



C. PCT 920
– 07.2

June 16, 2003

Madam,
Sir,

Modifications of the Administrative Instructions Under the PCT

*Standard for the electronic filing and processing
of international applications: Modifications of Annex F, including Appendix I*

1. This Circular concerns modifications of Annex F of the Administrative Instructions under the Patent Cooperation Treaty (PCT), including Appendix I thereof, to be promulgated with effect from June 19, 2003.
2. Part 7 and Annex F of the Administrative Instructions containing, respectively, the legal framework and the technical standard for electronic filing and processing of international applications entered into force on January 7, 2002 (see document PCT/AI/1 Rev.1 Add.2 dated December 20, 2001, and *PCT Gazette* Special Issue No. S-04/2001 dated December 27, 2001). Annex F was subsequently the subject of modifications that entered into force on December 12, 2002 (see document PCT/AI/1 Rev.1 Add.5 dated November 25, 2002, and *PCT Gazette* No. 50/2002 dated December 12, 2002).
3. The present modifications are based on proposals for change to Annex F that were received or prepared by the International Bureau in 2002. Those proposals are available on WIPO's Web site (see http://pcteasy.wipo.int/efiling_standards/EFPage.htm). A number of changes have been made to the proposals taking into account comments received after consultation, under Rule 89.2(b) of the Regulations under the PCT, with the national and regional patent Offices of, or acting for, all PCT Contracting States, with PCT International Authorities and with interested intergovernmental organizations, as well as with certain non-governmental organizations representing users of the PCT system (see Circular C. PCT 885, dated November 25, 2002).

/...

4. Following that consultation, Annex F is modified as follows:

- (i) section 3.1.1.1.1 – new section “Paragraph numbering in XML documents (description)”;
- (ii) section 3.2 – referenced document added to Figure 2;
- (iii) section 4.2.1 – changes concerning the description of signers;
- (iv) section 4.2.3 – new section “Compound WASP (C-WASP)”;
- (v) section 4.3 – new section “Recommended file naming convention”;
- (vi) section 5.1.3 – new section “Application layer protocol for notification” (including new Figures 6, 10, 11, and 12);
- (vii) sections 5.1.4 to 5.1.9 – some changes and additions to the E-filing interoperability protocol;
- (viii) sections 5.2.1 to 5.2.3 – some changes and additions to the package/transmission combinations.

5. In addition, Appendix I of Annex F (“XML DTDs for the E-PCT Standard”) is modified as follows:

- (i) section 3.9 – new section “Table”;
- (ii) section 3.10 – new section “*Ex officio* correction”;
- (iii) section 4.1 – modifications to “Package header”;
- (iv) sections 4.3 and 4.4 – new sections “Dispatch list” and “Receipt list”;
- (v) section 5 – new section “Other E-PCT DTDs,” including:
 - section 5.1 “Demand form,”
 - section 5.2 “Received information on the demand form by IPEA,”
 - section 5.3 “Chapter II fee sheet,”
 - section 5.4 “Priority document bibliographic data,”
 - section 5.5 “Priority document body,”
 - section 5.6 “IB bibliographic data,”
 - section 5.7 “IB publication,” and
 - section 5.8 “Filing of amendment, including any statement, under Article 19 and Article 34.2(b)” (former section 3.5);

/...

(vi) minor corrections and additions to existing DTDs.

6. The text of the modifications of the main body of Annex F and of Appendix I thereto is set out, respectively, in document PCT/AI/1 Rev.1 Add.7, enclosed herewith, and document PCT/AI/1 Rev.1 Add.8 (which is available, together with document PCT/AI/1 Rev.1 Add.7, on WIPO's Web site at <http://www.wipo.int/pct/en/texts/index.htm>). Paper copies of document PCT/AI/1 Rev.1 Add.8 are available from the International Bureau upon request.

