	PREPARE
602	CONFIRM
603	CANCEL
604	RESIGNED
605	CONFIRM_ONE_PHASE
606	SUPERIOR_STATE
607	***************************************
608	to an enrolled Inferior.
609	to an emoned finerior.
610	A Superior receives
611	A Superior receives
612	PREPARED
613	CANCELLED
614	CONFIRMED
615	HAZARD
616	RESIGN
617	INFERIOR_STATE
618	
619	from an enrolled Inferior.
620	
621	Inferior
622	
623	Responsible for applying the Outcome to some set of associated operations – the application
624	determines which operations are the responsibility of a particular Inferior.
625	
626	An Inferior is Enrolled with a single Superior (hereafter referred to as "its Superior"),
627	establishing a Superior:Inferior relationship. If the Inferior is able to ensure that either a
628	confirm or cancel decision can be applied to the associated operations, and can persist
629	information to retain that condition, it sends a PREPARED message to the Superior. When
630	the Outcome is received from the Superior, the Inferior applies it, deletes the persistent
631	information, and replies with CANCELLED or CONFIRMED as appropriate.
632	information, and represent the residence of the residence
633	If an Inferior is unable to come to a prepared state, it cancels the associated operations and
634	informs the Superior with a CANCELLED message. If it is unable to either come to a
635	prepared state, or to cancel the associated operations, it informs the Superior with a
	• •
636	HAZARD message.
637	A - Tuforion that has become annual many assertionally make an autonomous desirion to be
638	An Inferior that has become prepared may, exceptionally, make an autonomous decision to be
639	applied to the associated operations, without waiting for the Outcome from the Superior. It is
640	required to persist this autonomous decision and report it to the Superior with CONFIRMED
641	or CANCELLED as appropriate. If, when CONFIRM or CANCEL is received, the
642	autonomous decision and the decision received from the Superior are contradictory, the
643	Inferior must retain the record of the autonomous decision until receiving a
644	CONTRADICTION message.
645	
646	An Inferior receives
647	

648	PREPARE
649	CONFIRM
650	CANCEL
651	RESIGNED
652	CONFIRM_ONE_PHASE
653	SUPERIOR_STATE
654	
655	from its Superior.
656	
657	An Inferior sends
658	
659	PREPARED
660	CANCELLED
661	CONFIRMED
662	HAZARD
663	RESIGN
664	INFERIOR_STATE
665	
666	to its Superior.
667	1
668	An Inferior receives REQUEST_STATUS and replies with STATUS. If it is also a Superior,
669	the STATUS concerns the Inferior as a whole.
670	
671	Enroller
672	
673	Causes the enrolment of an Inferior with a Superior. This role is distinguished because in
674	some implementations the enrolment request will be performed by the application, in some
675	the application will ask the actor that will play the role of Inferior to enrol itself, and a
676	Factory may enrol a new Inferior (which will also be Superior) as a result of receiving
677	BEGIN&CONTEXT.
678	2201 (40 01 1211)
679	An Enroller sends
680	
681	ENROL
682	
683	to a Superior.
684	
685	An Enroller receives
686	
687	ENROLLED
688	
689	in reply to ENROL if the Enroller asked for a response when the ENROL was sent.
690	1 5
691	An ENROL message sent from an Enroller that did not require an ENROLLED response may
692	be modified <i>en route</i> to the Superior by an intermediate actor to ask for an ENROLLED
693	response to be sent to the intermediate. (This may occur in the "one-shot" scenario, where an
694	ENROL/no-rsp-req is received in relation to a CONTEXT_REPLY/related; the receiver of
695	the CONTEXT_REPLY will need to ensure the enrolment is successful).
	_ /

696697 Participant

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

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

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

Note – Possible approaches are:

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

Sub-coordinator

An Inferior which is also an Atomic Superior.

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

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

739 740	A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is
741 742	propagated to all Inferiors.
743 744	Sub-composer
745 746	An Inferior which is also a Cohesive Superior.
747 748 749	Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from the perspective of its Superior.
750 751 752 753	A sub-composer is similar to a sub-coordinator, except that the constraints linking the different Inferiors concern only those Inferiors in the confirm-set. How the confirm-set is controlled, and when, is not defined in this specification.
754 755 756 757	If the sub-composer is instructed to cancel, by receiving a CANCEL message from its Superior, the cancellation is propagated to all its Inferiors.
758 759	Roles involved in the Terminator:Decider relationship
760	Decider
761	
762	A Superior that is not the Inferior on a Superior:Inferior relationship. It is the top-node in the
763	transaction tree and receives requests from a Terminator as to the desired outcome for the
764	business transaction. If the Terminator asks the Decider to confirm the business transaction, it
765	is the responsibility of the Decider to finally take the confirm decision. The taking of the
766	decision is synonymous with the persisting of information identifying the Inferiors that are to
767 768	be confirmed. An Inferior cannot be confirmed unless PREPARED has been received from it.
769	A Decider is instructed to cancel by receiving CANCEL/whole.
770	A Decider is instructed to earlier by receiving extineers, whole.
771	A Decider that is an Atomic Superior (all Inferiors will have the same outcome) is a
772	Coordinator. A Decider that is a Cohesive Superior (some Inferiors may cancel, some
773	confirm) is a Cohesion.
774	
775	All Deciders receive
776	REQUEST_CONFIRM
777	CANCEL/whole
778	REQUEST_STATUSES
779	
780	All Deciders send
781	CONFIRMED
782	CANCELLED
783	INFERIOR_STATUSES
784 785	A D '1 DEOLIEGE CTATUS 1 1' '4 CTATUS 2' '4 CTATUS
785	A Decider also receives REQUEST_STATUS and replies with STATUS, reporting its state

as a whole.

787	
788	Coordinator
789	
790	A Decider that is an Atomic Superior. The same outcome decision will be applied to all
791 792	Inferiors (excluding any from which RESIGN is received).
792 793	PREPARED must be received from all remaining Inferiors for a confirm decision to be taker
794	TREE ARED must be received from an remaining interiors for a committi decision to be taken
795	A Coordinator must make a cancel decision if
796	it is instructed to cancel by the Terminator
797	if CANCELLED is received from any Inferior
798	if it is unable to persist a confirm decision
799	•
800	Composer
801	
802	A Decider that is a Cohesive Superior. If the Terminator requests confirmation of the
803	Cohesion, that request will determine the confirm-set of the Cohesion.
804	DDEDARED
805	PREPARED must be received from all Inferiors in the confirm-set (excluding any from
806 807	which RESIGN is received) for a confirm decision to be taken.
808	A Composer must make a cancel decision (applying to all Inferiors) if
809	it is instructed to cancel by the Terminator
810	if CANCELLED is received from any Inferior in the confirm-set
811	if it is unable to persist a confirm decision
812	if it is unable to persist a commit decision
813	A Composer may be asked to prepare some or all of its Inferiors by receiving PREPARE. It
814	issues PREPARE to any of those Inferiors from which none of PREPARED, CANCELLED
815	or RESIGNED have been received, and replies to the PREPARE with
816	INFERIOR_STATUSES.
817	-
818	A Composer may be asked to cancel some of its Inferiors, but not itself, by receiving
819	CANCEL/inferiors.
820	
821	In addition to the messages received by the Composer as a Decider, it receives
822	PREPARE
823	CANCEL/inferiors
824	
825	Terminator
826	
827	Asks a Decider to confirm the business transaction, or instructs it to cancel all or (for a
828	Cohesion) part of the business transaction.
829	
830	All communications between Terminator and Decider are initiated by the Terminator. A
831	Terminator is usually an application element.
832 833	A request to confirm is made by sending REQUEST_CONFIRM to the target Decider. If the
834	Decider is a Cohesion Composer, the Terminator may select which of the Composer's

835 836	Inferiors are to be included in the confirm-set. If the Decider is an Atom Coordinator, all Inferiors are included. After applying the decision, the Decider replies with CONFIRMED,
837 838	CANCELLED or (in the case of problems) INFERIOR_STATUSES.
839	A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its
840	Inferiors with PREPARE/inferiors. The Composer replies with INFERIOR_STATUSES.
841	
842	A Terminator may send CANCEL to instruct the Decider to cancel the whole business
843 844	transaction, or, if it is a Cohesion Composer, some of its Inferiors. The Decider replies with CANCELLED, or for a selective cancel or in the case of problems, INFERIOR_STATUSES
845	
846	A Terminator may check the status of the Inferiors of the Decider by sending
847 848	REQUEST_STATUSES. The Decider replies with INFERIOR_STATUSES.
849	A Terminator sends
850	REQUEST_CONFIRM
851	CANCEL
852	PREPARE/inferiors
853	REQUEST STATUSES
854	REQUEST_STATUSES
855	A Terminator receives
856	CONFIRMED
857	CANCELLED
858	INFERIOR_STATUSES
859	IN ERGON_STATESEES
860	Initiator
861	
862	Requests a Factory to create a Superior – this will either be a Decider (representing a new
863	top-level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an
864	existing business transaction.
865 866	An Initiator sends
	All illuator series
867 868	BEGIN
869	BEGIN & CONTEXT
870	BEOIN & CONTEAT
871	to a Factory, and receives in reply
872	to a ractory, and receives in reply
873	BEGUN & CONTEXT
874	BLOCIVE CONTEXT
875 876	Factory
877	Creates Superiors and returns the CONTEXT for the new Superior. The following types of
878	Superior are created:
879	
880	Decider, which is either
881	Composer or
882	Coordinator

883	Sub-composer		
884	Sub-coordinator Sub-coordinator		
885			
886	A Factory receives		
887	·		
888	BEGIN		
889	BEGIN & CONTEXT		
890			
891	and replies with		
892	•		
893	BEGUN & CONTEXT		
894			
895	If the BEGIN has no related CONTEXT, the Factory creates a Decider, either a Cohesion		
896	Composer or an Atom Coordinator, as determined by the "superior type" parameter on the		
897	BEGIN.		
898			
899	If the BEGIN has a related CONTEXT, the new Superior is also enrolled as an Inferior of the		
900	Superior identified by the CONTEXT. The new Superior is thus a sub-composer or sub-		
901	coordinator, as determined by the "superior type" parameter on the BEGIN.		
902			
903			
904			
905	Other roles		
905 906	Other roles		
907	Redirector		
908	Troum outor		
909	Sends a REDIRECT message to inform any actor that an address previously supplied for		
910	some other actor is no longer appropriate, and to supply a new address or set of addresses to		
911	replace the old one.		
912	replace the old olic.		
913	A Redirector may send a REDIRECT message in response to receiving a message using the		
914	old address, or may send REDIRECT at its own initiative.		
915	If a Superior moves from the superior-address in its CONTEXT, or an Inferior moves from		
916	the inferior-address in the ENROL message, the implementation must ensure that a		
917	Redirector catches any inbound messages using the old address and replies with a		
918	REDIRECT message giving the new address. (Note that the inbound message may itself be a		
919	REDIRECT message.)		
920	REDIRECT message.)		
921	A Redirector may also be used to change the address of other BTP actors.		
922	A Redirector may also be used to change the address of other BTT actors.		
922 923	After receiving a REDIRECT message, the BTP actor must use the new address not the old		
923 924			
924 925	one, unless failure prevents it updating its information.		
	Chalus Damusahan		
926	Status Requestor		
927			
	Democrate and manifestable assumed status of as Infinite and Deviate The sail of Co.		
928	Requests and receives the current status of an Inferior or a Decider. The role of Status		
928 929 930	Requests and receives the current status of an Inferior or a Decider. The role of Status Requestor has no responsibilities – it is just a name for where the REQUEST_STATUS comes from.		

931			
932	A Status Requestor sends		
933	1		
934	REQUEST_STATUS		
935	_		
936	and receives		
937			
938	STATUS		
939			
940	in response.		
941			
942	The information returned will always relate to the actor concerned in its role as an Inferior,		
943 944	even if it is also a Superior.		
	Abstract Messages and Associated Contracts		
946			
947	BT Protocol Messages are defined in this section in terms of the abstract information that has		
948	to be communicated. These abstract messages will be mapped to concrete messages		
949	communicated by a particular carrier protocol (there can be several such mappings defined).		
950			
951	The abstract message set and the associated state table assume the carrier protocol will		
952			
953	deliver messages completely and correctly, or not at all (corrupted messages will		
954 955	not be delivered);		
955 956	report some communication failures, but will not necessarily report all (i.e. not all		
950 957	message deliveries are positively acknowledged within the carrier);		
958	message deriveries are positively acknowledged within the carrier),		
959	sometimes deliver successive messages in a different order than they were sent;		
960	sometimes deriver successive messages in a different order than they were sent,		
961	and		
962			
963	does not have built-in mechanisms to link a request and a response		
964	1		
965	Note that these assumptions would be met by a mapping to SMTP and more than met by		
966	mappings to SOAP.		
967			
968	However, when the abstract message set is mapped to a carrier protocol that provides a richer		
969	service (e.g. reports all delivery failures, guarantees ordered delivery or offers a		
970	request/response mechanism), the mapping can take advantage of these features. Typically ir		
971	such cases, some of the parameters of an abstract message will be implicit in the carrier		
972	mechanisms, while the values of other parameters will be directly represented in transmitted		
973	elements.		
974			
975			
976 977	Addresses		

All of the messages except CONTEXT and CONTEXT_REPLY 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 invariably have been originally determined by the implementation that is the target or desired future 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 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.

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

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

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

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

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

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

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

Request/response pairs

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

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

Compounding messages

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

- a) Sending the messages together has semantic significance. One message is said to be "related to" the other.
- b) Sending of the messages has no semantic significance, but is merely a convenience or optimisation. This is termed "bundling".

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

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

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

In "one-wire", all message exchanges between two sides of a Superior:Inferior relationship, including the associated application messages, pass via the same "endpoints". These "endpoints" may in fact be relays, routing messages on to particular actors within their domain. The onward routing will require some further addressing, but this has to be opaque to

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

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

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

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

Extensibility

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

1122 1123	How the sub-parameter is associated with the new parameter is determined by the particular binding.			
1124 1125 1126	No special mechanism is provided to a	l mechanism is provided to allow for the introduction of completely new messages.		
1127 1128 1129 1130 1131 1132 1133 1134 1135	Inferior handle Some of the messages exchanged between a Terminator and a Decider are concerned with the individual Inferiors enrolled with the Decider, and not with the business transaction as a whole. These messages distinguish the Inferiors of Decider using an "inferior handle". This i created by the Decider and is unambiguous within the scope of the Decider. The "inferior handle" is distinct from the "inferior identifier" passed on an ENROL message			
1135 1136 1137 1138 1139 1140 1141 1142 1143	unambiguous within the scope of the a within any of the individual addresses identify the Inferior across all the place responsibility for it). The "inferior handle" is only used by	rior handle" is only used by the Terminator to refer to the inferiors of the Decider. ges between the Decider and its Inferiors, the address-as-inferior and inferior		
1144 1145	Messages			
1146 1147	Qualifiers			
1147 1148 1149 1150 1151	1148 1149 All messages have a Qualifiers parameter which contains zero or more Qualifier value Qualifier has sub-parameters:			
1101	Sub-parameter	Туре		
	qualifier name	string		
	qualifier group	URI		
	must-be-understood	Boolean		
	to-be-propagated	Boolean		
	content	Arbitrary – depends on type		
1152				
1153		he Qualifier name is unambiguous. Qualifiers in the		
1154	same group need not have any functional relationship. The qualifier group will			
1155 1156		fy the specification that defines the qualifier's meaning e defined in this or other standard specifications, in		
1150		lar community of users or of implementations or by		

bilateral agreement.

1158

Qualifier name this identifies the meaning and use of the Qualifier, using a name 1160 that is unambiguous within the scope of the Qualifier group. 1161 1162 1163 **Must-be-understood** if this has the value "true" and the receiving entity does not recognise the Qualifier type (or does not implement the necessary 1164 functionality), a FAULT "UnsupportedQualifier" shall be returned and the 1165 message shall not be processed. Default is "true". 1166 1167 1168 **To-be-propagated** if this has the value "true" and the receiving entity passes the 1169 BTP message (which may be a CONTEXT, but can be other messages) onwards 1170 to other entities, the same Qualifier value shall be included. If the value is "false", the Qualifier shall not be automatically included if the BTP message is 1171 1172 passed onwards. (If the receiving entity does support the qualifier type, it is 1173 possible a propagated message may contain another instance of the same type, even with the same Content – this is not considered propagation of the original 1174 qualifier.). Default is "false". 1175 1176 **Content** the type (which may be structured) and meaning of the content is 1177 defined by the specification of the Qualifier. 1178 1179 1180 1181 CONTEXT 1182 1183 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more application messages. (The means by which this relationship is represented is determined by 1184 the binding and the binding mechanisms of the application protocol.) The "superior type" 1185 1186 parameter identifies whether the Superior will apply the same decision to all Inferiors enrolled using the same superior identifier ("superior type" is "atom") or whether it may 1187 apply different decisions ("superior type" is "cohesion"). 1188 1189 Parameter Type address-as-superior Set of BTP addresses Identifier superior identifier cohesion/atom superior type qualifiers List of qualifiers 1190 1191 1192 address-as-superior the address to which ENROL and other messages from an 1193 enrolled Inferior are to be sent. This can be a set of alternative addresses. 1194

superior identifier identifies the Superior within the scope of the address-as-

superior type identifies whether the CONTEXT refers to a Cohesion or an

Atom. Default is atom.

superior

1195 1196

1197

1198

1200				
1201				
1202	qualifiers standardised or other qualifiers. The standard qualifier "Transaction			
1203	timelimit" is carried by CONTEXT.			
1204				
1205			for CONTEXT as it is only transmitted in relation to the	
1206	application	messages.		
1207				
1208			CONTEXT/atom refer to CONTEXT messages with the	
1209	superior typ	pe with the appropriate va	lue.	
1210				
1211				
1212	CONTEXT_	_REPLY		
1213				
1214			eipt of CONTEXT (related to application message(s)) to	
1215			nents have already completed (ENROLLED has been	
1216			ROL messages sent in relation to the	
1217			ent attempt has failed. CONTEXT_REPLY may be sent	
1218			pically the response to the application message related to	
1219		EXI). In some bindings th	e CONTEXT_REPLY may be implicit in the application	
1220 1221	message.			
1221		D	-	
		Parameter	Туре	
		superior-address	BTP address	
		superior identifier	Identifier	
		completion_status	complete/related/repudiated	
		qualifiers	List of qualifiers	
1222				
1223			of the addresses from the address-as-superior from the	
1224	CONTEXT. (The parameter is present in CONTEXT_REPLY to disambiguate			
1225		the superior identifier.)		
1226				
1227 1228		superior identifier the superior identifier from the CONTEXT		
1229		completion status: rep	orts whether all enrol operations made necessary by the	
1230 1231			NTEXT message have completed. Values are	
1231		value	meaning	
		completed	All enrolments (if any) have succeeded already	
		related	At least some enrolments are to be performed by ENROL messages related to the CONTEXT_REPLY. All other enrolments (if any) have succeeded already.	
		repudiated	At least one enrolment has failed. The implications of receiving the CONTEXT have not been honoured.	

1232			
1233	qualifiers standardised or other qualifiers.		
1234	THE CONTENTS DEDUCE TO A CONTENT DEDUCE TO A		
1235	The form CONTEXT_REPLY/completed, CONTEXT_REPLY/related and		
1236 1237	CONTEXT_REPLY/repudiated refer to CONTEXT_REPLY messages with status having the		
1237	appropriate value. The form CONTEXT_REPLY/ok refers to either of CONTEXT_REPLY/completed or CONTEXT_REPLY/related.		
1239	CONTEXT_REFLT/completed of	CONTEXT_REFET/Telated.	
1240	If there are no necessary enrolments (e.g. the application messages related to the received		
1241	CONTEXT did not require the enrolment of any Inferiors), then		
1242	CONTEXT_REPLY/completed is		
1243	-		
1244		d is received, the receiving implementation must ensure	
1245	that the business transaction will not be confirmed.		
1246			
1247	BEGIN		
1248 1249	DEGIN		
1249	Δ request to a Factory to create a n	ew Rusiness Transaction. This may either be a new ton-	
1251	A request to a Factory to create a new Business Transaction. This may either be a new top-level transaction, in which case the Composer or Coordinator will be the Decider, or the new		
1252	Business Transaction may be immediately made the Inferior within an existing Business		
1253	Transaction (thus creating a sub-Co	· · · · · · · · · · · · · · · · · · ·	
1254	-		
	Parameter	Туре	
	target address	BTP address	
	reply address	BTP address	
	transaction type	cohesion/atom	
	qualifiers	List of qualifiers	
1255			
1256		dress of the entity to which the BEGIN is sent. How this	
1257	•	I the nature of the entity are outside the scope of this	
1258	specification.		
1259			
1260	reply address the address to which the replying BEGUN and related		
1261	CONTEXT message sl	hould be sent.	
1262 1263	transaction type iden	tifies whether a new Cohesian or new Atom is to be	
1264	transaction type identifies whether a new Cohesion or new Atom is to be		
1265	created; this value will be the "superior type" in the new CONTEXT		
1266	qualifiers standardised	d or other qualifiers. The standard qualifier "Transaction	
1267	qualifiers standardised or other qualifiers. The standard qualifier "Transaction timelimit" may be present on BEGIN, to set the timelimit for the new business		
1268	transaction and will be copied to the new CONTEXT. The standard qualifier		
1269	"Inferior name" may b	e present if there is a CONTEXT related to the BEGIN.	

1271 1272 1273 1274 1275 1276	A new top-level Business Transaction is created if there is no CONTEXT related to the BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is created if the CONTEXT message for the existing Business Transaction is related to the BEGIN. In this case, the Factory is responsible for enrolling the new Composer or Coordinator as an Inferior of the Superior identified in that CONTEXT.		
1277 1278 1279	Note – This specification does not provide a standardised means to determine which of the Inferiors of a sub-Composer are in its confirm set. This is considered part of the application:inferior relationship.		
1280 1281 1282 1283	the corresponding value.	N/atom refer to BEGIN with "transaction type" having	
1284 1285	Types of FAULT possible (sent to Rep	oly address)	
1286 1287	General		
1288 1289	BEGUN		
1290 1291 1292	BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT for the new business transaction.		
	Parameter	Туре	
	target address	BTP address	
	address-as-decider	Set of BTP addresses	
	transaction-identifier	Identifier	
	inferior-handle	Handle	
	address-as-inferior	Set of BTP addresses	
	qualifiers	List of qualifiers	
1293 1294 1295 1296	target address the address address from the BEGIN.	ss to which the BEGUN is sent. This will be the reply	
1297 1298 1299 1300 1301	address-as-decider for a top-level transaction (no CONTEXT related to the BEGIN), this is the address to which PREPARE, REQUEST_CONFIRM, CANCEL and REQUEST_STATUS messages are to be sent; if a CONTEXT was related to the BEGIN this parameter is absent		
1302 1303 1304 1305 1306	transaction-identifier identifies the new Composer or Coordinator within the scope of the address-as-decider. If this is not a top-level transaction, the transaction-identifier is optional, but if present shall be the inferior-identifier used in the enrolment with the Superior identified by the CONTEXT related to the BEGIN.		

inferior handle Shall be absent if this is a top-level transaction and rance to the Superior identified in the CONTEXT related to the BEGIN). It inferior handle will identify this new business transaction as in the in parameters in messages between the Superior identified in the CONTEXT related to the BEGIN (acting as a Decider) and its Terminator. The value shadifferent for each enrolled Inferior of that Superior. address-as-inferior This parameter shall be absent if this is a top-le transaction and may be present, at implementation option otherwise. shall be the address-as-inferior used in the enrolment with the Superior by the CONTEXT related to the BEGIN. If this is a top-level transaction and may be present, at implementation option otherwise.		Presence or absence will be determined by the nature in the CONTEXT related to the BEGIN). If present, the sy this new business transaction as in the inferiors-list tween the Superior identified in the CONTEXT related Decider) and its Terminator. The value shall be Inferior of that Superior. parameter shall be absent if this is a top-level esent, at implementation option otherwise. If present, it erior used in the enrolment with the Superior identified to the BEGIN. If this is a top-level transaction		
1323 1324 1325 1326 1327 1328	"address-as-superior" in the related CC is no general requirement that they eve the target address of the BEGIN messa are applied to the appropriate Compose	nentation option, the "address-as-decider" and/or "address-as-inferior" and the as-superior" in the related CONTEXT may be the same or may be different. There ral requirement that they even use the same bindings. Any may also be the same as address of the BEGIN message (the inferior identifier on messages will ensure they it to the appropriate Composer or Coordinator).		
1329 1330	No FAULT messages are issued on rec	eiving BEGUN.		
1331 1332 1333 1334 1335 1336	ENROL A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a CONTEXT message in relation to an application request. The actor issuing ENROL plays the role of Enroller.			
	Parameter	type		
	target address	BTP address		
	superior identifier	Identifier		
	reply requested	Boolean		
	reply address	BTP address		
	address-as-inferior	Set of BTP addresses		
	inferior identifier	Identifier		
	Qualifiers	List of qualifiers		
1337				
1338 1339 1340	target address the address address-as-superior from t	s to which the ENROL is sent. This will be the he CONTEXT message.		
1340	superior identifier. The su	perior identifier as on the CONTEXT message		

10.40			
1342			
1343	reply requested true if an ENROLLED response is required, false otherwise.		
1344	Default is false.		
1345			
1346	reply address the addre	ss to which a replying ENROLLED is to be sent, if	
1347	"reply requested" is true. If this field is absent and "reply requested" is true, the		
1348	ENROLLED should be sent to the "address-as-inferior" (or one of them, at		
1349	sender's option)		
1350	-		
1351	address-as-inferior the	address to which PREPARE, CONFIRM, CANCEL and	
1352		ssages for this Inferior are to be sent.	
1353	_		
1354	inferior identifier an ide	ntifier that unambiguously identifies this Inferior within	
1355		ddress-as-inferior set of BTP-addresses.	
1356	the scope of any of the ac	duress-as-interior set of BTF-addresses.	
	qualifiere	and an alicina The standard and Confidence	
1357		or other qualifiers. The standard qualifier "Inferior	
1358	name" may be present.		
1359	T CEALLE 11 ()	1 11)	
1360	Types of FAULT possible (sent to Re	eply address)	
1361			
1362	General		
1363	<i>InvalidSuperior</i> – if superior identifier is unknown		
1364	DuplicateInferior – if inferior with at least one of the set address-as-		
1365	inferior the same	and the same inferior identifier is already enrolled	
1366	<i>WrongState</i> – if	it is too late to enrol new Inferiors (generally if the	
1367		ady sent a PREPARED message to its superior or	
1368		it has already issued CONFIRM to other Inferiors).	
1369		•	
1370	The form ENROL/rsp-req refers to a	n ENROL message with "reply requested" having the	
1371		rs to an ENROL message with "reply requested" having	
1372	the value "false"		
1373			
1374	ENROL/no-rsp-reg is typically sent i	n relation to CONTEXT_REPLY/related. ENROL/rsp-	
1375	req is typically when CONTEXT_REPLY/completed will be used (after the ENROLLED		
1376	message has been received.)	1	
1377	,		
1378	ENROLLED		
1379	LINICOLLED		
1380	Sent from Superior in reply to an FN	ROL/rsp-req message, to indicate the Inferior has been	
1381	* * * * * * * * * * * * * * * * * * *	ore be included in the termination exchanges)	
1382	saccessiany emoned (and will therefore	or meraded in the termination exchanges)	
1302	ъ.	-	
	Parameter	Туре	
	target address	BTP address	
	inferior identifier	Identifier	
	inferior-handle	Handle	
	แแบบบาเลเนเธ	Tanac	

	Qualifiers	List of qualifiers		
1383				
1384 1385 1386	reply address from	target address the address to which the ENROLLED is sent. This will be the reply address from the ENROL message (or one of the address-as-inferiors if the reply address was empty)		
1387 1388	inferior identifier	inferior identifier The inferior identifier as on the ENROL message		
1389 1390 1391 1392 1393	in the inferiors-list Decider) and its T	inferior handle the inferior handle that will identify this newly enrolled Inferior in the inferiors-list parameters in messages between the Superior (acting as a Decider) and its Terminator. This parameter is optional. The value shall be different for each enrolled Inferior of the Superior.		
1394 1395	qualifiers standa	rdised or other qualifiers.		
1396 1397 1398 1399	No FAULT messages are issu	•		
1400	RESIGN			
1401 1402 1403 1404 1405		n an enrolled Inferior to the Superior to remove the Inferior from the enrolment. This be sent if the operations of the business transaction have had no effect as perceived ferior.		
1406 1407 1408	RESIGN may be sent in response to a PREPARE message (instead of a PREPARED), or at any point prior to the sending of a PREPARED or CANCELLED message.			
	Parameter	type		
	target address	BTP address		
	superior identifier	identifier		
	address-as-inferior	Set of BTP addresses		
	inferior identifier	identifier		
	response requested	Boolean		
	Qualifiers	List of qualifiers		
1409 1410 1411		ne address to which the RESIGN is sent. This will be the is used on the ENROL message.		
1412 1413 1414	superior-identifie	The superior identifier as on the ENROL message		
1415 1416		or The address-as-inferior as on the earlier ENROL message identifier, this determines who the message is from)		
1417 1418	inferior-identifier	The inferior identifier as on the earlier ENROL message		

1.110		
1419	rocponco roquoct	od : " "'C DEGICNED : 1
1420 1421	response-requested is set to "true" if a RESIGNED response is required.	
1421	qualifiers standardised or other qualifiers.	
1423	qualifiers standardised of other qualifiers.	
1424	Note RESIGN is equivalent to readonly vote in some other protocols, but can be issued	
1425	early.	t to readomy vote in some other protocols, but can be issued
1426	curiy.	
1427	Types of FAULT possible (sent to address-as-inferior)	
1428		
1429	General	
1430	<i>InvalidSuperior</i> – if superior identifier is unknown	
1431	InvalidInf	erior – if no ENROL had been received for this address-as-
1432		nd identifier (Inferior Identity)
1433	WrongState – if a PREPARED or CANCELLED has already been	
1434	received by the Superior from this Inferior	
1435 1436		
1430	value "true"; RESIGN /no-rsp-req refers to an RESIGN message with "reply requested"	
1438	having the value "false"	
1439	maying the value table	
1440		
1441	RESIGNED	
1442		
1443 1444	Sent in reply to a RESIGN/rsp-req message.	
1111	Parameter	Туре
	target address	BTP address
	ŭ	
	inferior identifier	Identifier
	qualifiers	List of qualifiers
1445		
1446	target address the address to which the RESIGNED is sent. This will be the	
1447	address-as-inferior from the ENROL message.	
1448	inforiar identifier	The 'afe' a '1 a 'f' a and the afe ENDOL
1449 1450	inferior identifier The inferior identifier as on the earlier ENROL message for this Inferior.	
1450	this interior.	
1452	qualifiers standardised or other qualifiers.	
1453	The state of surface of surface description	
1454	After receiving this message the Inferior will not receive any more messages with this	
1455	address-as-inferior and identifier.	
1456		
1457	No FAULT messages are issued on receiving RESIGNED.	
1458		
1459	PREPARE	
1460		

1462 RES	Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after receiving a PREPARED message.		
1465 Sen 1466 sen 1467 CA 1468 all t	Sent from a Terminator to a Composer to tell it to prepare all or some of its inferiors, by sending PREPARE to any that have not already sent PREPARED, RESIGN or CANCELLED to the Composer. If the inferiors-list parameter is absent, the request applies to all the inferiors; if the parameter is present, it applies only to the identified inferiors of the Composer.		
1470		_	
	Parameter	Type	
	target address	BTP address	
	inferior identifier	Identifier	
	reply address	BTP address	
	transaction-identifier	Identifier	
	inferiors-list	List of inferior handles	
1471	qualifiers	List of qualifiers	
1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1488 1489 1490 1491 1492 1493	from Superior to Inferior, message,. When sent from address from the BEGUN inferior identifier When on the earlier ENROL meterminator to Composer. reply address When sent Terminator sending the Pasent from Superior to Inferiors and will be the parameter shall be absent inferiors-list When sent for Inferiors of this Composer when sent to a Composer shall be absent when sent to a Composer shall be a comp	sent from Superior to Inferior, the inferior identifier as essage. This parameter shall be absent when sent from at from Terminator to Composer, the address of the REPARE message. This parameter shall be absent when	

1498 1499	On receiving PREPARE, an Inferior should RESIGN.	I reply with a PREPARED, CANCELLED or		
1500 1501 1502 1503 1504 1505 1506 1507	Inferiors if the parameter is absent), from w RESIGNED has been received, the Compos Terminator, using the reply address on the I INFERIOR_STATUSES message giving the	ent to a Composer, for all Inferiors identified in the inferiors-list parameter (all s if the parameter is absent), from which none of PREPARED, CANCELLED or NED has been received, the Composer shall issue PREPARE. It will reply to the ator, using the reply address on the PREPARE message, sending an IOR_STATUSES message giving the status of the Inferiors identified on the inferiors-uneter (all of them if the parameter was absent).		
1508 1509	Types of FAULT possible (sent to Superior	address)		
1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521 1522	InvalidInferior — if inferiors on the inferiors-list is unknown WrongState — if a CON this Inferior; if a REQU been received by this CON The form PREPARE/whole refers to a PRE "inferiors-list" parameter is absent. The form	NFIRM or CANCEL has already been received by JEST_CONFIRM or CANCEL/whole has already Composer. PARE message sent to a Composer where the m PREPARE/inferiors refers to a PREPARE eriors-list" parameter is present. The unqualified		
1523	PREPARED			
1524 1525 1526 1527 1528 1529 1530 1531	the Inferior has determined the operations a can be cancelled, as may be instructed by th (i.e. it is the Inferiors choice, as constrained	cited or in response to PREPARE, but only when ssociated with the Inferior can be confirmed and ne Superior. The level of isolation is a local matter by the shared understanding of the application may see applied results of operations or may see		
	Parameter	Туре		
	target address	BTP address		
	superior identifier	ldentifier		
	address-as-inferior	Set of BTP addresses		

1534

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

Identifier

Boolean

List of qualifiers

inferior identifier

default is cancel

qualifiers

1535		
1536	superior identifier w	nen the message is sent from an Inferior to the Superior,
1537		s on the ENROL message
1538	-	-
1539	address-as-inferior w	Then the message is sent from an Inferior to the Superior,
1540		as on the earlier ENROL message (with the inferior
1541	identifier, this determine	es who the message is from)
1542		
1543	inferior identifier The	inferior identifier as on the ENROL message
1544		
1545	default is cancel if "t	rue", the Inferior states that if the outcome at the Superior
1546		ons associated with this Inferior, no further messages need
1547		If the Inferior does not receive a CONFIRM message, it
1548		ed operations. The value "true" will invariably be used
1549		ng under what circumstances (usually a timeout) an
1550		cancel will be made. If "false", the Inferior will expect
1551		EL message as appropriate, even if qualifiers indicate that
1552	an autonomous decisio	
1553		
1554	qualifiers standardised	or other qualifiers. The standard qualifier "Inferior
1555	timeout" may be carrie	
1556	·	
1557	On sending a PREPARED, the Infe	rior undertakes to maintain its ability to confirm or cancel
1558		ons until it receives a CONFIRM or CANCEL message.
1559	Qualifiers may define a time limit of	or other constraints on this promise. The "default is
1560	cancel" parameter affects only the	subsequent message exchanges and does not of itself state
1561	that cancellation will occur.	
1562		
1563	Types of FAULT possible (sent to	address-as-inferior)
1564		
1565	General	
1566	InvalidSuperio	or − if Superior identifier is unknown
1567	InvalidInferior	- if no ENROL has been received for this address-as-
1568	inferior and ide	entifier, or if RESIGN has been received from this Inferior
1569		
1570	The form PREPARED/cancel refer	s to a PREPARED message with "default is cancel" =
1571		ARED refers to a PREPARED message with "default is
1572	cancel" = "false".	
1573		
1574		
1575	CONFIRM	
1576		
1577	Sent by the Superior to an Inferior	rom whom PREPARED has been received.
1578		
	Parameter	Туре
	target address	BTP address

	inferior identifier	Identifier	
	qualifiers	List of qualifiers	
1579	·	·	
1580		ess to which the CONFIRM message is sent. This will	
1581	be the address-as-inferior	r from the ENROL message.	
1582	inferior identifier The in	forion identifican on on the continu ENDOI masses of on	
1583 1584	this Inferior.	nferior identifier as on the earlier ENROL message for	
1585	uns micror.		
1586	qualifiers standardised of	or other qualifiers.	
1587			
1588		is released from its promise to be able to undo the	
1589 1590	to everyone (if they weren't already)	rior. The effects of the operations can be made available	
1591	to everyone (if they weren't uneady)	•	
1592	Types of FAULT possible (sent to Su	perior address)	
1593			
1594	General		
1595		if inferior identifier is unknown	
1596 1597	<i>WrongState</i> – if no PREPARED has been sent by, or if CANCEL has been received by this Inferior.		
1598	been received by	this interior.	
1599			
1600	CONFIRMED		
1601 1602	Sont after the Inferior has applied the	confirmation, both in reply to CONFIRM or when the	
1603	Inferior has made an autonomous cor		
1604		erior decides to confirm its associated operations.	
1605			
1606 1607		r to a Terminator in reply to REQUEST_CONFIRM if	
1608	reporting hazards.	for a Cohesion, all other Inferiors cancel) without	
1609	reporting nazards.		
	Parameter	Туре	
	target address	BTP address	
	superior identifier	Identifier	
	address-as-inferior	Set of BTP addresses	
	inferior identifier	Identifier	

BTP address

List of qualifiers

identifier

Boolean

1610

address-as-decider transaction-identifier

confirm received

qualifiers

1611 1612	target address the address to which the CONFIRMED is sent. When sent by an Inferior to a Superior, this will be the Superior address as on the CONTEXT
1613	message. When sent from a Decider to a Terminator it will be the reply address
1614	from the REQUEST_CONFIRM message.
1615	ouncelor identifies TTT 1
1616	superior identifier When the message is sent from an Inferior to the Superior,
1617	this shall be the superior identifier as on the CONTEXT message. This parameter
1618	shall be absent when CONFIRMED is sent from Decider to Terminator.
1619	
1620	address-as-inferior When the message is sent from an Inferior to the Superior,
1621	this shall be the address-as-inferior as on the earlier ENROL message (with the
1622	inferior identifier, this determines who the message is from). This parameter shall
1623	be absent when CONFIRMED is sent from Decider to Terminator.
1624	
1625	inferior identifier When the message is sent from an Inferior to the Superior, this
1626	shall be the inferior identifier as on the earlier ENROL message. This parameter
1627	shall be absent when CONFIRMED is sent from Decider to Terminator.
1628	
1629	address-as-decider When the message is sent from a Decider to the
1630	Terminator, this shall be the address-as-decider of the Decider as on the BEGUN
1631	message (with the transaction identifier, this determines who the message is
1632	from). This parameter shall be absent when CONFIRMED is sent from an
1633	Inferior to Superior.
1634	illicitor to Superior.
	transaction identifier When the masses is sent from a Decidents the
1635	transaction identifier When the message is sent from a Decider to the
1636	Terminator, this shall be the transaction identifier as on the BEGUN message (i.e.
1637	the identifier of the Decider as a whole). This parameter shall be absent when
1638	CONFIRMED is sent from an Inferior to Superior
1639	C
1640	confirm received "true" if CONFIRMED is sent after receiving a CONFIRM
1641	message; "false" if an autonomous confirm decision has been made and either if
1642	no CONFIRM message has been received or the implementation cannot
1643	determine if CONFIRM has been received (due to loss of state information in a
1644	failure). This parameter shall be absent when CONFIRMED is sent from Decider
1645	to Terminator.
1646	
1647	qualifiers standardised or other qualifiers.
1648	
1649	Types of FAULT possible (sent to address-as-inferior)
1650	
1651	General
1652	<i>InvalidSuperior</i> – if Superior identifier is unknown
1653	<i>InvalidInferior</i> – if no ENROL has been received for this address-as-
1654	inferior and identifier, or if RESIGN has been received from this Inferior.
1655	interior and identifier, of it NESTON has been received from this interior.
1033	
1656	Note – A CONFIRMED message arriving before a CONFIRM message is
1657	
1057	sent, or after a CANCEL has been sent will occur when the Inferior has