7. The modifications will be promulgated in *PCT Gazette* No. 25/2003 on June 19, 2003.

Sincerely yours,



Francis Gurry
Assistant Director General

Enclosure: document PCT/AI/1 Rev.1 Add.7

WIPO



PCT/AI/1 Rev.1 Add.7

ORIGINAL: English

DATE: May 28, 2003

E

WORLD INTELLECTUAL PROPERTY ORGANIZATION

GENEVA

PATENT COOPERATION TREATY (PCT)

ADMINISTRATIVE INSTRUCTIONS UNDER THE PATENT COOPERATION TREATY: MODIFICATIONS OF ANNEX F

with effect from June 19, 2003

1. This document contains the text of modifications, with effect from June 19, 2003, of the Administrative Instructions Under the Patent Cooperation Treaty (PCT), as in force from January 1, 2003 (see documents PCT/AI/1 Rev.1 dated August 23, 2001, PCT/AI/1 Rev.1 Add.1 dated October 26, 2001, PCT/AI/1 Rev.1 Add.2 dated December 20, 2001, PCT/AI/1 Rev.1 Add.3 dated September 2, 2002, PCT/AI/1 Rev.1 Add.4 dated October 14, 2002, PCT/AI/1 Rev.1 Add.5 dated November 25, 2002, and PCT/AI/1 Rev.1 Add.6 dated December 10, 2002; see also further modifications with effect from June 19, 2003, set out in document PCT/AI/1 Rev.1 Add.8 dated May 28, 2003, which is published on WIPO's Web site at <http://www.wipo.int/pct/en/texts/index.htm>; paper copies of this document are available from the International Bureau of WIPO upon request). The modifications contained in the present document, which are promulgated after consultation with the interested Offices and Authorities pursuant to Rule 89.2(b) of the Regulations under the PCT, involve modifications and additions of a number of provisions of the main body of Annex F of the Administrative Instructions.
2. The text of the present modifications will be published in *PCT Gazette* No. 25/2003 on June 19, 2003.

MODIFICATIONS OF THE ADMINISTRATIVE INSTRUCTIONS
(with effect from June 19, 2003)

ANNEX F
STANDARD FOR THE ELECTRONIC FILING AND PROCESSING
OF INTERNATIONAL APPLICATIONS

TABLE OF CONTENTS

1. and 2. [No change]
3. E-PCT submission structure and format
 - 3.1 Allowable electronic document formats
 - 3.1.1 Character coded formats
 - 3.1.1.1 XML
 - 3.1.1.1.1 Paragraph Numbering in XML documents (description) [New]
 - 3.1.1.2 and 3.1.1.3 [No change]
 - 3.1.2 and 3.1.3 [No change]
 - 3.2 E-PCT document and submission structure
 - 3.3 and 3.4 [No change]
 4. IA documents packaging
 - 4.1 [No change]
 - 4.2 PKI package types
 - 4.2.1 Wrapped and signed package (WASP)
 - 4.2.2 [No change]
 - 4.2.3 Compound WASP (C-WASP) [New]
 - 4.3 Recommended file naming convention [New]
 - 4.3.1 Tables
 - 4.3.2 Applicant's identifier
 - 4.3.3 Examples
 5. Transmission
 - 5.1 The E-filing interoperability protocol
 - 5.1.1 Principles
 - 5.1.2 Application layer protocol for application
 - 5.1.2.1 Use of the SSL tunnel for application
 - 5.1.2.2 Application level events for application
 - 5.1.3 Application layer protocol for notification [New]
 - 5.1.3.1 Use of the SSL tunnel for notification
 - 5.1.3.2 Application level events for notification
 - 5.1.4 Transaction management header elements
 - 5.1.5 Transaction management data elements
 - 5.1.6 Server parameters
 - 5.1.7 Client parameters
 - 5.1.8 Division mechanism
 - 5.1.9 Event level protocol
 - 5.1.9.1 Begin transaction
 - 5.1.9.2 Send package header
 - 5.1.9.3 Send package data
 - 5.1.9.4 Get receipt