1658 1659	taken an autonomous decision and is not regarded as occurring in the wrong state. (The latter will cause a CONTRADICTION message to be sent.)			
1660 1661 1662 1663 1664 1665 1666 1667		received" = "false"; CONFIRME with "confirm received" = "true"	ers to a CONFIRMED message with "confirm Depression of the Confirm Con	
1668 1669	CANCEL			
1670 1671	Sent by the	Superior to an Inferior at any time	before (and unless) CONFIRM has been sent.	
1672 1673	Sent by a T	Terminator to a Decider at any time	before REQUEST_CONFIRM has been sent.	
		Parameter	Туре	
		target address	BTP address	
		inferior identifier	Identifier	
		reply address	BTP address	
		transaction identifier	Identifier	
		inferiors-list	List of inferior handles	
		qualifiers	List of qualifiers	
1674				
1675 1676 1677 1678 1679		target address the address to which the CANCEL message is sent. When sent from Superior to Inferior, this will be the address-as-inferior from the ENROL message,. When sent from Terminator to Composer, this will be the decideraddress from the BEGUN message.		
1680 1681 1682 1683		inferior identifier When sent from Superior to Inferior, the inferior identifier as on the earlier ENROL message. This parameter shall be absent when sent from Terminator to Decider.		
1684 1685 1686		reply address When sent from Terminator to Decider, the address of the Terminator sending the CANCEL message. This parameter shall be absent when sent from Superior to Inferior.		
1687 1688 1689 1690 1691			from Terminator to Decider, identifies the on-identifier from the BEGUN message This ent from Superior to Inferior.	
1692 1693	inferiors-list When sent from Terminator to Composer, defines which of the Inferiors of this Composer are to be cancelled. This parameter shall be absent			

1694	when sent from a Superior to an interior and when sent from a Terminator to a
1695 1696	Coordinator.
	qualifiers and a limit and a sufficient and the sure of the sure o
1697	qualifiers standardised or other qualifiers.
1698	When court to an Informing the affects of any anapptions associated with the Informing should be
1699	When sent to an Inferior, the effects of any operations associated with the Inferior should be
1700	undone. If the Inferior had sent PREPARED, the Inferior is released from its promise to be
1701	able to confirm the operations.
1702 1703	When cant to a Decider with the inferiors list peremeter is absent, the business transaction is
	When sent to a Decider with the inferiors-list parameter is absent, the business transaction is
1704 1705	cancelled – this is propagated to any remaining Inferiors by issuing CANCEL to them. No
1705	more Inferiors will be permitted to enrol.
1700	When sont to a Composer with the inferiors list represent and the Inferiors
1707	When sent to a Composer, with the inferiors-list parameter present, only the Inferiors identified in the inferiors-list are to be cancelled. Any other inferiors are unaffected by a
1708	CANCEL/inferiors. Further Inferiors may be enrolled.
1710	CAINCEL/Interiors. Further interiors may be enfolied.
1710	
1711	Note – A CANCEL/inferiors issued to a Cohesion Composer identifying all
1712	of its currently enrolled Inferiors will leave the Cohesion 'empty', but
1713	permitted to continue with new Inferiors, if any enrol.
1715	permitted to continue with new interiors, it day emot.
1714	
1715	Types of FAULT possible (sent to Superior address)
1716	
1717	General
1718	<i>UnknownTransaction</i> – if the transaction-identifier is unknown
1719	<i>InvalidInferior</i> – if inferior identifier is unknown, or an inferior-handle
1720	on the inferiors-list is unknown
1721	WrongState – if a CONFIRM has been received by this Inferior; if a
1722	REQUEST_CONFIRM has been received by this Composer.
1723	The Queen I was the received by this composer.
1724	The form CANCEL/whole refers to a CANCEL message sent to a Decider where the
1725	"inferiors-list" parameter is absent. The form CANCEL/inferiors refers to a CANCEL
1726	message sent to a Composer where the "inferiors-list" parameter is present. The unqualified
1727	form CANCEL is used to refer to a CANCEL message sent from a Superior to an Inferior.
1728	
1729	
1730	CANCELLED
1731	
1732	Sent when the Inferior has applied (or is applying) cancellation of the operations associated
1733	with the Inferior. CANCELLED is sent from Inferior to Superior in the following cases:
1734	
1735	1. before (and instead of) sending PREPARED, to indicate the Inferior is unable to
1736	apply the operations in full and is cancelling all of them;
1737	
1738	2. in reply to CANCEL, regardless of whether PREPARED has been sent;
1739	

1740 1741	3.	after sending PREPARED and then making and applying an autonomous decision to cancel.		
1742 1743 1744	4.	in reply to CONFIRM_ONE_PHASE if the Inferior decides to cancel the associated operations		
1745		······································		
1746	As is specif	fied in the state tables, cases 1, 2 and 3 are not always distinct in some		
1747	circumstan	ices of recovery and resending of messages.		
1748				
1749		<u> </u>	rminator in reply to REQUEST_CONFIRM if	
1750	all Interior	s cancel without reporting hazards.		
1751		Parameter		
		target address	BTP address	
		superior identifier	Identifier	
		address-as-inferior	Set of BTP address	
		inferior identifier	Identifier	
			BTP address	
		address-as-decider		
		transaction-identifier	identifier	
17.50		qualifiers	List of qualifiers	
1752		Assessed a delicate and a second		
1753 1754			the CANCELLED is sent. When sent by an	
1754		Inferior to a Superior, this will be the Superior address as on the CONTEXT		
1756		message. When sent from a Decider to a Terminator it will be the reply address from the REQUEST_CONFIRM message.		
1757			mossage.	
1758		superior identifier When the message is sent from an Inferior to the Superior,		
1759		this shall be the superior identifier as on the CONTEXT message. This parameter		
1760		shall be absent when CANCELLI	ED is sent from Decider to Terminator.	
1761				
1762		address-as-inferior When the me	essage is sent from an Inferior to the Superior,	
1763			or as on the earlier ENROL message (with the	
1764			who the message is from). This parameter shall	
1765		be absent when CANCELLED is	sent from Decider to Terminator.	
1766		inforior identifier William the man	The Control of the Co	
1767			sage is sent from an Inferior to the Superior, this	
1768 1769		shall be the inferior identifier as on the earlier ENROL message. This parameter shall be absent when CANCELLED is sent from Decider to Terminator.		
1709		shan be absent when CANCELLI	ED is sent from Decider to Terminator.	
1770		address-as-decider When the message is sent from a Decider to the		
1772				
1773		Terminator, this shall be the address-as-decider of the Decider as on the BEGUN message (with the transaction identifier, this determines who the message is		
1774		from). This parameter shall be absent when CANCELLED is sent from an		
1775		Inferior to Superior.		

1776 1777 1778 1779 1780 1781 1782	transaction identifier When the message is sent from a Decider to the Terminator, this shall be the transaction identifier as on the BEGUN message (i.e. the identifier of the Decider as a whole). This parameter shall be absent when CANCELLED is sent from an Inferior to Superior qualifiers standardised or other qualifiers.		
1783 1784	Types of FAULT possible (sent to add	tress-as-inferior)	
1785		2000 40 411011011	
1786	General		
1787 1788	INVAIIQSUPERIOR— InvalidInferior ;	if Superior identifier is unknown f no ENROL has been received for this address-as-	
1789		fier, or if RESIGN has been received from this Inferior	
1790		CONFIRM has been sent	
1791	-		
1792 1793 1794 1795	sent, or after a CONFIRM has taken an autonomous decision	sage arriving before a CANCEL message is s been sent will occur when the Inferior has a and is not regarded as occurring in the wrong CONTRADICTION message to be sent.)	
1796			
1797	CONFIDM ONE DUACE		
1798 1799	CONFIRM_ONE_PHASE		
1800 1801 1802 1803	this case the two-phase exchange is no	ferior, when there is only one such enrolled Inferior. In our performed between the Superior and Inferior and the associated with the Inferior is determined by the Inferior.	
	Parameter	Туре	
	target address	BTP address	
	inferior identifier	Identifier	
	report-hazard	boolean	
	qualifiers	List of qualifiers	
1804 1805 1806 1807		ss to which the CONFIRM_ONE_PHASE message is ess-as-inferior on the ENROL message.	
1808 1809 1810	inferior identifier The inf this Inferior.	Ferior identifier as on the earlier ENROL message for	
1811 1812 1813	condition occurs for the op-	nether the superior wishes to be informed if a mixed perations associated with the Inferior. If "report hazard" reply with HAZARD if a mixed condition occurs, or if	

1814 1815 1816	hazard" is false, the Inferio	the Inferior cannot determine that a mixed condition has not occurred. If "report hazard" is false, the Inferior will report only its own decision, regardless of whether that decision was correctly and consistently applied. Default is false.		
1817	ue.			
1818	qualifiers standardised or	other qualifiers.		
1819	CONFIDM ONE DIAGE 1 :			
1820 1821		ed by a Superior to an Inferior from whom		
1822	Inferior).	et to the requirement that there is only one enrolled		
1823	micrior).			
1824	Types of FAULT possible (sent to Sup	perior address)		
1825	1 \	,		
1826	General			
1827	<i>InvalidInferior</i> — is	f inferior identifier is unknown		
1828		PREPARE has already been received from this		
1829	Inferior			
1830	114.74.00			
1831	HAZARD			
1832 1833	Sent when the Inferior has either discovered	vered a "mixed" condition: that is unable to correctly		
1834		operations in accord with the decision (either the		
1835		own autonomous decision), or when the Inferior is		
1836	unable to determine that a "mixed" cor			
1837				
1838		ONFIRM_ONE_PHASE if the Inferior determines there		
1838 1839	is a mixed condition within its associat	ONFIRM_ONE_PHASE if the Inferior determines there ed operations or is unable to determine that there is not		
1838 1839 1840				
1838 1839	is a mixed condition within its associat			
1838 1839 1840	is a mixed condition within its associat a mixed condition.	ed operations or is unable to determine that there is not		
1838 1839 1840	is a mixed condition within its associat a mixed condition. Parameter target address	Type BTP address		
1838 1839 1840	is a mixed condition within its associat a mixed condition. Parameter target address superior identifier	Type BTP address Identifier		
1838 1839 1840	is a mixed condition within its associat a mixed condition. Parameter target address superior identifier address-as-inferior	Type BTP address Identifier Set of BTP addresses		
1838 1839 1840	is a mixed condition within its associat a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier	Type BTP address Identifier Set of BTP addresses Identifier		
1838 1839 1840 1841	is a mixed condition within its associat a mixed condition. Parameter target address superior identifier address-as-inferior	Type BTP address Identifier Set of BTP addresses		
1838 1839 1840 1841	is a mixed condition within its associat a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier Qualifiers	Type BTP address Identifier Set of BTP addresses Identifier List of qualifiers		
1838 1839 1840 1841	is a mixed condition within its associat a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier Qualifiers target address the address	Type BTP address Identifier Set of BTP addresses Identifier List of qualifiers s to which the MIXED is sent. This will be the		
1838 1839 1840 1841 1841	is a mixed condition within its associat a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier Qualifiers	Type BTP address Identifier Set of BTP addresses Identifier List of qualifiers s to which the MIXED is sent. This will be the		
1838 1839 1840 1841 1841 1842 1843 1844 1845	is a mixed condition within its associate a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier Qualifiers target address the address superior address from the leading and the superior address from the superior address f	Type BTP address Identifier Set of BTP addresses Identifier List of qualifiers st to which the MIXED is sent. This will be the ENROL message.		
1838 1839 1840 1841 1841 1842 1843 1844 1845 1846	is a mixed condition within its associate a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier Qualifiers target address the address superior address from the leading and the superior address from the superior address f	Type BTP address Identifier Set of BTP addresses Identifier List of qualifiers s to which the MIXED is sent. This will be the		
1838 1839 1840 1841 1841 1842 1843 1844 1845 1846 1847	is a mixed condition within its associate a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier Qualifiers target address the address superior address from the I superior identifier.	Type BTP address Identifier Set of BTP addresses Identifier List of qualifiers st to which the MIXED is sent. This will be the ENROL message.		
1838 1839 1840 1841 1841 1842 1843 1844 1845 1846	is a mixed condition within its associate a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier Qualifiers target address the address superior address from the language superior identifier. The superior identifier The	Type BTP address Identifier Set of BTP addresses Identifier List of qualifiers st to which the MIXED is sent. This will be the ENROL message.		
1838 1839 1840 1841 1841 1842 1843 1844 1845 1846 1847 1848	is a mixed condition within its associate a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier Qualifiers target address the address superior address from the I superior identifier The superior identifier The address-as-inferior The address-as-inferior identifier	Type BTP address Identifier Set of BTP addresses Identifier List of qualifiers st to which the MIXED is sent. This will be the ENROL message. Apperior identifier as used on the ENROL message address-as-inferior as on the earlier ENROL message address-as-inferior as on the message is from)		
1838 1839 1840 1841 1841 1842 1843 1844 1845 1846 1847 1848 1849	is a mixed condition within its associate a mixed condition. Parameter target address superior identifier address-as-inferior inferior identifier Qualifiers target address the address superior address from the I superior identifier The superior identifier The address-as-inferior The address-as-inferior identifier	Type BTP address Identifier Set of BTP addresses Identifier List of qualifiers st to which the MIXED is sent. This will be the ENROL message. Inperior identifier as used on the ENROL message		

1853	qualifiers standardised or other qualifiers.			
1854 1855	Types of FAULT possible (sent	pes of FAULT possible (sent to address-as-inferior)		
1856 1857 1858 1859 1860 1861 1862	InvalidInfe	General InvalidSuperior – if Superior identifier is unknown InvalidInferior – if no ENROL has been received for this address-as-inferior and identifier, or if RESIGN has been received from this Inferior		
1863 1864 1865		HAZARD/mixed refers to a HAZARD message with "level" = "mixed", the form D/possible refers to a HAZARD message with "level" = "possible".		
1866 1867	CONTRADICTION			
1868 1869 1870 1871 1872	decision for the atom. This is de	Sent by the Superior to an Inferior that has taken an autonomous decision contrary to the decision for the atom. This is detected by the Superior when the 'wrong' one of CONFIRMED or CANCELLED is received. CONTRADICTION is also sent in response to a HAZARD message.		
	Parameter	Туре		
	target address	BTP address		
	inferior identifier	Identifier		
	Qualifiers	List of qualifiers		
1873	Annual adduses it	II CONTRADICTION		
1874 1875 1876		address to which the CONTRADICTION message is sent. lress-as-inferior from the ENROL message.		
1877 1878 1879	inferior identifier of this Inferior.	The inferior identifier as on the earlier ENROL message for		
1880	qualifiers standard	ised or other qualifiers.		
1881 1882 1883	Types of FAULT possible (sent	to Superior address)		
1884 1885 1886 1887		<i>rior</i> – if inferior identifier is unknown <i>e</i> – if neither CONFIRMED or CANCELLED has been sent rior		
1888 1889	SUPERIOR_STATE			
1890 1891	Sent by a Superior as a query to	an Inferior when		
1892 1893 1894	1. in the active state			

1895 1896 1897	•			
1898 1899 1900	Also sent by the Superior particular states.	by the Superior to the Inferior in response to a received INFERIOR_STATE, in states.		
	Parameter		Туре	
	target addres	SS	BTP address	
	inferior identi	fier	Identifier	
	Status		see below	
	reply request	ted	Boolean	
	Qualifiers		List of qualifiers	
1901 1902 1903 1904 1905	This will be inferior ide	target address the address to which the SUPERIOR_STATE message is sent. This will be the address-as-inferior from the ENROL message. inferior identifier The inferior identifier as on the earlier ENROL message for		
1906 1907	this Inferior			
1907 1908 1909 1910		status states the current state of the Superior, in terms of its relation to this Inferior only.		
	status value	e meanir	ng	
	active	perspe PREPA	ationship with the Inferior is in the active state from the ctive of the Superior; ENROLLED has been sent, ARE has not been sent and PREPARED has not been d (as far as the Superior knows)	
	prepared-red		ARED has been received from the Inferior, but no outcome vailable	
	inaccessible	this Infe	te information for the Superior, or for its relationship with erior, if it exists, cannot be accessed at the moment. This be a transient condition	
	unknown	of the S	erior is not known – it does not exist from the perspective Superior. The Inferior can treat this as an instruction to any associated operations	
1911 1912 1913 1914 1915 1916 1917 1918	initiative; fa INFERIOR prepared-re	llse, if SUPERIOR_S _STATE or other mo	RIOR_STATE is sent as a query at the Superior's STATE is sent in reply to a received essage. Can only be true if status is active or qualifiers.	

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

A status of unknown shall only be sent if it has been determined for certain that the Superior has no knowledge of the Inferior, or (equivalently) it can be determined that the relationship with the Inferior was cancelled. If there could be persistent information corresponding to the

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

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

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

INFERIOR_STATE

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

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

Parameter	Туре
target address	BTP address
superior identifier	Identifier
address-as-inferior	BTP address
inferior identifier	Identifier
Status	see below
reply requested	Boolean
Qualifiers	List of qualifiers

1949
 1950 target address the address to which the INFERIOR_STATE is sent. This will
 1951 be the target address as used the original ENROL message.
 1952
 1953 superior identifier The superior identifier as used on the ENROL message

1955 1956 1957		address-as-inferior The address-as-inferior as on the ENROL message (with the inferior identifier, this determines who the message is from)			
1958 1959	inferior identifier The inferior identifier as on the ENROL message				
1960 1961 1962 1963		ent state of the Inferior for the atomic business transaction, the last message sent to the Superior by (or in the case of ior			
	status value	meaning/previous message sent			
	active	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.			
	inaccessible	The state information for the relationship with the Superior, if it exists, cannot be accessed at the moment. This should be a transient condition			
	unknown	The Inferior is not known – it does not exist from the perspective of the Superior. The Inferior can be treated as cancelled			
1964 1965 1966 1967 1968	Superior's initiative; "SUPERIOR_STATE	e" if INFERIOR_STATE is sent as a query at the false" if INFERIOR_STATE is sent in reply to a received or other message. Can only be "true" if "status" is "active". ". Can only be "true" if "status" is "active".			
1969 1970 1971	qualifiers standardise	d or other qualifiers.			
1972 1973 1974 1975		IOR_STATE with "reply requested" = "true", should reply on its state) repeating the previous message it sent or by the appropriate status value.			
1976 1977 1978 1979 1980 1981 1982	has no knowledge of a relationship corresponding to the Superior, but SUPERIOR_STATE/*/y (or other)	be sent if it has been determined for certain that the Inferior with the Superior. If there could be persistent information it is not accessible from the entity receiving an message targetted on the Inferior or the entity cannot tent information exists, the response shall be			
1982 1983 1984 1985 1986		lso used as a response to messages, other than eceived when the Inferior is not known (and it is known.			
1980 1987 1988 1989 1990	are in the active state does not request phase commit protocols). The relati	R_STATE exchange that determines that one or both sides uire that the Inferior be cancelled (unlike some other two-tionship between Superior and Inferior, and related nued, with new application messages carrying the same			

1991 1992 1993		Γ. Similarly, if the Inferior is prepared but the Superior is active, there is no npact on the progression of the relationship between them.			
1993 1994 1995 1996 1997 1998 1999	value equiv = "false". I = "true". T	NFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a valent to "abcd" (for active, unknown and inaccessible) and with "reply requested" NFERIOR_STATE/abcd/y refers to a similar message, but with "reply requested" he form INFERIOR_STATE/*/y refers to a INFERIOR_STATE message with nested" = "true" and any value for status.			
2000	REQUEST_CO	NFIRM			
2001 2002 2003 2004 2005		a Terminator to a Decider to request confirmation of the business transaction. If the ransaction is a Cohesion, the confirm-set is specified by the "inferiors-list"			
		Parameter	Туре		
		target address	BTP address		
		reply address	BTP address		
		transaction identifier	Identifier		
		inferiors-list	List of inferior handles		
		Report hazard	boolean		
		Qualifiers	List of qualifiers		
2006		<u> </u>			
2007 2008 2000		target address the address to which the REQUEST_CONFIRM message is sent. This will be the address-as-decider on the BEGUN message.			
2009 2010		reply address the address of the Terminator sending the REQUEST_CONFIRM			
2011		message.			
2012					
2013 2014		transaction identifier identifies the Decider. This will be the transaction-identifier from the BEGUN message.			
2015		identifier from the BEGON message.			
2016		inferiors-list defines which Inferiors enrolled with the Decider, if it is a			
2017		Cohesion Composer, are to be confirmed. Shall be absent if the Decider is an			
2018 2019		Atom Coordinator.			
2020		report hazard Defines whether the Terminator wishes to be informed of hazard			
2021		events and contradictory decisions within the business transaction. If "report			
2022		hazard" is "true", the receiver will wait until responses (CONFIRMED,			
2023		CANCELLED or HAZARD) have been received from all of its inferiors,			
2024 2025		ensuring that any hazard events are reported. If "report hazard" is "false", the			
2023		Decider will reply with CONFIRMED or CANCELLED as soon as the decision for the transaction is known			

for the transaction is known.