- 5.1.9.5 End transaction
 - 5.1.9.6 Get package header for notification
 - 5.1.9.7 Get package data for notification
 - 5.1.9.8 Send receipt check notice for notification
 - 5.1.9.9 Get package header for dispatch list
 - 5.1.9.10 Get package data for dispatch list
 - 5.1.9.11 Send receipt check notice for dispatch list
 - 5.1.9.12 Get Package header for application receipt list
 - 5.1.9.13 Get package data for application receipt list
 - 5.1.9.14 Send receipt check notice for application receipt list
- 5.2 Package/transmission combinations
- 5.2.1 Applicant-Office (international phase) sector
 - 5.2.2 Office-Office sector
 - 5.2.3 Designated Office sector
6. to 9. [No change]

1. and 2. [No change]

3. E-PCT SUBMISSION STRUCTURE AND FORMAT

[No change to the introductory text]

3.1 *Allowable electronic document formats*

[No change to the introductory text]

3.1.1 *Character coded formats*

3.1.1.1 *XML*

[No change to the introductory text]

3.1.1.1.1 *Paragraph Numbering in XML documents (description) [New]*

If the description part of an international application is encoded in XML format, the paragraphs of that description part shall be numbered by a four-digit Arabic number, with leading zeros where required, for example, [0099], enclosed in square brackets and placed to the right of the left margin of the document.

If the number of paragraphs exceeds four digits, then the numbering of paragraphs should increase by one digit, and so forth, according to need. For example, paragraph [10000] follows paragraph [9999] and paragraph [100000] follows paragraph [99999].

3.1.1.2 and 3.1.1.3 [No change]

3.1.2 and 3.1.3 [No change]

3.2 *E-PCT document and submission structure*

[No change to the introductory text]