2025 2026

2028	qualifiers standardis	ed or other qualifiers.
2029	TC (1 (4) C : 1' (2)	
2030		present, the Inferiors identified shall be the "confirm-set" of
2031		absent and the business transaction is a Cohesion, the
2032		ng Inferiors. If the business transaction is an Atom, the
2033	"confirm-set" is automatically all	the Interiors.
2034		
2035	Any Inferiors from which RESIG	N is received are not counted in the confirm-set.
2036		
2037		ved from any Inferiors in the confirm-set, PREPARE shall
2038	be issued to them.	
2039		
2040	A confirm decision may be made	only if PREPARED has been received from all Inferiors in
2041	the "confirm-set". The making of	the decision shall be persistent (and if it is not possible to
2042	persist the decision, it is not made	e). If there is only one remaining Inferior in the "confirm
2043	set", CONFIRM_ONE_PHASE 1	nay be sent to it.
2044		
2045	All remaining Inferiors that are no	ot in the confirm set shall be cancelled.
2046	Č	
2047	If a confirm decision is made and	"report-hazard" was "false", a CONFIRMED message shall
2048	be sent to the "reply-address".	
2049	1 3	
2050	If a cancel decision is made and '	report-hazard" was "false", a CANCELLED message shall
2051	be sent to the "reply-address".	1
2052	er seem to use septy manager .	
2053	If "report-hazard" was "true" and	any HAZARD or contradictory message was received (i.e.
2054		the confirm-set or CONFIRMED from an Inferior not in
2055		STATUSES reporting the status for all Inferiors shall be sent
2056	to the "reply-address".	of the state of the minimum of shall be sent
2057	to the Teply address .	
2058	Types of FAULT possible (sent t	reply address)
2059	Types of Tite 21 possion (sem s	o Topij dadroso)
2060	General	
2060 2061		ansaction – if the transaction-identifier is unknown
2062		or – if an inferior handle in the inferiors-list is unknown
2063	vvrongstate	- if a CANCEL/whole has already been received .
2064		
2065	• –	/whole refers to a REQUEST_ CONFIRM message where
2066		osent. The form REQUEST_ CONFIRM /inferiors refers to
2067	a REQUEST_ CONFIRM messa	ge where the "inferiors-list" parameter is present.
2068		
2069	REQUEST_STATUSES	
2070		
2071		rt the status of its Inferiors with an
2072	INFERIOR_STATUSES messag	2.
2073		
	Parameter	Type

		target address	BTP address		
		reply address	BTP address		
		transaction identifier	Identifier		
		inferiors-list	List of inferior handles		
		Qualifiers	List of qualifiers		
2074					
2075 2076			target address the address to which the REQUEST_STATUS message is sentThis will be the address-as-decider from the BEGUN message.		
2077 2078 2079		reply address the address to whole sent	nich the replying INFERIOR_STATUSES is to		
2080					
2081 2082		transaction identifier identifies identifier from the BEGUN mes	the Decider. This will be the transaction-sage.		
2083 2084 2085 2086		 inferiors-list defines which inferiors enrolled with the Composer or Coordinator are to be included in the INFERIOR_STATUSES. If the list is absent, the status of all enrolled inferiors will be reported. qualifiers standardised or other qualifiers. AULT possible (sent to reply-address) 			
2087 2088					
2089 2090 2091	Types of F				
2091 2092 2093		General			
2093 2094 2095 2096 2097	list absent.		Fers to a REQUEST_STATUS with the inferiors- nferiors refers to a REQUEST_STATUS with		
2097 2098 2099	INFERIOR_ST	ATUSES			
2100 2101 2102 2103	REQUEST		or some of its inferiors in response to a CEL/inferiors and REQUEST_CONFIRM with		
		Parameter	Туре		
		target address	BTP address		
		address-as-decider	BTP address		
		transaction-identifier	identifier		
		status-list	Set of Status items - see below		
		general-qualifiers	List of qualifiers		

2105 2106	target address the address to which the INFERIOR_STATUSES is sent. This will be the reply address on the received message
2107	
2108	address-as-decider The address-as-decider of the Decider as on the BEGUN
2109	message (with the transaction identifier, this determines who the message is
2110	from)
2111	
2112	transaction identifier The transaction identifier as on the BEGUN message (i.e.
2113	the identifier of the Decider as a whole)
2114	
2115	status-list contains a number of Status-items, each reporting the status of one of
2116	the inferiors of the Decider. The fields of a Status-item are
2117	

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

2121

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

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

	status value	Meaning			
	cancelled	CANCELLED has been received, and PREPARED was not received previously			
	cancel-contradiction	Confirm had been ordered (and may have been sent), but CANCELLED was received			
	confirm-contradiction	Cancel had been ordered (and may have been sent) but CONFIRM/auto was received			
	hazard	A HAZARD message has been received			
2122 2123 2124 2125	INFERIOR_STATUSES	lardised or other qualifiers applying to the sa a whole. Each Status-item contains a "qualifiers" applying to (and received from) the portionary Inferior			
2125	neid containing quaimer	s applying to (and received from) the particular Inferior.			
2127 2128 2129 2130 2131 2132	If the inferiors-list parameter was present on the received message, only the inferiors identified by that parameter shall have their status reported in status-list of this message. If the inferiors-list parameter was absent, the status of all enrolled inferiors shall be reported, except that an inferior that had been reported as <i>cancelled</i> or <i>resigned</i> on a previous INFERIOR_STATUSES message may be omitted (sender's option).				
2133	REQUEST_STATUS				
2134 2135 2136	Sent to an Inferior or to a Decider to	ask it to reply with STATUS.			
	Parameter	Туре			
	target address	BTP address			
	reply address	BTP address			
	inferior identifier	Identifier			
	transaction-identifier	Identifier			
	Qualifiers	List of qualifiers			
2137 2138 2139 2140 2141 2142	If the target is an Inferior	ess to which the REQUEST_ STATUS message is sent. r, this will be the address-as-inferior on the ENROL a Decider, this will be the address-as-decider on the			
2143 2144	reply address the addre	reply address the address to which the replying STATUS should be sent			
2144 2145 2146 2147		inferior identifier If the target is an Inferior, the "inferior-identifier" on the ENROL message. If the target is a Decider, this parameter shall be absent.			
2148 2149	transaction-identifier If the BEGUN message. If	f the target is a Decider, the "transaction-identifier" on the target is an Inferior, this parameter shall be absent.			

2151 2152	qualif	qualifiers standardised or other qualifiers.			
2152 2153 2154	Types of FAULT	FAULT possible (sent to reply address)			
2155 2156		General			
2157 2158 2159	STATUS				
2160 2161 2162		or Decider in reply to a R e node represented by the	EQUEST_STATUS, reporting the overall state of Inferior or Decider.		
2102	Param	eter	Туре		
	target a	address	BTP address		
	addres	s-as-inferior	BTP address		
	inferior	identifier	Identifier		
	addres	s-as-decider	BTP address		
	transac	ction-identifier	Identifier		
	status		See below		
	qualifie	ers	List of qualifiers		
2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180	addres addres ENRO from). inferio ENRO addres BEGU messa transa	target address the address to which the STATUS is sent. This will be the reply address on the REQUEST_STATUS message address-as-inferior If the sender is an Inferior, the address-as-inferior as on the ENROL message (with the inferior-identifier, this determines who the message is from). If the sender is a Decider, this parameter shall be absent inferior-identifier If the sender is an Inferior, the inferior-identifier as on the ENROL message. If the sender is a Decider, this parameter shall be absent. address-as-decider If the sender is a Decider, the address-as-decider on the BEGUN message (with the "transaction-identifier", this determines who the message is from). If the sender is an Inferior, this parameter shall be absent. transaction-identifier If the sender is a Decider, the transaction identifier as on the BEGUN message. If the sender is an Inferior, this parameter shall be absent.			
2181 2182 2183		status states the current status of the transaction tree node represented by the sender.			
	status value	Meaning from Inferior	Meaning from Decider		
	Created	The Inferior exists (and is	Not applicable		

status value	Meaning from Inferior addressable) but it has not been enrolled with a Superior	Meaning from Decider
Enrolling	ENROL has been sent, but ENROLLED is awaited	Not applicable
Active	The Inferior is enrolled	New enrolment of inferiors is possible; no decision has been made.
Resigning	RESIGN has been sent; RESIGNED is awaited	Not applicable
Resigned	RESIGNED has been received	Not applicable
Preparing	PREPARE has been received; PREPARED has not been sent	Not applicable
Prepared	PREPARED has been sent; no outcome has been received or autonomous decision made	Not applicable
Confirming	CONFIRM has been received; CONFIRMED/response has not bee sent	Confirm decision has been made but responses from inferiors are pending
Confirmed	CONFIRMED/response has been sent	CONFIRMED has been sent
Cancelling	CANCEL has been received or auto-cancel has been decided	Cancel decision has been made but responses from inferiors are pending
Cancelled	CANCELLED has been sent	CANCELLED has been sent
cancel- contradiction	Autonomous cancel decision was made, CONFIRM received; CONTRADICTION has not been received	Not applicable
confirm- contradiction	Autonomous confirm decision was made, CANCEL received; CONTRADICTION has not been received	Not applicable
Hazard	A hazard has been discovered; CONTRADICTION has not been received	A hazard has been reported from at least one Inferior
Contradicted	CONTRADICTION has been received	Not applicable
Unknown	No state information for the identifier exists; no such Inferior exists	No state information for the transaction identifier exists; no such Decider exists

	status value	Meaning from Inferior		Meaning from Decider
	Inaccessible	There may be state inform this identifier but it cannot reached/existence cannot determined	be	There may be state information for this identifier but it cannot be reached/existence cannot be determined
2184	aa.l	floro 1 11 1 1	11.01	
2185 2186	quan	fiers standardised or other	r qualifier	S.
2187	REDIRECT			
2188				
2189 2190 2191 2192		rmation is now accessible		r or Inferior is no longer valid and the ferent address (but the same superior or
	Parar	neter	Type	
	targe	address	BTP add	ress
	super	ior identifier	Identifier	
	inferio	or identifier	Identifier	
	oldad	ldress	Set of B7	P addresses
	new a	iddress	Set of B7	P addresses
	qualif	iers	List of qu	ualifiers
2193	·			
2194				REDIRECT is sent. This may be the
2195 2196			-	the address of the opposite side
2190	(supe	(superior/inferior) as given in a CONTEXT or ENROL message		
2198	supe	superior identifier The superior identifier as on the CONTEXT message and		
2199	used	used on an ENROL message. (present only if the REDIRECT is sent from the		
2200 2201	Inter	Inferior).		
2201	infer	inferior identifier The inferior identifier as on the ENROL message		
2203		The interior identifier as on the ENROL message		
2204		daddress The previous address of the sender of REDIRECT. A match is		
2205 2206	consi	onsidered to apply if any of the old addresses match one that is already known.		
2207	new	new address The (set of alternatives) new addresses to be used for messages		
2208		ent to this entity.		
2209		<i>a</i>		
2210	quali	fiers standardised or other	r qualifier	s.
2211 2212	If the	actor whose address is ch	anged is a	n Inferior, the new address value
2213		If the actor whose address is changed is an Inferior, the new address value replaces the address-as-inferior as present in the ENROL.		

2215 2216 2217 2218 2219		replaces the Superior add	s is changed is a Superior, the naturess as present in the CONTEX used to establish the Superior:In	T message (or as present
2220	FAULT			
2221 2222 2223	Sent in rep	oly to various messages to r	report an error condition	
2223		Parameter	Туре	
		target address	BTP address	
		superior identifier	Identifier	
		inferior identifier	Identifier	
		fault type	See below	
		fault data	See below	
		qualifiers	List of qualifiers	
2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240		address from a received of (superior/inferior) as given superior identifier the sused on the ENROL mest inferior identifier the infinite of the FAULT is sent to the fault type identifies the of messages.	ess to which the FAULT is sent. message or the address of the open in a CONTEXT or ENROL nuperior identifier as on the CONsage (present only if the FAULT ferior identifier as on the ENRO the inferior) mature of the error, as specified felevant to the particular error. Ea	posite side nessage ITEXT message and as Γ is sent to the superior). L message (present only for each of the main
		fault type	meaning	fault data
		General	Any otherwise unspecified problem	Free text explanation
		UnknownParameter	A BTP message has been received with an unrecognised parameter	Free text explanation
		WrongState	The message has arrived when the recipient is in an invalid state.	

	CommunicationFailure	Any fault arising from the carrier mechanism and communication infrastructure.	Determined by the carrier mechanism and binding specification
	InvalidSuperior	The received identifier is not known or does not identify a known Superior	The identiifier
	DuplicateInferior	An inferior with the same address and identifier is already enrolled with this Superior	The identiifier
	InvalidInferior	The Superior is known but the Inferior identified by the address-as-inferior and identifier are not enrolled in it	The Inferior Identity (address-as- inferior and identifier)
	UnsupportedQualifier	A qualifier has been received that is not recognised and on which "must-be-Understood" is "true".	Qualifier group and name
2241 2242 2243	qualifiers standardised	or other qualifiers.	
2244	Note: If the corrier mechan	nism used for the transmission of	DTD massages
2244		nism used for the transmission of ssages in a different order than the	
2246		is not sent and should be ignored:	
2247			
2248 2249	Standard qualifiers		
2250	The following qualifiers are expecte	d to be of general use to many ap	plications and
2251	environments. The URI "urn:oasi	s:names:tc:BTP:qualifiers	is used in the
2252	Qualifier group value for the qualifi	ers defined here.	
2253 2254			
	Transaction timelimit		
2255 2256	Transaction unclimit		
2257	The transaction timelimit allows the	Superior (or an application eleme	ent initiating the
2258	business transaction) to indicate the		
2259	indication to the Inferior of when it	* * *	
2260	phase appears to continue too long.		ps) when the Inferior
2261 2262	decides to be prepared and issues PF	REPARED to the Superior.	
2263	It should be noted that the expiry of	the time limit does not change the	e permissible actions of
2264	the Inferior. At any time prior to dec		
2265	permitted to initiate cancellation fo		
2266	entity of when it will be useful to ex		,
	-	-	

2268 2269	The qualifier is propagated on a Co	ONTEXT message.		
2270 2271	The "Qualifier name" shall be "tra	ansaction-timelimit".		
2272 2273	The "Content" shall contain the fo	The "Content" shall contain the following field:		
	Content field	Туре		
	Timelimit	Integer		
2274				
2275 2276 2277 2278		(further) duration, expressed as whole seconds from the ing CONTEXT, of the active phase of the business		
2279	Inferior timeout			
2280 2281 2282 2283 2284 2285 2286	PREPARED, that it will maintain to operations. Without this qualifier,	limit the duration of its "promise", when sending the ability to confirm or cancel the effects of all associated an Inferior is expected to retain the ability to confirm or loes expire, the Inferior is released from its promise and the qualifier.		
2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298	a confirm or cancel decision before timeout expires (or if this qualifier and (as with other transaction mech heuristic decisions, the taking of ar expiry of this timeout, is liable to transaction. BTP ensures that at lea (eventually) reported to the Superior decisions and autonomous decision	ises the possibility that an Inferior may be forced to apply the CONFIRM or CANCEL is received and before this is not used). Such a decision is termed a heuristic decision nanisms), is considered to be an exceptional event. As with a autonomous decision by a Inferior subsequent to the cause contradictory decisions across the business ast the occurrence of such a contradiction will be or of the business transaction. BTP treats "true" heuristic as after timeout the same way – in fact, the expiry in this e (state table) change in what can happen, but rather a step of the business transaction.		
2299 2300 2301 2302 2303 2304 2305	intended decision, only that is at lil apply the decision if there is content Nevertheless, Superiors are recommendated as a superior of the s	strictly require that the Inferior immediately invokes the perty to do so. An implementation may choose to only intion for the underlying resource, for example, mended to avoid relying on this and ensure decisions for perfore these timeouts expire (and allow a margin of error		
2306 2307 2308 2309		PREPARED message. If the PREPARED message has the ", then the "IntendedDecision" field of this qualifier shall		

The "Content" shall contain the following fields:

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

2310 2311

2313		
2313	Content field	Туре
	Timeout	Integer
	IntendedDecision	"confirm" or "cancel"
2314		
2315 2316 2317 2318		s whole seconds from the time of transmission of the maintain its ability to either confirm or cancel the dered by the receiving Superior.
2319	IntendedDecision indicates which outcome	ne will be applied, if the timeout comple tes and an
2320 2321	autonomous decision is made.	
2322 2323	Minimum inferior timeout	
2324 2325 2326 2327 2328 2329 2330	Inferior. If a Superior knows that the deci determined for some period, it can require with Inferior timeouts that would expire b	ain the Inferior timeout qualifier received from the sion for the business transaction will not be that Inferiors do not send PREPARED messages refore then. An Inferior that is unable or unwilling to the total cancel, and reply with
2331 2332 2333 2334 2335	present on more than one, and with different	EXT, ENROLLED or PREPARE message. If ent values of the MinimumTimeout field, the value CONTEXT and the value on PREPARE shall
2336 2337	The "Qualifier name" shall be "minimum	-inferior-timeout".
2338 2339	The "Content" shall contain the following	g field:
	Content field	Туре
	MinimumTimeout	Integer
2340 2341 2342 2343	Minimum Timeout is the minimum value acceptable in the Inferior timeout qualifie	of timeout, expressed as whole seconds, that will be r on an answering PREPARED message.
2344 2345	Inferior name	
2346 2347 2348 2349 2350	INFERIOR_STATUSES and thus allow t Composer or Coordinator) is related to wl	a name for the Inferior that will be visible on the Terminator to determine which Inferior (of the nich application work. This is in addition to the thuman-readable and can also be used in fault

2352	The name is never used by the BTP actors the	nemselves to identify each other or to direct
2353	messages. (The BTP actors use the addresse	s and the identifiers in the message parameters
2354	for those purposes.)	
2355		
2356	*	t the names are unambiguous within any scope
2357	(unlike the "inferior-handle" on ENROLLEI	•
2358		r). Other specifications, including those defining
2359		y place requirements on the use and form of the
2360	•	nation passed in application messages or in other,
2361	non-standardised, qualifiers.)	
2362		
2363		ROL and in the "qualifiers" field of a Status-item
2364		BEGIN only if there is a related CONTEXT; if
2365	present, the same qualifier value should be	^
2366	INFERIOR_STATUSES includes a Status-i	
2367	inferior-name qualifier, the same qualifier va	alue should be included in the Status-item.
2368		
2369	The "Qualifier-name" shall be "inferior-	name"
2370		
2371	The "Content" shall contain the following fi	elds:
2372		
	Content field	Туре
	inferior-name	String
2373		
2374	Inferior name the name assigned to the enro	lling Inferior.
2375		

State Tables

Explanation of the state tables

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

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

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

Status queries

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

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

Decision events

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

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

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 (inferior) implementation that "decided to be prepared", and recorded a timeout (to cancel) in the persistent information for that decision (signalled via the appropriate qualifier on PREPARED), could treat the presence of an expired record as a record of "decide to cancel autonomously", provided it always updated such a record as part of the "apply ordered confirmation" decision event.

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

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

Disruptions - failure events

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

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

Invalid cells and assumptions of the communication mechanism

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

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

For all communication mechanisms, it is assumed that

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

 b) messages may be lost arbitrarily

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

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

Meaning of state table events

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

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

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

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

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

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

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

Table 1: send, receive and disruption events

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

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

Table 2 : Decision events for Inferior

Event name	Meaning
decide to resign	Any associated operations have had no effect (data state is unchanged)).
decide to be prepared	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	As "decide to be prepared";
	 the persistent information specifies that the default action will be to cancel
decide to confirm autonomously	Decision to confirm autonomously has been made persistent;
	 the effects of associated operations will be confirmed regardless of failures
decide to cancel autonomously	Decision to cancel autonomously has been made persistent
	 the effects of associated operations will be cancelled regardless of failures
apply ordered confirmation	Effects of all associated operations have been confirmed;
	Persistent information is effectively removed
remove persistent information	Persistent information is effectively removed;
detect problem	For at least some of the associated operations, EITHER
	 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

Event name	Meaning
detect and record problem	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)

Table 3: Decision events for a Superior

Event name	Meaning	
decide to request confirm	 All associated application messages to be sent to the service have been sent; 	
	There are no other remaining Inferiors	
	 All enrolments that would create other Inferiors have completed (no outstanding CONTEXT_REPLYs) 	
	The Superior has been requested to confirm	
decide to prepare	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	Either	
	 PREPARED or PREPARED/cancel has been received from all other remaining Inferiors; AND 	
	 Superior has been requested to confirm; AND 	
	 persistent information records the confirm decision and identifies all remaining Inferiors; 	
	• Or	
	 persistent information records an offer of confirmation and has been instructed to confirm 	
decide to cancel	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	Persistent information has been effectively removed;	
record contradiction	 Information recording the contradiction has been persisted (to the degree considered appropriate) 	

2542 2543 2544

Persistent information

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

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

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

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

Table 4 : Superior states

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

Table 5 : Inferior states

C1 1	
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
12	autonomously confirmed, CONTRADICTION received
m1	confirmation applied
n1	cancelling
p1	hazard detected, not recorded
p2	hazard detected in prepared state, not recorded
q1	hazard recorded
s1	REQUEST CONFIRM received after prepared state
s2	REQUEST CONFIRM received
s3	REQUEST CONFIRM received, confirming
s4	REQUEST CONFIRM received, cancelling
s5	REQUEST CONFIRM received, hazard detected
s6	REQUEST CONFIRM received, hazard recorded
x1	completed, presuming abort
x2	completed, presuming abort after prepared/cancel
y1	completed, queried

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

Table 6: Superior state table – normal forward progression

	I 1	A1	B1	C1	D1	E1	E2	F1	F2
receive ENROL/rsp-req	A1								
receive ENROL/no-rsp-req	B 1								
receive RESIGN/rsp-req	Y1		C1	C1	C1				
receive RESIGN/no-rsp-req	Z		Z	Z	Z				
receive PREPARED	Y1		E 1		E1	E1		F1	
receive PREPARED/cancel	Y1		E2		E2		E2	F1	
receive CONFIRMED/auto	Q1		H1		H1	H1		F1	
receive CONFIRMED/response								F2	F2
receive CANCELLED	Y1		Z		Z	J1	J1	K1	
receive HAZARD	P1	P1	P1		P1	P1	P1	Р3	
receive INF_STATE/active/y	Y1	A1	B1		D1				
receive INF_STATE/active			B1		D1				
receive INF_STATE/unknown			Z	Z	Z				
send ENROLLED		B 1							
send RESIGNED				Z					
send PREPARE					D1	E1	E2		
send CONFIRM_ONE_PHASE									
send CONFIRM								F1	
send CANCEL									
send CONTRADICTION									
send SUP_STATE/active/y			B 1						
send SUP_STATE/active			B 1						
send SUP_STATE/prepared-rcvd/y						E1	E2		
send SUP_STATE/prepared-rcvd						E1	E2		
send SUP_STATE/unknown									
decide to request confirm			S1			S1	S1		
decide to prepare			D1						
decide to confirm						F1	F1		
decide to cancel			G1		G1	G1	Z		
remove persistent information									Z
record contradiction									
disruption I	Z	Z	Z	Z	Z	Z	Z		F1
disruption II						D1	D1		
disruption III						B1	B 1		
disruption IV									

 $Table \ 7 \hbox{:} \ Superior \ state \ table - cancellation \ and \ contradiction$

	G1	G2	G3	G4	H1	J1	K1	L1
receive ENROL/rsp-req								
receive ENROL/no-rsp-req								
receive RESIGN/rsp-req	G3	Z	G3					
receive RESIGN/no-rsp-req	Z	Z	Z					
receive PREPARED	G1	G2						
receive PREPARED/cancel	G1	G2						
receive CONFIRMED/auto	L1	L1			H1			L1
receive CONFIRMED/response								
receive CANCELLED	G4	Z		G4		J1	K1	
receive HAZARD	P4	P4						
receive INF_STATE/active/y	G1	G2						
receive INF_STATE/active	G1	G2						
receive INF_STATE/unknown	Z	Z	Z	Z				
send ENROLLED								
send RESIGNED								
send PREPARE								
send CONFIRM_ONE_PHASE								
send CONFIRM								
send CANCEL	G2	G2	Z	Z				
send CONTRADICTION								
send SUP_STATE/active/y								
send SUP_STATE/active								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
decide to request confirm								
decide to prepare								
decide to confirm					F1	K1		
decide to cancel					L1	G4		
remove persistent information								
record contradiction							R1	R1
disruption I	Z	Z	Z	Z	Z	Z	F1	Z
disruption II			G2	G2	E 1	E1		G2
disruption III					D1	D1		
disruption IV					B1	B1		

Table 8: Superior state table – hazard and request confirm

	P1	P2	Р3	P4	Q1	R1	R2	S1
receive ENROL/rsp-req								
receive ENROL/no-rsp-req								
receive RESIGN/rsp-req								C1
receive RESIGN/no-rsp-req								Z
receive PREPARED								S1
receive PREPARED/cancel								S1
receive CONFIRMED/auto					Q1	R1	R1	S1
receive CONFIRMED/response					Z	R2		Z
receive CANCELLED						R1	R1	Z
receive HAZARD	P1	P2	P3	P4		R1	R1	Z
receive INF_STATE/active/y								S1
receive INF_STATE/active								S1
receive INF_STATE/unknown	P1	P2		P4		R2	R2	Z
send ENROLLED								
send RESIGNED								
send PREPARE								
send CONFIRM_ONE_PHASE								S1
send CONFIRM								
send CANCEL								
send CONTRADICTION						R2		
send SUP_STATE/active/y								
send SUP_STATE/active								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
decide to request confirm								
decide to prepare								
decide to confirm								
decide to cancel								
remove persistent information							Z	
record contradiction	R1	R1	R1	R1	R1			
disruption I	Z	Z	Z	Z	Z		R1	Z
disruption II	D1		F1	G2				
disruption III	B1							
disruption IV								

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

	Y1	Z
receive ENROL/rsp-req		Y1
receive ENROL/no-rsp-req		Y 1
receive RESIGN/rsp-req	Y1	Y1
receive RESIGN/no-rsp-req	Z	Z
receive PREPARED	Y1	Y1
receive PREPARED/cancel	Y1	Y1
receive CONFIRMED/auto	Q1	Q1
receive CONFIRMED/response	Z	Z
receive CANCELLED	Y1	Y1
recei ve HAZARD	P2	P2
receive INF_STATE/active/y	Y1	Y 1
receive INF_STATE/active	Y1	Z
receive INF_STATE/unknown	Z	Z
send ENROLLED		
send RESIGNED		
send PREPARE		
send CONFIRM_ONE_PHASE		
send CONFIRM		
send CANCEL		
send CONTRADICTION		
send SUP_STATE/active/y		
send SUP_STATE/active		
send SUP_STATE/prepared-rcvd/y		
send SUP_STATE/prepared-rcvd		
send SUP_STATE/unknown	Z	
decide to request confirm		
decide to prepare		
decide to confirm		
decide to cancel		
remove persistent information		
record contradiction		
di srupti on I	Z	
disruption II		
disruption III		
disruption IV		

Table 10: Inferior state table – normal forward progression

	i 1	a1	b1	с1	d1	e1	e2	f1	f2
send ENROL/rsp-req	a1								
send ENROL/no-rsp-req	b 1								
send RESIGN/rsp-req				c1					
send RESIGN/no-rsp-req				\mathbf{z}					
send PREPARED						e 1			
send PREPARED/cancel							e2		
send CONFIRMED/auto									
send CONFIRMED/response									
send CANCELLED			Z		Z				
send HAZARD									
send INF_STATE/active/y		a1	b 1		d1				
send INF_STATE/active			b1		d1				
send INF_STATE/unknown									
receive ENROLLED		b1							
receive RESIGNED				\mathbf{z}					
receive PREPARE		d1	d1	c1	d1	e 1	e2		
receive CONFIRM_ONE_PHASE		s2	s2	c1		s1	s1		
receive CONFIRM						f1	f2	f1	f2
receive CANCEL		n1	n1	\mathbf{z}	n1	g1	g2		
receive CONTRADICTION									
receive SUP_STATE/active/y		b1	b1	c1		e 1	e2		
receive SUP_STATE/active		b1	b 1	c1		e 1	e2		
receive SUP_STATE/prepared-rcvd/y						e 1	e2		
receive SUP_STATE/prepared-rcvd						e 1	e2		
receive SUP_STATE/unknown		Z	Z	Z	z	x 1	x2		
decide to resign			c1		c1				
decide to be prepared			e1		e 1				
decide to be prepared/cancel			e2		e2				
decide to confirm autonomously						h1			
decide to cancel autonomously						j 1	z1		
apply ordered confirmation								m1	m1
remove persistent information									
detect problem		p1	p1		p1	p2	p2	p2	p2
detect and record problem									
disruption I		Z	\mathbf{z}	\mathbf{z}	Z			e 1	e2
disruption II					b1				
disruption III									

Table 11: Inferior state table – cancellation and contradiction

	g1	g2	h1	h2	j 1	j 2	k1	k2	l 1	12
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			h1						11	
send CONFIRMED/response										
send CANCELLED					j 1		k1			
send HAZARD										
send INF_STATE/active/y										
send INF_STATE/active										
send INF_STATE/unknown										
receive ENROLLED										
receive RESIGNED										
receive PREPARE			h1		j 1					
receive CONFIRM_ONE_PHASE			s3		s4					
receive CONFIRM			h2	h2	k1		k1			
receive CANCEL	g1	g2	l 1		j 2	j 2			l 1	
receive CONTRADICTION			12		k2		k2	k2	12	12
receive SUP_STATE/active/y			h1		j 1					
receive SUP_STATE/active			h1		j 1					
receive SUP_STATE/prepared-rcvd/y			h1		j 1					
receive SUP_STATE/prepared-rcvd			h1		j 1					
receive SUP_STATE/unknown	x 1	x2	l 1		j 2	j 2	k2	k2	11	
decide to resign										
decide to be prepared										
deci de to be prepared/cancel										
decide to confirm autonomously										
decide to cancel autonomously										
apply ordered confirmation										
remove persistent information	n1	n1		m1		\mathbf{z}		\mathbf{z}		Z
detect problem	p2	p2								
detect and record problem										
disruption I	e 1	e2		h1		j 1	j 1	k1	h1	l 1
disruption II								j 1		h1
disruption III										

send ENROL/rsp-req m1 n1 p1 p2 q1 send ENROL/no-rsp-req send RESI GN/rsp-req send RESI GN/rsp-req send RESI GN/no-rsp-req send PREPARED send PREPARED/cancel send CONFI RMED/auto send CONFI RMED/response z send CANCELLED z p1 p2 q1 send LNF_STATE/active/y send INF_STATE/active/y p1 p2 q1 send INF_STATE/unknown receive ENROLLED p1 p1 q1 receive PREPARE p1 p2 q1 receive CONFI RM_ONE_PHASE s5 s5 s6 receive CONFI RM m1 p1 p2 q1 receive CONTRADI CTI ON z z z z receive SUP_STATE/active/y p1 p2 q1 receive SUP_STATE/active/y p1 p2 q1
send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM m1 receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y send INF_STATE/active/y
send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto z send CONFIRMED/response z send CANCELLED z send INF_STATE/active/y p1 p2 q1 send INF_STATE/active p1 q1 receive ENROLLED p1 q1 receive RESIGNED p1 p2 q1 receive PREPARE p1 p2 q1 receive CONFIRM_ONE_PHASE s5 s5 s6 receive CANCEL n1 p1 p2 q1 receive CONTRADICTION z z z z receive SUP_STATE/active/y p1 p2 q1
send RESIGN/no-rsp-req
send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y
send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y
send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y send INF_STATE/unknown p1 p2 q1 q1 q1 q1 q1 q2 q1 q2 q1 q2 q1 q2 q1 q2 q2
send CONFIRMED/response z z z z z g q
send CANCELLED z p1 p2 q1 send HAZARD p1 p2 q1 send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown send INF_STATE/unknown p1 q1 receive ENROLLED p1 p1 q1 receive RESIGNED p1 p2 q1 receive PREPARE p1 p2 q1 receive CONFIRM_ONE_PHASE s5 s5 s6 receive CONFIRM m1 p2 q1 receive CANCEL n1 p1 p2 q1 receive CONTRADICTION z z z z receive SUP_STATE/active/y p1 p2 q1
send HAZARD p1 p2 q1 send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown send INF_STATE/unknown p1 q1 receive ENROLLED p1 q1 q1 receive RESIGNED p1 p2 q1 receive PREPARE p1 p2 q1 receive CONFIRM_ONE_PHASE s5 s5 s6 receive CONFIRM m1 p2 q1 receive CANCEL n1 p1 p2 q1 receive CONTRADICTION z
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y
send INF_STATE/active p q1 receive ENROLLED p1 q1 receive RESIGNED p1 p2 q1 receive PREPARE p1 p2 q1 receive CONFIRM_ONE_PHASE s5 s5 s6 receive CONFIRM m1 p2 q1 receive CANCEL n1 p1 p2 q1 receive CONTRADICTION z z z z receive SUP_STATE/active/y p1 p2 q1
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
receive ENROLLED p1 q1 receive RESIGNED receive PREPARE p1 p2 q1 receive PREPARE s5 s5 s6 receive CONFIRM_ONE_PHASE m1 p2 q1 receive CONFIRM m1 p2 q1 receive CANCEL n1 p1 p2 q1 receive CONTRADICTION z z z z z z z z q1 receive SUP_STATE/active/y p1 p2 q1 p2 q1
receive RESIGNED p1 p2 q1 receive PREPARE p1 p2 q1 receive CONFIRM_ONE_PHASE s5 s5 s6 receive CONFIRM m1 p2 q1 receive CANCEL n1 p1 p2 q1 receive CONTRADICTION z z z receive SUP_STATE/active/y p1 p2 q1
receive PREPARE p1 p2 q1 receive CONFIRM_ONE_PHASE s5 s5 s6 receive CONFIRM m1 p2 q1 receive CANCEL n1 p1 p2 q1 receive CONTRADICTION z z z receive SUP_STATE/active/y p1 p2 q1
receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y n1
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$
receive CANCEL n1 p1 p2 q1 receive CONTRADICTION z z z receive SUP_STATE/active/y p1 p2 q1
receive CONTRADICTION z z z z receive SUP_STATE/active/y p1 p2 q1
receive SUP_STATE/active/y p1 p2 q1
receive SUP_STATE/active p1 p2 q1
receive SUP_STATE/prepared-rcvd/y p2 q1
receive SUP_STATE/prepared-rcvd p2 q1
receive SUP_STATE/unknown z p1 p2 q1
decide to resign
decide to be prepared
decide to be prepared/cancel
decide to confirm autonomously
decide to cancel autonomously
apply ordered confirmation
remove persistent information
detect problem
detect and record problem q1 q1
disruption I z z z
disruption II d1
ursiupcion ii ui