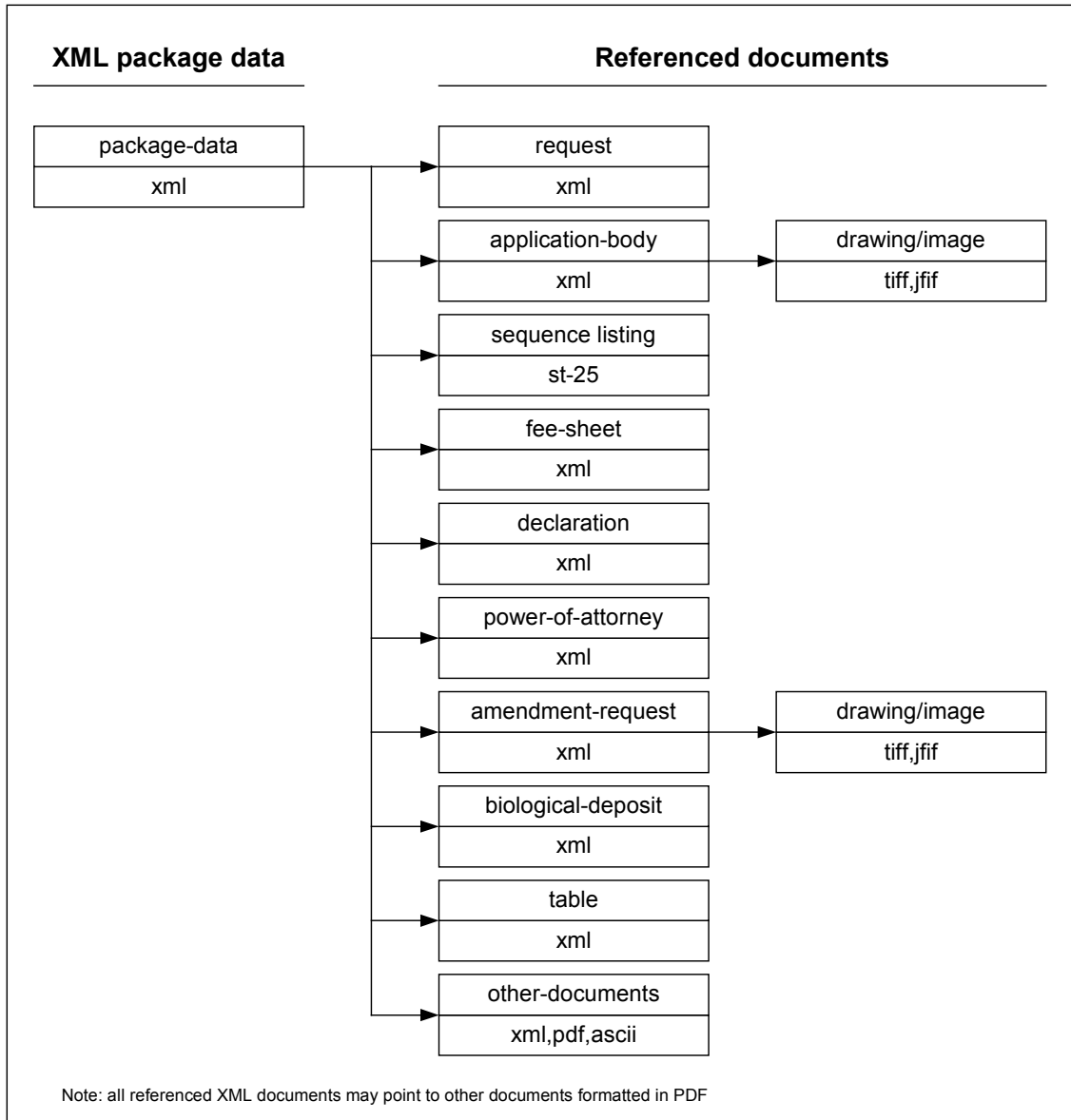


Figure 2 – Example E-PCT submission document structure

3.3 and 3.4 [No change]

4. IA DOCUMENTS PACKAGING

[No change to the introductory text]

4.1 [No change]

4.2 PKI package types

[No change to the introductory text]

4.2.1 *Wrapped and signed package (WASP)*

When a person who signs the WASP is the applicant (or his representative), the signature of the WASP may also serve as an enhanced electronic signature of the application (see section 3.3) if technical systems in place provide that the application is automatically signed thereby.

A low-level or high-level digital certificate (see definitions in section 9) accompanies the digital signature.

Figure 3 is a simplified anatomy of the WASP. The diagram has been intentionally simplified to obscure technical detail that may distract the reader from the key issues of the package design. For example, the PKZIP wrapping has been left out of the diagram.

In case of a notification sent by the Office to the applicant, the Office prepares, signs and sends the WASP which contains such notification.

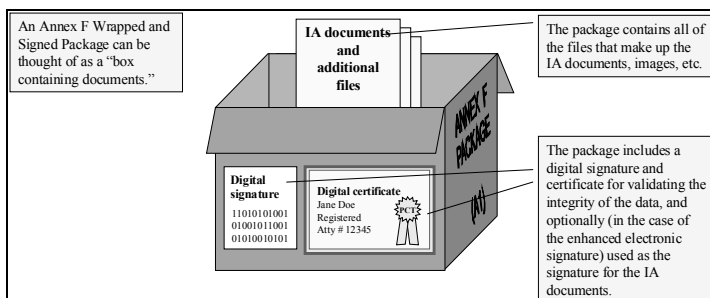


Figure 3 – Wrapped and signed package (WASP)

See Appendix II for additional detail on the WASP technical specification.

4.2.2 [No change]

4.2.3 *Compound WASP (C-WASP)* [New]

The one or more WASPs sent to the applicant from the Office are wrapped using the ZIP as shown in the section 4.1.1 and treated as one data block. This data block is called "Compound WASP" (C-WASP).

4.3 Recommended file naming convention [New]

An electronic filing of a patent application will have multiple files associated with it. Filing name conventions need to be established in order to enhance server automation, as well as to establish a client side software workflow and a good work practice for user understanding. The following set of tables constitutes the recommended file naming convention and the client side software should automatically produce the suffixes and extensions accordingly. Each of these tables addresses a level of the standard with a final table of examples to follow.