Table 13: Inferior state table – request confirm states

	s1	s2	s3	s4	s5	s6
send ENROL/rsp-req						
send ENROL/no-rsp-req						
send RESIGN/rsp-req						
send RESIGN/no-rsp-req						
send PREPARED						
send PREPARED/cancel						
send CONFIRMED/auto						
send CONFIRMED/response			\mathbf{z}			
send CANCELLED				\mathbf{z}		
send HAZARD					Z	Z
send INF_STATE/active/y						
send INF_STATE/active						
send INF_STATE/unknown						
receive ENROLLED						
receive RESIGNED						
receive PREPARE						
receive CONFIRM_ONE_PHASE	s1	s2	s3	s4	s5	s6
receive CONFIRM						
receive CANCEL						
receive CONTRADICTION			s3		\mathbf{Z}	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	x 1	Z	Z	Z	Z	Z
decide to resign						
decide to be prepared						
decide to be prepared/cancel						
decide to confirm autonomously		s3				
decide to cancel autonomously		s4				
apply ordered confirmation						
remove persistent information	s2					
detect problem						
detect and record problem		s 6				
disruption I	e 1	Z		Z	\mathbf{z}	
disruption II						
disruption III						

send ENROL/rsp-req x2 y1 y2 z z1 send ENROL/no-rsp-req send ENROL/no-rsp-req send RESIGN/no-rsp-req send RESIGN/no-rsp-req send RESIGN/no-rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED send CONFIRMED/auto send CONFIRMED/auto send CONFIRMED/response send CONFIRMED/response send CONFIRMED/response send INF_STATE/active/y send INF_STATE/active/y send INF_STATE/active send INF_STATE/active send INF_STATE/active z z receive ENROLLED z z receive ENROLLED z z z receive ENROLED z z z z z receive ENROLED z z z z z receive ENROLED z z z z z z receive ENROLED z z z z receive ENGONFIRM y1 y2 y1 y1 y2 y1 y1 y2 y1 y1 y2 y1 y1 y2 y2 y2 z z z z z z							
send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/rsp-req send RESIGN/rsp-req send RESIGN/rsp-req send PREPARED send PREPARED send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active/ send INF_STATE/active receive ENROLLED receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/active r		x1	x 2	y1	y2	Z	z1
send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active s	send ENROL/rsp-req						
send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/active suP_STATE/active suP_STATE/repepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/brepared-rcvd/y receive SUP_STATE/unknown x1 x2 y1 y2 y2 y2 receive SUP_STATE/unknown x1 x2 y1 y2 z z receive SUP_STATE/onclive suP_STATE/onclive receive	send ENROL/no-rsp-req						
send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/unknown x1 x2 y1 y2 z z decide to resign decide to be prepared decide to be prepared/cancel decide to cancel autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect and record problem disruption I disruption II	send RESIGN/rsp-req						
send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive RESIGNED receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd decide to resign decide to be prepared decide to be prepared decide to cancel autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect and record problem disruption II	send RESIGN/no-rsp-req						
send CONFIRMED/response send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/unknown x1 x2 y1 y2 y2 receive SUP_STATE/unknown x1 x2 y1 y2 y2 receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive	send PREPARED						
send CONFIRMED/response send CANCELLED send HAZARD 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 CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/unknown x1 x2 y1 y2 y2 receive SUP_STATE/unknown x1 x2 y1 y2 y2 receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active rec	send PREPARED/cancel						
send CANCELLED send HAZARD send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive CONFIRM_ONE_PHASE receive CONFIRM receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/	send CONFIRMED/auto						
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/active receive SUP_STATE/prepared-decide to be prepared decide to resign decide to cancel autonomously decide to cancel autonomously apply ordered confirmation remove persistent information remove persistent information disruption I disruption II	send CONFIRMED/response						
send INF_STATE/active/y send INF_STATE/unknown receive ENROLLED receive RESIGNED receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/unknown x1 x2 y1 y2 y2 y2 decide to resign decide to be prepared decide to be prepared decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem disruption II	send CANCELLED				z1		
send INF_STATE/unknown receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd decide to resign decide to be prepared decide to be prepared decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem disruption II	send HAZARD						
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM_ONE_PHASE receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_sta	send INF_STATE/active/y						
receive ENROLLED Image: Company of the co	send INF_STATE/active						
receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/oreared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/oreared-rcvd receive SUP_STA	send INF_STATE/unknown			Z			
receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CONFIRM receive CONFIRM receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_STATE/active receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/active receive SUP_S	receive ENROLLED					z	
receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown x1 x2 y1 y2 z z decide to resign decide to be prepared decide to be prepared decide to confirm autonomously decide to cancel autonomously decide to cancel autonomously apply ordered confirmation remove persistent information remove persistent information detect problem detect and record problem disruption I disruption II	receive RESIGNED			y 1		Z	
receive CONFIRM receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown receive SUP_STATE/prepared-rcvd receive SUP_STA	receive PREPARE			y1	y2	y 1	z1
receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown x1 x2 y1 y2 z z1 decide to resign decide to be prepared decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information remove persistent information detect problem detect and record problem disruption I disruption II	receive CONFIRM_ONE_PHASE			y 1	y2	y 1	y1
receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown x1 x2 y1 y2 z z decide to resign decide to be prepared decide to be prepared decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information remove persistent information detect problem detect and record problem disruption I disruption II	receive CONFIRM				y2	m1	y2
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown x1 x2 y1 y2 z z decide to resign decide to be prepared decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information remove persistent information detect problem detect and record problem disruption I disruption II	receive CANCEL			y 1	\mathbf{z}	y1	y1
receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown x1 x2 y1 y2 z z decide to resign decide to be prepared decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information remove persistent information detect problem detect and record problem disruption I disruption II	receive CONTRADICTION			\mathbf{z}	\mathbf{Z}	Z	Z
receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown x1 x2 y1 y2 z z decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information remove persistent information detect problem detect and record problem disruption I disruption II	receive SUP_STATE/active/y			y 1	y2	y1	y2
receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown 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 disruption I disruption II	receive SUP_STATE/active			y1	y2	Z	z1
receive SUP_STATE/unknown x1 x2 y1 y2 z z decide to resign decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem disruption I disruption II	receive SUP_STATE/prepared-rcvd/y				y2		y2
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 disruption I disruption II					y2		y2
decide to be prepared decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem disruption I disruption II	receive SUP_STATE/unknown	x1	x2	y 1	y2	Z	Z
decide to be prepared/cancel decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem disruption I disruption II	decide to resign						
decide to confirm autonomously decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem disruption I disruption II	decide to be prepared						
decide to cancel autonomously apply ordered confirmation remove persistent information detect problem detect and record problem disruption I disruption II	decide to be prepared/cancel						
apply ordered confirmation remove persistent information detect problem detect and record problem disruption I disruption II	decide to confirm autonomously						
remove persistent information z z detect problem detect and record problem e1 e2 disruption II							
detect problem detect and record problem disruption I disruption II	apply ordered confirmation						
detect and record problem disruption I e1 e2 disruption II	-	Z	Z				
disruption I e1 e2 disruption II	-						
disruption II	detect and record problem						
·	-	e 1	e2				
disruption III	disruption II						
	disruption III						

2590

Failure Recovery

Types of failure

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

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

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

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

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

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

Persistent information

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

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

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

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

Inferior identity (this may be an index used to locate the information) Superior address (as on CONTEXT)

 Superior identifier (as on CONTEXT) default-is-cancel value (as on PREPARED)

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

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

Inferior address (as on ENROL) Inferior identifier (as on ENROL)

A Superior that is the Decider for the business transaction need only persist this information if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as atom in a cohesion, sub-coordinator or sub-composer) must persist this information as Superior (to this Inferior) as part of the persistent information of its decision to be prepared (as an Inferior). For such an entity, the "decision to confirm" as Superior is made when (and if) 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. 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).

After failure, an implementation may not be able to restore an endpoint to the appropriate state immediately – in particular, the necessary persistent information may be inaccessible, although the implementation can respond to received BTP messages. In such a case, a Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a "reply-requested" value "false") with SUPERIOR_STATE/inaccessible and an Inferior to any BTP message except SUPERIOR_STATE/* with "INFERIOR_STATE/inaccessible. Receipt of the *_STATE/inaccessible messages has no effect on the endpoint state.

Redirection

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

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

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

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

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

Terminator: Decider failures

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

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

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

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:xml

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

Addresses

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

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

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

Oualifiers

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

Examples:

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

Identifiers

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

```
Examples: "01", "FAB224234CCCC2"
```

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

Message References

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

Messages

CONTEXT

CONTEXT-REPLY

BEGIN

```
2884
                 </br></btp:qualifiers>
2885
               </btp:begin>
2886
2887
          BEGUN
2888
2889
2890
               <btp:begun id? transaction-type="cohesion|atom">
2891
                  <btp:target-additional-information>
2892
                    ...additional address information...
2893
                  </btp:target-additional-information>
2894
                 <btp:decider-address> ?
2895
                    ...address...
2896
                  </br></br></decider-address>
2897
                  <btp:transaction-identifier>...hexstring...
2898
               identifier> ?
2899
                  <btp:inferior-handle>...hexstring...</ptp:inferior:handle> ?
2900
                  <btp:inferior-address> ?
2901
                    ...address...
2902
                  </br></btp:inferior-address>
2903
                  <btp:qualifiers> ?
2904
                    ...qualifiers...
2905
                 </br></btp:qualifiers>
2906
               </btp:begun>
2907
2908
          ENROL
2909
2910
2911
               <btp:enrol reply-requested="true | false" id?>
2912
                  <btp:target-additional-information>
2913
                    ...additional address information...
2914
                  </btp:target-additional-information>
2915
                  <btp:superior-identifier>...hexstring.../btp:superior-
2916
               identifier>
2917
                 <btp:reply-address> ?
2918
                    ...address...
2919
                  </br></btp:reply-address>
2920
                 <btp:inferior-address> +
2921
                    ...address...
2922
                  </br></bbp:inferior-address>
2923
                  <btp:inferior-identifier>...hexstring...
2924
               identifier>
2925
                  <btp:qualifiers> ?
2926
                    ...qualifiers...
2927
                  </br></btp:qualifiers>
2928
               </btp:enrol>
2929
2930
          ENROLLED
2931
2932
2933
               <btp:enrolled id?>
2934
               <btp:target-additional-information>
2935
                    ...additional address information...
```

</btp:target-additional-information>

```
2937
                 <btp:inferior-identifier>...hexstring...
2938
               identifier>
2939
                 <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?
2940
                 <btp:qualifiers> ?
2941
                   ...qualifiers...
2942
                 </br></btp:qualifiers>
2943
               </btp:enrolled>
2944
2945
```

RESIGN

2946

2964

2965 2966 2967

2968

2969

2970

2971

2972

2973

2974

2975

2976

2977 2978

2979

```
2947
2948
               <btp:resign response-requested="true|false" id?>
2949
               <btp:target-additional-information>
2950
                   ...additional address information...
2951
                 </btp:target-additional-information>
2952
                 <btp:superior-identifier>...hexstring...
2953
               identifier>
2954
                 <btp:inferior-address> +
2955
                   ...address...
2956
                 </br></btp:inferior-address>
2957
                 <btp:inferior-identifier>...hexstring...
2958
               identifier>
2959
                 <btp:qualifiers> ?
2960
                   ...qualifiers...
                 </br></btp:qualifiers>
2961
2962
               </btp:resign>
2963
```

RESIGNED

PREPARE

```
2980
2981
               <btp:prepare id?>
2982
                 <btp:target-additional-information>
2983
                   ...additional address information...
2984
                 </btp:target-additional-information>
2985
                 <btp:inferior-identifier>...hexstring...
2986
               identifier> ?
2987
                 <btp:reply-address> ?
2988
                   ...address...
2989
                 </br></btp:reply-address>
```

```
2990
                 <btp:transaction-identifier>...hexstring...
2991
               identifier> ?
2992
                 <btp:inferiors-list> ?
2993
                      <btp:inferior-handle>...hexstring...</ptp:inferior-handle>
2994
2995
                 </br></rbtp:inferiors-list>
2996
                 <btp:qualifiers> ?
2997
                   ...qualifiers...
2998
                 </br></btp:qualifiers>
2999
               </btp:prepare>
3000
3001
```

PREPARED

```
3003
3004
               <btp:prepared default-is-cancel="false|true" id?>
3005
                 <btp:target-additional-information>
3006
                   ...additional address information...
3007
                 </btp:target-additional-information>
3008
                 <btp:superior-identifier>...hexstring...
3009
               identifier>
3010
                 <btp:inferior-address> +
3011
                   ...address...
3012
                 </br></btp:inferior-address>
3013
                 <btp:inferior-identifier>...hexstring...
3014
               identifier>
3015
                 <btp:qualifiers> ?
3016
                   ...qualifiers...
3017
                 </br></btp:qualifiers>
3018
               </btp:prepared>
3019
```

CONFIRM

CONFIRMED

```
3043
                <btp:inferior-address> ?
3044
                   ...address...
3045
                </br></bbp:inferior-address>
3046
                <btp:inferior-identifier>...hexstring...
3047
              identifier> ?
3048
                <btp:decider-address> ?
3049
                   ...address...
3050
                </br></bul>
3051
                <btp:transaction-identifier>...hexstring...
3052
              identifier> ?
3053
                <btp:qualifiers> ?
3054
                   ...qualifiers...
3055
                </br></btp:qualifiers>
3056
              </br></bbp:confirmed>
3057
3058
```

CANCEL

3059

3080

3081

```
3060
3061
               <br/><btp:cancel id?>
3062
                 <btp:target-additional-information>
3063
                   ...additional address information...
3064
                 </btp:target-additional-information>
3065
                 <btp:inferior-identifier>...hexstring...
3066
               identifier> ?
3067
                 <btp:reply-address> ?
3068
                   ...address...
3069
                 </btp:reply-address>
3070
                 <btp:transaction-identifier>...hexstring...
3071
               identifier> ?
3072
                 <btp:inferiors-list> ?
3073
                      <btp:inferior-handle>...hexstring...</ptp:inferior-handle>
3074
                 </br></ri></ri>
3075
                 <btp:qualifiers> ?
3076
                   ...qualifiers...
3077
                 </br></btp:qualifiers>
3078
               </br></bbp:cancel>
3079
```

CANCELLED

```
3082
3083
               <btp:cancelled id?>
3084
                 <btp:target-additional-information>
3085
                   ...additional address information...
3086
                 </btp:target-additional-information>
3087
                 <btp:superior-identifier>...hexstring.../btp:superior-
3088
               identifier>
3089
                 <btp:inferior-address> +
3090
                   ...address...
3091
                 </btp:inferior-address> ?
3092
                 <btp:inferior-identifier>...hexstring...
3093
               identifier> ?
3094
                 <btp:decider-address> ?
3095
                   ...address...
3096
                 </br></decider-address>
```

```
<btp:transaction-identifier>...hexstring...
3098
               identifier> ?
3099
                 <btp:qualifiers> ?
3100
                   ...qualifiers...
3101
                 </br></btp:qualifiers>
3102
               </br></bbp:cancelled>
3103
3104
         HAZARD
3105
3106
3107
               <br/><btp:hazard id?>
3108
                 <btp:target-additional-information>
                   ...additional address information...
3109
3110
                 </btp:target-additional-information>
3111
                 <btp:superior-identifier>...hexstring...
3112
               identifier>
3113
                 <btp:inferior-address> +
3114
                   ...address...
3115
                 </br></btp:inferior-address>
3116
                 <btp:inferior-identifier>...hexstring...
3117
               identifier>
3118
                 <btp:qualifiers> ?
3119
                   ...qualifiers...
                 </br></btp:qualifiers>
3120
3121
               </btp:hazard>
3122
3123
         CONTRADICTION
3124
3125
3126
               <btp:contradiction id?>
3127
                 <btp:target-additional-information>
3128
                   ...additional address information...
3129
                 </btp:target-additional-information>
3130
                 <btp:inferior-identifier>...hexstring...
3131
               identifier>
3132
                 <btp:qualifiers> ?
3133
                   ...qualifiers...
3134
                 </br></btp:qualifiers>
3135
               </br></btp:contradiction>
3136
3137
         SUPERIOR_STATE
3138
3139
3140
               <btp:superior-state reply-requested="true|false" id?>
3141
                 <btp:target-additional-information>
3142
                   ...additional address information...
3143
                 </btp:target-additional-information>
3144
                 <btp:inferior-identifier>...hexstring...
3145
               identifier>
3146
                 <btp:status>active|prepared-
3147
               received | inaccessible | unknown < / btp:status>
3148
                 <btp:qualifiers> ?
```

...qualifiers...

3097

```
3150
                 </br></btp:qualifiers>
3151
               </br></btp:superior-state>
3152
3153
3154
3155
         INFERIOR_STATE
3156
3157
               <btp:inferior-state reply-requested="true|false" id?>
3158
                 <btp:target-additional-information>
3159
                   ...additional address information...
3160
                 </btp:target-additional-information>
3161
                 <btp:superior-identifier>...hexstring...
               identifier>
3162
3163
                 <btp:inferior-address> +
3164
                   ...address...
3165
                 </br></btp:inferior-address>
3166
                 <btp:inferior-identifier>...hexstring...
3167
               identifier>
3168
                 <btp:status> active|prepared-
3169
               received | inaccessible | unknown < / btp: status >
3170
                 <btp:qualifiers> ?
3171
                   ...qualifiers...
3172
                 </br></btp:qualifiers>
3173
               </br></rbtp:inferior-state>
3174
3175
3176
         CONFIRM_ONE_PHASE
3177
3178
3179
               <btp:confirm-one-phase report-hazard="true|false" id?>
3180
                 <btp:target-additional-information>
                   ...additional address information...
3181
3182
                 </btp:target-additional-information>
3183
                 <btp:inferior-identifier>...hexstring...
3184
               identifier>
3185
                 <btp:qualifiers> ?
3186
                   ...qualifiers...
3187
                 </br></btp:qualifiers>
3188
               </br></btp:confirm-one-phase>
3189
3190
3191
         REQUEST_CONFIRM
3192
3193
               <btp:request_confirm report-hazard="true|false" id?>
3194
                 <btp:target-additional-information>
3195
                   ...additional address information...
3196
                 </btp:target-additional-information>
3197
                 <btp:reply-address>
3198
                   ...address...
3199
                 </btp:reply-address>
3200
                 <btp:transaction-identifier>...hexstring...
3201
               identifier>
```

<btp:inferiors-list> ?

REQUEST_STATUSES

3212

3232 3233

```
3213
3214
               <btp:request_statuses id?>
3215
                <btp:target-additional-information>
3216
                   ...additional address information...
3217
                </btp:target-additional-information>
3218
                <btp:reply-address>
3219
                   ...address...
3220
                </btp:reply-address>
3221
                <btp:transaction-identifier>...hexstring...
3222
               identifier>
3223
                <btp:inferiors-list> ?
3224
                     <btp:inferior-handle>...hexstring...
3225
3226
                </btp:inferiors-list>
                <btp:qualifiers> ?
3227
3228
                   ...qualifiers...
3229
                </br></btp:qualifiers>
3230
               </br></btp:request_statuses>
3231
```

INFERIOR_STATUSES

```
3234
3235
               <btp:inferior_statuses id?>
3236
                 <btp:target-additional-information>
3237
                    ...additional address information...
3238
                 </btp:target-additional-information>
3239
                 <btp:decider-address>
3240
                    ...address...
3241
                 </br></bbp:decider-address>
3242
                 <btp:transaction-identifier>...hexstring...
3243
               identifier>
3244
                 <br/><btp:status-list>
3245
                       <br/><btp:status-item> +
3246
                          <btp:inferior-handle>...hexstring...
3247
               handle>
3248
                          <btp:status>active|resigned|preparing|prepared|
3249
                               autonomously-confirmed | autonomously-cancelled |
3250
                              confirming|confirmed|cancelling|cancelled|
3251
                              cancel-contradiction|confirm-contradiction|
3252
                              hazard</btp:status>
3253
                          <btp:qualifiers> ?
3254
                               ...qualifiers...
3255
                         </br></btp:qualifiers>
3256
                       </br>
</btp:status-item>
```

REQUEST_STATUS

3264

3282

3283

```
3265
3266
               <btp:request_status id?>
3267
                 <btp:target-additional-information>
3268
                   ...additional address information...
3269
                 </btp:target-additional-information>
3270
                 <btp:reply-address>
3271
                   ...address...
3272
                 </br></btp:reply-address>
3273
                 <btp:inferior-identifier>...hexstring...
3274
               identifier> ?
3275
                 <btp:transaction-identifier>...hexstring...
3276
               identifier> ?
3277
                 <btp:qualifiers> ?
3278
                   ...qualifiers...
3279
                 </br></btp:qualifiers>
3280
               </br></btp:request_status>
3281
```

STATUS

```
3284
3285
               <br/><br/>tp:status id?>
3286
                 <btp:target-additional-information>
3287
                    ...additional address information...
3288
                 </btp:target-additional-information>
3289
                 <btp:inferior-address> ?
3290
                    ...address...
3291
                 </br></ri></ri>
3292
                 <btp:inferior-identifier>...hexstring...
3293
               identifier> ?
3294
                 <btp:decider-address> ?
3295
                    ...address...
3296
                 </br></bbp:decider-address>
3297
                 <btp:transaction-identifier>...hexstring...
3298
               identifier> ?
3299
                 <btp:status-value>created|enrolling|active|resigning|
3300
                         resigned preparing prepared
3301
                          confirming | confirmed | cancelling | cancelled |
3302
                          cancel-contradiction | confirm-contradiction |
3303
                         hazard|contradicted|unknown|inaccessible</btp:status-
3304
               value>
3305
                 <btp:qualifiers> ?
3306
                    ...qualifiers...
3307
                 </br></btp:qualifiers>
3308
               </br></bbp:status>
3309
```

```
3310
3311
          REDIRECT
3312
3313
                <br/><br/>tp:redirect id?>
3314
                  <btp:target-additional-information>
3315
                    ...additional address information...
3316
                  </btp:target-additional-information>
3317
                  <btp:superior-identifier>...hexstring...
3318
                identifier> ?
3319
                  <btp:inferior-identifier>...hexstring...
3320
                identifier>
3321
                  <btp:old-address>
3322
                    ...address...
3323
                  </br></bup:old-address>
3324
                  <br/><btp:new-address>
3325
                    ...address...
3326
                  </br></btp:new-address>
3327
                  <btp:qualifiers> ?
3328
                    ...qualifiers...
3329
                  </br></btp:qualifiers>
3330
                </btp:redirect>
3331
3332
          FAULT
3333
3334
3335
                <br/>
<br/>
tp:fault id?>
3336
                  <btp:target-additional-information>
3337
                    ...additional address information...
3338
                  </btp:target-additional-information>
3339
                  <btp:superior-identifier>...hexstring.../btp:superior-
3340
                identifier> ?
3341
                  <btp:inferior-identifier>...hexstring.../btp:inferior-
3342
                identifier> ?
3343
                  <btp:fault-type>...fault type name...
3344
                  <btp:fault-data>...fault data.../btp:fault-data> ?
3345
                  <btp:qualifiers> ?
3346
                    ...qualifiers...
3347
                  </br></btp:qualifiers>
3348
                </btp:fault>
3349
3350
3351
          The following fault type names are represented by simple strings, corresponding to the entries
3352
          defined in the abstract message set:
3353
3354
                    0
                        general
3355
                        unknown-parameter
                    0
3356
                        wrong-state
                    0
3357
                        communication-failure
                    0
```

invalid-superior

duplicate-inferior

unknown-inferior

0

o

o

3358

3359

3360

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

Fault data can take on various forms:

3368 Free text:

```
<btp:fault-data>...string data...
```

Identifier:

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

Inferior Identity:

Standard qualifiers

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

Transaction timelimit

Inferior timeout

Minimum inferior timeout

3413
3414

c/btpq:minimum-inferior-timeout>

Compounding of Messages

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

```
<btp:messages>
  <btp:enrol>...</btp:enrol>
   <btp:prepared>...</btp:prepared>
</btp:messages>
```

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

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

The carrier protocol binding specifies how a relation between application and BTP messages is represented.