4.3.1 Tables

Table 1

<i>Codes used in the descriptions below</i>	
A	One character from the following set: {ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz}
A...	Any combination of at least two characters from the following set: {ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789}
AAA	Any combination of one to three characters from the following set: {ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789}
NNNNNN	Any combination of six characters from the following set: {0123456789}

Table 2

<i>Each instance of a document type</i>		
A...	Applicant's identifier, not to exceed 50 positions	Recommended
-	Separator (dash)	
A...	Document type (<i>see Table 6</i>)	
.	Separator (period)	
AAA	File type (<i>see Table 5</i>)	

Table 3

<i>External entities referenced from within document instances</i>		
A...	Applicant's identifier, not to exceed 50 positions	Recommended
-	Separator (dash)	
A...	Document type (<i>see Table 6</i>)	
-	Separator (dash)	
A	Entity type (<i>see Table 8</i>)	
NNNNNN	Entity sequence number, right-justified, left-padded with zero	Optional
-	Separator (dash)	
NNNNNN	Page sequence number, right-justified, left-padded with zero	
.	Separator (period)	Recommended
AAA	File type (<i>see Table 5</i>)	

Table 4

<i>Files not referenced from within document instances</i>		
A...	Applicant's identifier, not to exceed 50 positions	Recommended
-	Separator (dash)	
A...	Document name as provided by applicant, not to exceed 50 positions	
.	Separator (period)	
AAA	File type (<i>see Table 5</i>)	

Table 5

<i>File name extensions accepted</i>	
txt	Text file, see section 3.1.1.3.
xml	see section 3.1.1.1.
tif	TIFF, see section 3.1.3.1.
jpg	JFIF, see section 3.1.3.2.
zip	Archive file containing one or more files that might or might not be compressed.
app	ST.25 , see section 3.1.1.2.
pdf	Portable document format, see section 3.1.2.

Table 6

<i>Document types currently accepted for initial ePCT filing</i>	
<i>Document type</i>	<i>Code</i>
amendment-request	amnd
application-body	appb
bio-deposit	biod
declaration	decl
package-data	pkda
fee-sheet	fees
power-of-attorney	poat
priority-doc	pdoc
request	requ
ro-request-receiving-info	rrri
xmit-receipt	xmre
pkgheader	pkgh
Office-specific document types	[2-position country code]AA
ST.25	seq1
Table exceeding fifty printed pages	mtbl

Table 7

<i>Subdocument types currently accepted for initial ePCT filing</i>	
<i>Subdocument type</i>	<i>Code</i>
description	desc
claims	clam
abstract	abst
drawings	draw

Table 8

<i>Entity types</i>	
T	Table
M	Mathematical formula
C	Chemical structure or formula
S	Sequence listing
D	Drawing page (contains one or more figures per image page and one or more image pages)
F	Figure (exactly one figure on exactly one image page)
I	Embedded image (one or more image pages)
P	Document page

4.3.2 Applicant's identifier

The applicant's identifier is determined by the applicant with or without the help of the filing tool. The name of every file that is part of a submission will begin with the same applicant's identifier. Applicant's identifier might be a name or a docket number or some other string that has significance to the applicant. An applicant's identifier is not necessarily unique to each submission, that is, it might be used for another submission associated with prosecution of the same application; it could even be used by the applicant for all submissions for all his applications. The applicant's identifier is placed first so that in a directory listing, all the files for a particular submission or application or applicant will sort together.