Carrier Protocol Bindings

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

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 elsewhere shall have binding names that are URIs. Bindings specified in this document use numbers to identify the version of the binding, not the version(s) of the carrier protocol.

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

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

Mapping for BTP messages related to application messages: This section will define how BTP messages that are related to application messages are sent. A binding specification may defer details of this to a particular application (e.g. a mapping specification could just say "the CONTEXT may be carried as a parameter of an application invocation"). Alternatively, the binding may specify a general method that represents the relationship between application and BTP messages.

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

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

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

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

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

SOAP Binding

This binding describes how BTP messages will be carried using SOAP as in the <u>SOAP 1.1</u> specification.

Binding name: soap-http-1

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

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

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

3521	mapping for BTP messages related to application messages; All BTP messages sent
3528	with an application message, whether related to the application message or not, shall
3529	be sent in a single btp:messages element in the SOAP:Header. There shall be
3530	precisely one btp:message's element in the SOAP:Header.
3531	precisely one orplinessages element in the 507th fredder.
3532	Implicit massages. A SOAD fault on other communication failure received in response to a
3532 3533	Implicit messages : A SOAP fault, or other communication failure received in response to a SOAP request that had a CONTEXT in the SOAP:Header shall be treated as if a
	•
3534	CONTEXT_REPLY/repudiated had been received. See also the discussion under "other" about the SOAP mustUnderstand attribute.
3535	about the SOAP mustunderstand attribute.
3536	- ·
3537	Faults: A SOAP fault or other communication failure shall be treated as
3538	FAULT/communication-failure.
3539	
3540	Relationship to other bindings: A BTP address for Superior or Inferior that has the binding
3541	string "soap-http-1" is considered to match one that has the binding string "soap-attachments-
3542	http-1" if the binding address and additional information fields match.
3543	
3544	Limitations on BTP use: None
3545	
3546	Other: The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
3547	attribute. The SOAPAction HTTP header is left to be application specific when there are
3548	application messages in the SOAP:Body, as an already existing web service that is being
3549	upgraded to use BTP might have already made use of SOAP Action. The SOAP Action HTTP
3550	header shall be omitted when the SOAP message carries only BTP messages in the
3551	SOAP:Body.
3552	SOAI .bouy.
3553	The SOAP mustUnderstand attribute, when used on the btp:messages containing a the BTP
3554	CONTEXT, ensures that the server (as a whole) determines whether any enrolments are
3555	necessary and reply with CONTEXT_REPLY as appropriate. If mustUnderstand if false, a
3556	server can ignore the CONTEXT (if BTP is not supported there). It is an implementation or
3557	configuration option whether a CONTEXT_REPLY/ok is assumed to be implicit in the HTTP
3558	response in such a case. (If no CONTEXT_REPLY/ok is assumed, it will be impossible for
	*
3559	the business transaction to confirm).
3560	
2561	Note some COAD implementations may not support the myst Industry
3561	Note – some SOAP implementations may not support the mustUnderstand
3562	attribute sufficiently to enforce these requirements. If using such an
3563	implementation on the service side, it may be necessary to assume an
3564	CONTEXT_REPLY/ok.
3565	
	Evample cooperio using COAD hinding
3566	Example scenario using SOAP binding
3567	The annual halous shows an amiliastica manual with CONTEXT annual C
3568	The example below shows an application request with CONTEXT message sent from
3569	client.example.com (which includes the Superior) to services.example.com (Service).
3570	
3571	Annual Properties
3572	<pre><soap:envelope< pre=""></soap:envelope<></pre>

```
3573
                    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3574
                    soap-
3575
                env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
3576
3577
                  <soap:Header>
3578
3579
                    <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
3580
                      <btp:context superior-type="atom">
3581
                        <btp:superior-address>
3582
                           <btp:binding>soap-http-1
3583
                           <br/>btp:binding-
3584
                address>http://client.example.com/soaphandler</btp:binding-
3585
                address>
3586
                           <btp:additional-information>btpengine</btp:additional-</pre>
3587
                information>
3588
                        </br></btp:superior-address>
3589
                        <btp:superior-identifier>1001</btp:superior-identifier>
3590
                        <btp:qualifiers>
3591
                           <btpq:transaction-timelimit</pre>
3592
               xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers">1800</btpq:transact
3593
                ion-timelimit>
3594
                        </br></btp:qualifiers>
3595
                      </br></bbp:context>
3596
                    </br></btp:messages>
3597
3598
                  </soap:Header>
3599
3600
                  <soap:Body>
3601
3602
                    <ns1:orderGoods
3603
                xmlns:ns1="http://example.com/2001/Services/xyzgoods">
3604
                      <custID>ABC8329045</custID>
3605
                      <itemID>224352</itemID>
3606
                      <quantity>5</quantity>
3607
                    </ns1:orderGoods>
3608
3609
                  </soap:Body>
3610
3611
                </soap:Envelope>
3612
3613
```

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

```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap-
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <soap:Header>
    </soap:Header>
```

3614 3615

3616

3617

3618 3619

3620

3621 3622

3623 3624

3625

```
3627
                 <soap:Body>
3628
3629
                   <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
3630
                       <btp:context-reply>
3631
                         <btp:superior-address>
3632
                           <btp:binding>soap-http-1
3633
                           <btp:binding-address>
3634
                              http://client.example.com/soaphandler
3635
                           </br></btp:binding-address>
                           <btp:additional-information>
3636
3637
                              btpengine
3638
                           </br></btp:additional-information>
3639
                        </br></btp:superior-address>
3640
                        <btp:superior-identifier>1001</btp:superior-identifier>
3641
                        <completion-status>related</completion-status>
3642
3643
                        <btp:enrol reply-requested="false">
3644
                          <btp:target-additional-information>
3645
                              btpengine
3646
                          </btp:target-additional-information>
3647
                          <btp:superior-identifier>
3648
                              1001
3649
                          </br></btp:superior-identifier>
3650
                          <btp:inferior-address>
3651
                            <btp:binding>soap-http-1
3652
                            <btp:binding-address>
3653
                              http://services.example.com/soaphandler
3654
                            </btp:binding-address>
3655
                          </br></br></rb>
3656
                          <btp:inferior-identifier>
3657
                              AAAB
3658
                          </br></br></rb>
3659
                         </btp:enrol>
3660
3661
                       </br></btp:context-reply>
3662
3663
                    </br></bbp:messages>
3664
3665
                 </soap:Body>
3666
3667
               </soap:Envelope>
3668
3669
```

SOAP + Attachments Binding

3670 3671

3672 3673

3674

3675

3676

3677 3678 This binding describes how BTP messages will be carried using SOAP as in the <u>SOAP</u> <u>Messages with Attachments</u> specification. It is a superset of the Basic SOAP binding, soaphttp-1. The two bindings only differ when application messages are sent

Binding name: soap-attachments-http-1

BTP message representation: As for soap-http-1

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

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

Implicit messages: As for soap-http-1.

Faults: As for soap-http-1.

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

Limitations on BTP use: None

Other: As for soap-http-1

Example using SOAP + Attachments binding

```
3708
3709
               MIME-Version: 1.0
3710
               Content-Type: Multipart/Related; boundary=MIME boundary;
3711
               type=text/xml;
3712
                       start="someID"
3713
3714
               --MIME_boundary
3715
               Content-Type: text/xml; charset=UTF-8
3716
               Content-ID: someID
3717
3718
               <?xml version='1.0' ?>
3719
               <soap:Envelope
3720
                   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3721
3722
               env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
3723
3724
                 <soap:Header>
3725
3726
                   <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
                     <btp:context superior-type="atom">
3727
```

```
3728
                         <btp:superior-address>
3729
                            <btp:binding>soap-http-1
3730
                            <btp:binding-address>
3731
                                http://client.example.com/soaphandler
3732
                           </br></btp:binding-address>
3733
                         </br></btp:superior-address>
3734
                        <btp:superior-identifier>1001</btp:superior-identifier>
3735
                      </br></bup:context>
3736
                    </br></btp:messages>
3737
3738
                  </soap:Header>
3739
3740
                  <soap:Body>
3741
                    <orderGoods href="cid:anotherID"/>
3742
                  </soap:Body>
3743
3744
               </soap:Envelope>
3745
3746
               --MIME_boundary
3747
               Content-Type: text/xml
3748
               Content-ID: anotherID
3749
3750
                    <ns1:orderGoods
3751
               xmlns:ns1="http://example.com/2001/Services/xyzgoods">
3752
                      <custID>ABC8329045</custID>
3753
                      <itemID>224352</itemID>
3754
                      <quantity>5</quantity>
3755
                    </ns1:orderGoods>
3756
3757
3758
               --MIME_boundary--
3759
```

XML Schema for SOAP Bindings

```
3762
3763
      <?xml version="1.0"?>
3764
      <schema targetNamespace="urn:oasis:names:tc:BTP:xml"</pre>
3765
               xmlns="http://www.w3.org/2001/XMLSchema"
3766
               xmlns:tns="urn:oasis:names:tc:BTP:xml">
3767
3768
           <complexType name="qualifier_type">
3769
               <simpleContent>
3770
                   <extension base="string">
3771
                       <attribute name="must-be-understood" type="boolean"/>
3772
                       <attribute name="to-be-propagated" type="boolean"/>
3773
                   </extension>
3774
               </simpleContent>
3775
           </complexType>
3776
           <element name="qualifier" type="tns:qualifier_type"/>
3777
           <element name="qualifiers">
3778
               <complexType>
3779
                   <sequence>
3780
                        <element ref="tns:qualifier" maxOccurs="unbounded"/>
```

```
3781
                    </sequence>
3782
               </complexType>
3783
           </element>
3784
3785
           <complexType name="address">
3786
               <sequence>
3787
                    <element name="binding-name" type="string"/>
3788
                    <element name="binding-address" type="string"/>
3789
                    <element name="additional-information" type="string"</pre>
3790
      minOccurs="0"/>
3791
               </sequence>
3792
           </complexType>
3793
3794
           <simpleType name="identifier">
3795
             <restriction base="string">
3796
              <pattern value="([0-9,A-Z])*"/>
3797
             </restriction>
3798
           </simpleType>
3799
3800
           <simpleType name="superior-type">
3801
               <restriction base="string">
3802
                    <enumeration value="cohesion"/>
3803
                    <enumeration value="atom"/>
3804
               </restriction>
3805
           </simpleType>
3806
3807
           <simpleType name="transaction-type">
3808
               <restriction base="string">
3809
                    <enumeration value="cohesion"/>
3810
                    <enumeration value="atom"/>
3811
               </restriction>
3812
           </simpleType>
3813
3814
3815
           <element name="context">
3816
               <complexType>
3817
                    <sequence>
3818
                        <element name="superior-address" type="tns:address"</pre>
3819
      maxOccurs="unbounded"/>
3820
                        <element name="superior-identifier"</pre>
3821
       type="tns:identifier"/>
3822
                        <element ref="tns:qualifiers" minOccurs="0"/>
3823
                    </sequence>
3824
                    <attribute name="id" type="ID" use="optional"/>
3825
                    <attribute name="superior-type" type="tns:superior-type"
3826
       use="required"/>
3827
               </complexType>
3828
           </element>
3829
3830
           <element name="context-reply">
3831
               <complexType>
3832
                    <sequence>
3833
                        <element name="superior-address" type="tns:address"</pre>
3834
      maxOccurs="unbounded"/>
```

```
3835
                        <element name="superior-identifier"</pre>
3836
      type="tns:identifier"/>
3837
                        <element name="completion-status">
3838
                             <simpleType>
3839
                                 <restriction base="string">
3840
                                     <enumeration value="completed"/>
3841
                                     <enumeration value="related"/>
3842
                                     <enumeration value="repudiated"/>
3843
                                 </restriction>
3844
                             </simpleType>
3845
                        </element>
3846
                        <element ref="tns:qualifiers" minOccurs="0"/>
3847
                    </sequence>
3848
                    <attribute name="id" type="ID"/>
3849
                    <attribute name="superior-type" type="tns:superior-type"</pre>
3850
      use="required"/>
3851
               </complexType>
3852
           </element>
3853
3854
           <element name="begin">
3855
               <complexType>
3856
                    <sequence>
3857
                        <element name="target-additional-information"</pre>
3858
      type="string"/>
3859
                        <element name="reply-address" type="tns:address"/>
3860
                        <element ref="tns:qualifiers" minOccurs="0"/>
3861
                    </sequence>
3862
                    <attribute name="id" type="ID"/>
3863
                    <attribute name="transaction-type" type="tns:superior-type"</pre>
3864
      use="required"/>
3865
               </complexType>
3866
           </element>
3867
3868
           <element name="begun">
3869
               <complexType>
3870
                    <sequence>
3871
                        <element name="target-additional-information"</pre>
3872
      type="string"/>
3873
                        <element name="decider-address" type="tns:address"</pre>
3874
      minOccurs="0"/>
3875
                        <element name="transaction-identifier"</pre>
3876
      type="tns:identifier" minOccurs="0"/>
3877
                        <element name="inferior-handle" type="tns:identifier"</pre>
3878
      minOccurs="0"/>
3879
                        <element name="inferior-address" type="tns:address"</pre>
3880
      minOccurs="0"/>
3881
                        <element ref="tns:qualifiers" minOccurs="0"/>
3882
                    </sequence>
3883
                    <attribute name="id" type="ID"/>
3884
                    <attribute name="transaction-type" type="tns:superior-type"</pre>
3885
      use="required"/>
3886
               </complexType>
3887
           </element>
3888
```

```
3889
           <element name="enrol">
3890
               <complexType>
3891
                    <sequence>
3892
                         <element name="target-additional-information"</pre>
3893
       type="string"/>
3894
                        <element name="superior-identifier"</pre>
3895
       type="tns:identifier"/>
3896
                        <element name="reply-address" type="tns:address"</pre>
3897
      minOccurs="0"/>
3898
                        <element name="inferior-address" type="tns:address"</pre>
3899
      minOccurs="1" maxOccurs="unbounded"/>
3900
                        <element name="inferior-identifier"</pre>
3901
       type="tns:identifier"/>
3902
                        <element ref="tns:qualifiers" minOccurs="0"/>
3903
                    </sequence>
3904
                    <attribute name="id" type="ID"/>
3905
                    <attribute name="reply-requested" type="boolean"/>
3906
                </complexType>
3907
           </element>
3908
3909
3910
           <element name="enrolled">
3911
               <complexType>
3912
                    <sequence>
3913
                        <element name="target-additional-information"</pre>
3914
      type="string"/>
3915
                        <element name="inferior-identifier"</pre>
3916
      type="tns:identifier"/>
3917
                        <element name="inferior-handle" type="tns:identifier"</pre>
3918
      minOccurs="0"/>
3919
                        <element ref="tns:qualifiers" minOccurs="0"/>
3920
                    </sequence>
3921
                    <attribute name="id" type="ID"/>
3922
               </complexType>
3923
           </element>
3924
3925
           <element name="resign">
3926
               <complexType>
3927
                    <sequence>
3928
                        <element name="target-additional-information"</pre>
3929
       type="string"/>
3930
                        <element name="superior-identifier"</pre>
3931
      type="tns:identifier"/>
3932
                        <element name="inferior-address" type="tns:address"</pre>
3933
      minOccurs="1" maxOccurs="unbounded"/>
3934
                        <element name="inferior-identifier"</pre>
3935
       type="tns:identifier"/>
3936
                        <element ref="tns:qualifiers" minOccurs="0"/>
3937
                    </sequence>
3938
                    <attribute name="id" type="ID"/>
3939
                    <attribute name="response-requested" type="boolean"/>
3940
               </complexType>
3941
           </element>
3942
```

```
3943
           <element name="resigned">
3944
               <complexType>
3945
                    <sequence>
3946
                        <element name="target-additional-information"</pre>
3947
      type="string"/>
3948
                        <element name="inferior-identifier"</pre>
3949
      type="tns:identifier"/>
3950
                        <element ref="tns:qualifiers" minOccurs="0"/>
3951
                    </sequence>
3952
                    <attribute name="id" type="ID"/>
3953
                </complexType>
3954
           </element>
3955
3956
           <element name="prepare">
3957
                <complexType>
3958
                    <sequence>
3959
                        <element name="target-additional-information"</pre>
3960
       type="string"/>
3961
                        <element name="inferior-identifier"</pre>
3962
       type="tns:identifier" minOccurs="0"/>
3963
                        <element name="reply-address" type="tns:address"</pre>
3964
      minOccurs="0"/>
3965
                        <element name="transaction-identifier"</pre>
3966
       type="tns:identifier" minOccurs="0"/>
3967
                        <element name="inferiors-list" minOccurs="0">
3968
                             <complexType>
3969
                                 <sequence>
3970
                                     <element name="inferior-handle"</pre>
3971
      type="tns:identifier" maxOccurs="unbounded"/>
3972
                                 </sequence>
3973
                             </complexType>
3974
                        </element>
3975
                        <element ref="tns:qualifiers" minOccurs="0"/>
3976
                    </sequence>
3977
                    <attribute name="id" type="ID"/>
3978
               </complexType>
3979
           </element>
3980
3981
           <element name="prepared">
3982
               <complexType>
3983
                    <sequence>
3984
                        <element name="target-additional-information"</pre>
3985
       type="string"/>
3986
                        <element name="superior-identifier"</pre>
3987
      type="tns:identifier"/>
3988
                        <element name="inferior-address" type="tns:address"</pre>
3989
      maxOccurs="unbounded"/>
3990
                        <element name="inferior-identifier"</pre>
3991
      type="tns:identifier"/>
3992
                        <element ref="tns:qualifiers" minOccurs="0"/>
3993
                    </sequence>
3994
                    <attribute name="id" type="ID"/>
3995
                    <attribute name="default-is-cancel" type="boolean"/>
3996
               </complexType>
```

```
3997
           </element>
3998
3999
           <element name="confirm">
4000
               <complexType>
4001
                    <sequence>
4002
                         <element name="target-additional-information"</pre>
4003
       type="string"/>
4004
                        <element name="inferior-identifier"</pre>
4005
       type="tns:identifier"/>
4006
                        <element ref="tns:qualifiers" minOccurs="0"/>
4007
                    </sequence>
4008
                    <attribute name="id" type="ID"/>
4009
                </complexType>
4010
           </element>
4011
4012
           <element name="confirmed">
4013
                <complexType>
4014
                    <sequence>
4015
                        <element name="target-additional-information"</pre>
4016
       type="string"/>
4017
                        <element name="superior-identifier"</pre>
4018
       type="tns:identifier"/>
4019
                        <element name="inferior-address" type="tns:address"</pre>
4020
      minOccurs="0"/>
4021
                        <element name="inferior-identifier"</pre>
4022
       type="tns:identifier" minOccurs="0"/>
4023
                        <element name="decider-address" type="tns:address"</pre>
4024
      minOccurs="0"/>
4025
                        <element name="transaction-identifier"</pre>
4026
       type="tns:identifier" minOccurs="0"/>
4027
                        <element ref="tns:qualifiers" minOccurs="0"/>
4028
                    </sequence>
4029
                    <attribute name="id" type="ID"/>
4030
                    <attribute name="confirmed-received" type="boolean"/>
4031
                </complexType>
4032
           </element>
4033
4034
           <element name="cancel">
4035
               <complexType>
4036
                    <sequence>
4037
                        <element name="target-additional-information"</pre>
4038
       type="string"/>
4039
                        <element name="inferior-identifier"</pre>
4040
       type="tns:identifier" minOccurs="0"/>
4041
                        <element name="reply-address" type="tns:address"</pre>
4042
      minOccurs="0"/>
4043
                        <element name="transaction-identifier"</pre>
4044
       type="tns:identifier" minOccurs="0"/>
4045
                        <element name="decider-address" type="tns:address"</pre>
4046
      minOccurs="0"/>
4047
                        <element name="transaction-identifier"</pre>
4048
       type="tns:identifier" minOccurs="0"/>
4049
                        <element name="inferiors-list" minOccurs="0">
4050
                             <complexType>
```

```
4051
                                 <sequence>
4052
                                      <element name="inferior-handle"</pre>
4053
       type="tns:identifier" maxOccurs="unbounded"/>
4054
                                 </sequence>
4055
                             </complexType>
4056
                        </element>
4057
                        <element ref="tns:qualifiers" minOccurs="0"/>
4058
4059
                    <attribute name="id" type="ID"/>
4060
                </complexType>
4061
           </element>
4062
4063
           <element name="cancelled">
4064
               <complexType>
4065
                    <sequence>
4066
                        <element name="target-additional-information"</pre>
4067
       type="string"/>
4068
                        <element name="superior-identifier"</pre>
4069
       type="tns:identifier"/>
4070
                        <element name="inferior-address" type="tns:address"</pre>
4071
      maxOccurs="unbounded"/>
4072
                        <element name="inferior-identifier"</pre>
4073
       type="tns:identifier" minOccurs="0"/>
4074
                        <element name="decider-address" type="tns:address"</pre>
4075
      minOccurs="0"/>
4076
                        <element name="transaction-identifier"</pre>
4077
       type="tns:identifier" minOccurs="0"/>
4078
                        <element ref="tns:qualifiers" minOccurs="0"/>
4079
                    </sequence>
4080
                    <attribute name="id" type="ID"/>
4081
                </complexType>
4082
           </element>
4083
4084
           <element name="hazard">
4085
               <complexType>
4086
                    <sequence>
4087
                        <element name="target-additional-information"</pre>
4088
       type="string"/>
4089
                        <element name="superior-identifier"</pre>
4090
       type="tns:identifier"/>
4091
                        <element name="inferior-address" type="tns:address"</pre>
4092
      maxOccurs="unbounded"/>
4093
                        <element name="inferior-identifier"</pre>
4094
       type="tns:identifier"/>
4095
                        <element ref="tns:qualifiers" minOccurs="0"/>
4096
                    </sequence>
4097
                    <attribute name="id" type="ID"/>
4098
               </complexType>
4099
           </element>
4100
4101
           <element name="contradiction">
4102
               <complexType>
4103
                    <sequence>
```

```
4104
                        <element name="target-additional-information"</pre>
4105
       type="string"/>
4106
                        <element name="inferior-identifier"</pre>
4107
       type="tns:identifier"/>
4108
                        <element ref="tns:qualifiers" minOccurs="0"/>
4109
                    </sequence>
4110
                    <attribute name="id" type="ID"/>
4111
               </complexType>
4112
           </element>
4113
4114
           <element name="superior-state">
4115
               <complexType>
4116
                    <sequence>
4117
                        <element name="target-additional-information"</pre>
4118
       type="string"/>
4119
                        <element name="inferior-identifier"</pre>
4120
       type="tns:identifier"/>
4121
                        <element name="status">
4122
                            <simpleType>
4123
                                 <restriction base="string">
4124
                                     <enumeration value="active"/>
4125
                                     <enumeration value="prepared-received"/>
4126
                                     <enumeration value="inaccessible"/>
4127
                                     <enumeration value="unknown"/>
4128
                                 </restriction>
4129
                            </simpleType>
4130
                        </element>
4131
                        <element ref="tns:qualifiers" minOccurs="0"/>
4132
                    </sequence>
4133
                    <attribute name="id" type="ID"/>
4134
                    <attribute name="reply-requested" type="boolean"/>
4135
               </complexType>
4136
           </element>
4137
4138
           <element name="inferior-state">
4139
               <complexType>
4140
                    <sequence>
4141
                        <element name="target-additional-information"</pre>
4142
      type="string"/>
4143
                        <element name="superior-identifier"</pre>
4144
       type="tns:identifier"/>
4145
                        <element name="inferior-address" type="tns:address"</pre>
4146
      maxOccurs="unbounded"/>
4147
                        <element name="inferior-identifier"</pre>
4148
       type="tns:identifier"/>
4149
                        <element name="status">
4150
                            <simpleType>
4151
                                 <restriction base="string">
4152
                                     <enumeration value="active"/>
4153
                                     <enumeration value="prepared-received"/>
4154
                                     <enumeration value="inaccessible"/>
4155
                                     <enumeration value="unknown"/>
4156
                                 </restriction>
4157
                            </simpleType>
```

```
4158
                        </element>
4159
                        <element ref="tns:qualifiers" minOccurs="0"/>
4160
                   </sequence>
4161
                   <attribute name="id" type="ID"/>
4162
                   <attribute name="reply-requested" type="boolean"/>
4163
               </complexType>
4164
           </element>
4165
4166
           <element name="confirm-one-phase">
4167
               <complexType>
4168
                   <sequence>
4169
                        <element name="target-additional-information"</pre>
4170
      type="string"/>
4171
                        <element name="inferior-identifier"</pre>
4172
      type="tns:identifier"/>
4173
                        <element ref="tns:qualifiers" minOccurs="0"/>
4174
                    </sequence>
4175
                   <attribute name="id" type="ID"/>
4176
                   <attribute name="report-hazard" type="boolean"/>
4177
               </complexType>
4178
           </element>
4179
4180
           <element name="request-confirm">
4181
               <complexType>
4182
                   <sequence>
4183
                        <element name="target-additional-information"</pre>
4184
      type="string"/>
4185
                        <element name="reply-address" type="tns:address"/>
4186
                        <element name="transaction-identifier"</pre>
4187
      type="tns:identifier"/>
4188
                        <element name="inferiors-list" minOccurs="0">
4189
                            <complexType>
4190
                                <sequence>
4191
                                     <element name="inferior-handle"</pre>
4192
      type="tns:identifier" maxOccurs="unbounded"/>
4193
                                </sequence>
4194
                            </complexType>
4195
                        </element>
4196
                        <element ref="tns:qualifiers" minOccurs="0"/>
4197
                    </sequence>
4198
                   <attribute name="id" type="ID"/>
4199
                   <attribute name="report-hazard" type="boolean"/>
4200
               </complexType>
4201
           </element>
4202
4203
           <element name="request-statuses">
4204
               <complexType>
4205
                   <sequence>
4206
                        <element name="target-additional-information"</pre>
4207
      type="string"/>
4208
                        <element name="reply-address" type="tns:address"/>
4209
                        <element name="transaction-identifier"</pre>
4210
      type="tns:identifier"/>
4211
                        <element name="inferiors-list" minOccurs="0">
```

```
4212
                            <complexType>
4213
                                <sequence>
4214
                                     <element name="inferior-handle"</pre>
4215
      type="tns:identifier" maxOccurs="unbounded"/>
4216
                                 </sequence>
4217
                            </complexType>
4218
                        </element>
4219
                        <element ref="tns:qualifiers" minOccurs="0"/>
4220
                    </sequence>
4221
                    <attribute name="id" type="ID"/>
4222
               </complexType>
4223
           </element>
4224
4225
           <element name="inferior-statuses">
4226
               <complexType>
4227
                    <sequence>
4228
                        <element name="target-additional-information"</pre>
4229
       type="string"/>
4230
                        <element name="decider-address" type="tns:address"/>
4231
                        <element name="transaction-identifier"</pre>
4232
       type="tns:identifier"/>
4233
                        <element name="status-list">
4234
                          <complexType>
4235
                            <sequence>
4236
                               <element name="status-item" maxOccurs="unbounded">
4237
                            <complexType>
4238
                               <sequence>
4239
                                 <element name="inferior-handle"</pre>
4240
      type="tns:identifier"/>
4241
                                 <element name="status">
4242
                                   <simpleType>
4243
                                 <restriction base="string">
4244
                                     <enumeration value="active"/>
4245
                                     <enumeration value="resigned"/>
4246
                                     <enumeration value="preparing"/>
4247
                                     <enumeration value="prepared"/>
4248
                                     <enumeration value="autonomously-</pre>
4249
      confirmed"/>
4250
                                     <enumeration value="autonomously-</pre>
4251
      cancelled"/>
4252
                                     <enumeration value="confirming"/>
4253
                                     <enumeration value="confirmed"/>
4254
                                     <enumeration value="cancelling"/>
4255
                                     <enumeration value="cancelled"/>
4256
                                     <enumeration value="cancel-contradiction"/>
4257
                                     <enumeration value="confirm-contradiction"/>
4258
                                     <enumeration value="hazard"/>
4259
                                 </restriction>
4260
                                   </simpleType>
4261
4262
                                 <element ref="tns:qualifiers" minOccurs="0"/>
4263
                               </sequence>
4264
                            </complexType>
4265
                               </element>
```

```
4266
                            </sequence>
4267
                          </complexType>
4268
                        </element>
4269
                        <element ref="tns:qualifiers" minOccurs="0"/>
4270
                    </sequence>
4271
                    <attribute name="id" type="ID"/>
4272
               </complexType>
4273
           </element>
4274
4275
           <element name="request-status">
4276
               <complexType>
4277
                    <sequence>
4278
                        <element name="target-additional-information"</pre>
4279
      type="string"/>
4280
                        <element name="reply-address" type="tns:address"/>
4281
                        <element name="inferior-identifier"</pre>
4282
      type="tns:identifier" minOccurs="0"/>
4283
                        <element name="transaction-identifier"</pre>
4284
      type="tns:identifier" minOccurs="0"/>
4285
                        <element ref="tns:qualifiers" minOccurs="0"/>
4286
                    </sequence>
4287
                    <attribute name="id" type="ID"/>
4288
               </complexType>
4289
           </element>
4290
4291
           <element name="status">
4292
               <complexType>
4293
                    <sequence>
4294
                        <element name="target-additional-information"</pre>
4295
      type="string"/>
4296
                        <element name="inferior-address" type="tns:address"</pre>
4297
      minOccurs="0"/>
4298
                        <element name="inferior-identifier"</pre>
4299
      type="tns:identifier" minOccurs="0"/>
4300
                        <element name="decider-address" type="tns:address"</pre>
      minOccurs="0"/>
4301
4302
                        <element name="transaction-identifier"</pre>
4303
      type="tns:identifier" minOccurs="0"/>
4304
                        <element name="status-value">
4305
                              <simpleType>
4306
                            <restriction base="string">
4307
                                 <enumeration value="created"/>
4308
                                 <enumeration value="enrolling"/>
4309
                                 <enumeration value="active"/>
4310
                                 <enumeration value="resigning"/>
4311
                                 <enumeration value="resigned"/>
4312
                                 <enumeration value="preparing"/>
4313
                                 <enumeration value="prepared"/>
4314
                                 <enumeration value="confirming"/>
4315
                                 <enumeration value="confirmed"/>
4316
                                 <enumeration value="cancelling"/>
4317
                                 <enumeration value="cancelled"/>
4318
                                 <enumeration value="cancel-contradiction"/>
4319
                                 <enumeration value="confirm-contradiction"/>
```

```
4320
                                 <enumeration value="hazard"/>
4321
                                 <enumeration value="contradicted"/>
4322
                                 <enumeration value="unknown"/>
4323
                                 <enumeration value="inaccessible"/>
4324
                             </restriction>
4325
                               </simpleType>
4326
                        </element>
4327
                        <element ref="tns:qualifiers" minOccurs="0"/>
4328
                    </sequence>
4329
                    <attribute name="id" type="ID"/>
4330
               </complexType>
4331
           </element>
4332
4333
           <element name="redirect">
4334
               <complexType>
4335
                    <sequence>
4336
                        <element name="target-additional-information"</pre>
4337
       type="string"/>
4338
                        <element name="superior-identifier"</pre>
4339
       type="tns:identifier" minOccurs="0"/>
4340
                        <element name="inferior-identifier"</pre>
4341
       type="tns:identifier"/>
4342
                        <element name="old-address" type="tns:address"</pre>
4343
      maxOccurs="unbounded"/>
4344
                        <element name="new-address" type="tns:address"</pre>
4345
      maxOccurs="unbounded"/>
4346
                        <element ref="tns:qualifiers" minOccurs="0"/>
4347
                    </sequence>
4348
                    <attribute name="id" type="ID"/>
4349
               </complexType>
4350
           </element>
4351
4352
           <element name="fault">
4353
               <complexType>
4354
                    <sequence>
4355
                        <element name="target-additional-information"</pre>
4356
      type="string"/>
4357
                        <element name="superior-identifier"</pre>
4358
       type="tns:identifier" minOccurs="0"/>
4359
                        <element name="inferior-identifier"</pre>
4360
       type="tns:identifier" minOccurs="0"/>
4361
                        <element name="fault-type" type="string"/>
4362
                        <element name="fault-data" type="anyType"</pre>
4363
      minOccurs="0"/>
4364
                        <element ref="tns:qualifiers" minOccurs="0"/>
4365
                    </sequence>
4366
                    <attribute name="id" type="ID"/>
4367
               </complexType>
4368
           </element>
4369
4370
       </schema>
4371
```

4371 4372 Conformance 4373 4374 4375 A BTP implementation need not implement all aspects of the protocol to be useful. The level 4376 of conformance of an implementation is defined by which roles it can support using the 4377 specified messages and carrier protocol bindings for interoperation with other 4378 implementations. 4379 4380 A partially conformant implementation may implement some roles in a non-interoperable way, giving that implementation's users comparable proprietary functionality. 4381 4382 4383 The following Roles and Role Groups are used to define conformance:

4384

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

4386
4387
4388
4389

An implementation may support one or more Role Groups. The following combinations are defined as commonly expected conformance profiles, although other combinations or selections are equally possible.

Conformance Profile	Role Groups
Participant Only	Participant
Atomic	Atomic Superior Participant
Cohesive	Full Superior Participant
Atomic Coordination Hub	Initiator/Terminator Atomic Coordination Hub Participant
Cohesive Coordination Hub	Initiator/Terminator Cohesive Coordination Hub Participant
TP has several features, such as optional parameters, that allow alternative implementation chitectures. Implementations should pay particular attention to avoid assuming their pee	

BTP has several features, such as optional parameters, that allow alternative implementation architectures. Implementations should pay particular attention to avoid assuming their peers have made the same implementation options as they have (e.g. an implementation that always sends ENROL with the same inferior address and with the reply address absent (because the Inferior in all transactions are dealt with by the same addressable entity), must not assume that the same is true of received ENROLs)

Part 3. Appendices

4399 4400 4401

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

4402

4403 4404

A. Glossary

A datum which is produced and then consumed. Message

The producer of a message. Sender

The consumer of a message. Receiver

Transmission The passage of a message from a sender to a

receiver.

Endpoint A sender or receiver.

Address An identifier for an endpoint.

A protocol which defines how transmissions **Carrier Protocol**

occur.

Carrier Protocol The address of an endpoint for a particular carrier

Address protocol.

(CPA)

Business Transaction A compound address consisting of a mandatory **Protocol Address** carrier protocol address and an optional opaque

suffix. (BTPA)

PRF - suffix ? I've used "additional

information"

An entity which executes procedures, a software Actor

agent.

An actor which uses the Business Transaction **Application**

Protocol.

A message produced by an application and **Application Message**

consumed by an application.

An endpoint of an application message. **Application Endpoint**

A procedure which is started by a receiver when a **Operation**

message arrives at it.

Application Operation An operation which is started when an application

message arrives.

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.

Appropriate In accordance with a pertinent contract.

Inappropriate In violation of a pertinent contract.

Service An actor, which on receipt of an application

messages, may start an appropriate application operation. For example, a process which advertises an interface allowing defined RPCs to

be invoked by a remote client.

Client An actor which sends application messages to

services.

Effect The changes induced by the incomplete or

complete processing of a set of procedures by an

actor, which are observable by another

contemporary or future actor, and which are made in conformance with a contract known to any such observer. This contract must state the countereffect of the effect, and is known as the countereffect contract. An effect is **Completed** when the change-inducing processing of the set of procedures is finished. [Need an indirect or consequential damage exclusion clause]

PRF - Sentence about countereffect contract doesn't fit well

Ineffectual Describes a set of procedures which has no effect.

Countereffect An appropriate effect intended to counteract a

prior effect.

Countereffect Contract The contract which governs the relationship

between the effect and the countereffect of a procedure. In the absence of any other overriding contracts the countereffect contract is the promise

that

"The **Countereffect** will attempt so far as is possible to reverse or cancel the **Effect** such that an observer (on completion of the **Countereffect**)

is unaware that the **Effect** ever occurred, but this attempt cannot be guaranteed to succeed".

Cancel Process a countereffect for the current effect of a

set of procedures.

Confirm Ensure that the effect of a set of procedures is

completed.

Prepare Ensure that of a set of procedures is capable of

being successfully instructed to cancel or to

confirm.

Outcome A decision to either cancel or confirm.

Participant A set of procedures which is capable of receiving

instructions from a coordinator to prepare, cancel and confirm. A participant must also have a BTPA to which these instructions will be delivered, in the form of BTP messages. A participant is identified by a participant identifier.

Inferior Identifier An identifier assigned to an Inferior which is

unique within the scope of an Address-as-Inferior.

Atomic Business A set of participants (which may have only one member), all of which will receive instructions

that will result in a homogeneous outcome.

(Transitively, a set of operations, whose effect is capable of countereffect.) An atom is identified

by an atom identifier.

Atom Identifier A globally unique identifier assigned to an atom.

PRF – abs msgs define as unambiguous in scope of its address-as-superior, I

think.

Coordinator An actor which decides the outcome of a single

atom, and has a lifetime which is coincident with

that of the atom. A coordinator can issue

instructions to a participant to prepare, cancel and confirm. These instructions take the form of BTP messages. A coordinator is identified by its atom's atom identifier. A coordinator must also have a BTPA to which participants can send BTP

messages.

Address-as-Superior The address used to communicate with an actor

playing the role of an Superior

or

playing the role of an Superior

The address used to communicate with a Address-as-Composer

Composer by an application actor that controls its resolution. The messages that might be sent to or

received from this endpoint are undefined.

Address-as-Inferior The address used to communicate with an actor

playing the role of an Inferior.

The combination of Superior Identifier and **Identity-as-Superior**

Address-as-Superior of a given Superior.

The combination of Inferior Identifier and **Identity-as-Inferior**

Address-as-Inferior of a given Inferior.

4405