4.3.3 Examples

<i>File</i>	<i>Contents</i>
dupont003340-appb.xml	Application
dupont003340-appb-C000001.CDX	First chemical structure, ChemDraw format
dupont003340-appb-C000001.MOL	First chemical structure, MOL format
dupont003340-appb-C000001.TIF	First chemical structure, TIFF image format
dupont003340-pkda.xml	Package Data
dupont003340-fees.xml	Fee sheet
dupont003340-poat.xml	Power of attorney
dupont003340-requ.xml	Request
dupont003340-appb-T000001.TIF	First table, TIFF format
dupont003340-appb-T000002-000001.TIF	Second table, first page, TIFF format
dupont003340-appb-T000002-000002.TIF	Second table, second page, TIFF format
dupont003355-appb.xml	Application
dupont003355-appb-D000001.TIF	First drawing page, TIFF format
dupont003355-pkda.xml	Package Data
dupont003355-fees.xml	Fee sheet
dupont003355-appb-M000001.TIF	First mathematical formula, TIFF format
dupont003355-appb-M000002.TIF	Second mathematical formula, TIFF format
dupont003355-requ.xml	Request
dupont003355-appb-T000001.TIF	First table, TIFF format

5. TRANSMISSION

[No change to the introductory text]

5.1 *The E-filing interoperability protocol*

[No change to the introductory text]

5.1.1 *Principles*

[No change to the introductory text]

5.1.2 *Application layer protocol for application*

At the highest level for application, there are five events that the protocol requires a client and server to support. These events are:

- (a) Begin Transaction
- (b) Send Package Header
- (c) Send Package Data
- (d) Get Receipt
- (e) End Transaction

In between the Begin and End Transactions, there are three types of WASP sent between the client and the server:

- (i) The package header contains essential information for initial processing to identify the submission. It is a WASP containing the package header in XML format.
- (ii) The package data contains the information for submitting application. It is a WASP consisting of various types of files.
- (iii) The receipt is an acknowledgment of the submission. The content of this receipt (XML data plus an optional human readable certificate in PDF or TIFF), which is signed by the receiving Office, is defined in Annex F Appendix I. The date of receipt will be determined according to the usual principles applicable to the filing of applications on paper, including filing by electronic means (such as by facsimile transmission), that is, based on the date prevailing at the location of the Office at the time when the complete transmission of the application has been received.

5.1.2.1 Use of the SSL tunnel for application

These events are all performed within an SSL tunnel that is established before issuing the Begin Transaction event. The SSL tunnel is built using both client and server authentication. The SSL tunnel may be stopped at the end of the transaction or, if a batch of transmissions is foreseen, the SSL tunnel can be left open and only stopped when all transmissions are complete. The SSL tunnel uses the SSL protocol version 3.0.

5.1.2.2 Application level events for application

Start SSL session (See Figure 5)

Step 0: Begin Transaction

Client action:

Get transaction Information.

Server response:

Return values in the *transaction_id* and *max_division_size* transaction management header elements.

transaction_id is a unique identifier assigned by the server associating all transactions involved in the submission of an application.

max_division_size is the maximum number of bytes permitted by the server for the size of a division.

Step 1: Send Package Header

Client action:

Send package header

Server response:

- a) OK
- b) Error (Abort, go back to step 0)
- c) Package already received; go to step 3 to get the receipt.

After receiving the last division of the WASP containing the package header, the server must verify the signature of the WASP. If the signature is invalid (for instance expired), the Application Response Code (ARC) will remain OK, but the server will capture the error and provide a message on the receipt.

Step 2: Send Package Data

Client action:

Send package data

Server response:

- a) OK
- b) Error (Abort, go back to step 0)

After receiving the last division of the WASP containing the package data, the server must verify the signature of the WASP and compare the message digest of the unsigned package against the message digest provided in the Package Header in Step 1 of the transaction before returning the ARC to the client. If both conditions are met, the server should return an ARC indicating OK. If the hash values in package header and the WAD of the package data do not match, the ARC value should be set to FFF7. If the signature is invalid (for instance expired), the ARC will remain OK, but the server will capture the error and provide a message on the receipt.

Step 3: Request Receipt

Client action:

Send request

Server response:

- a) OK (Receipt object included in response)
- b) Error (Abort, go back to step 0)

Step 4: End Transaction.

Client action:

Send acknowledgment of completion including information about any client problem to the server.

Server response:

- a) OK
- b) Error (Client can ignore this response)

Close SSL session

In all cases of SSL Tunnel, the current protocol requires each individual transaction to be acknowledged by an individual receipt.

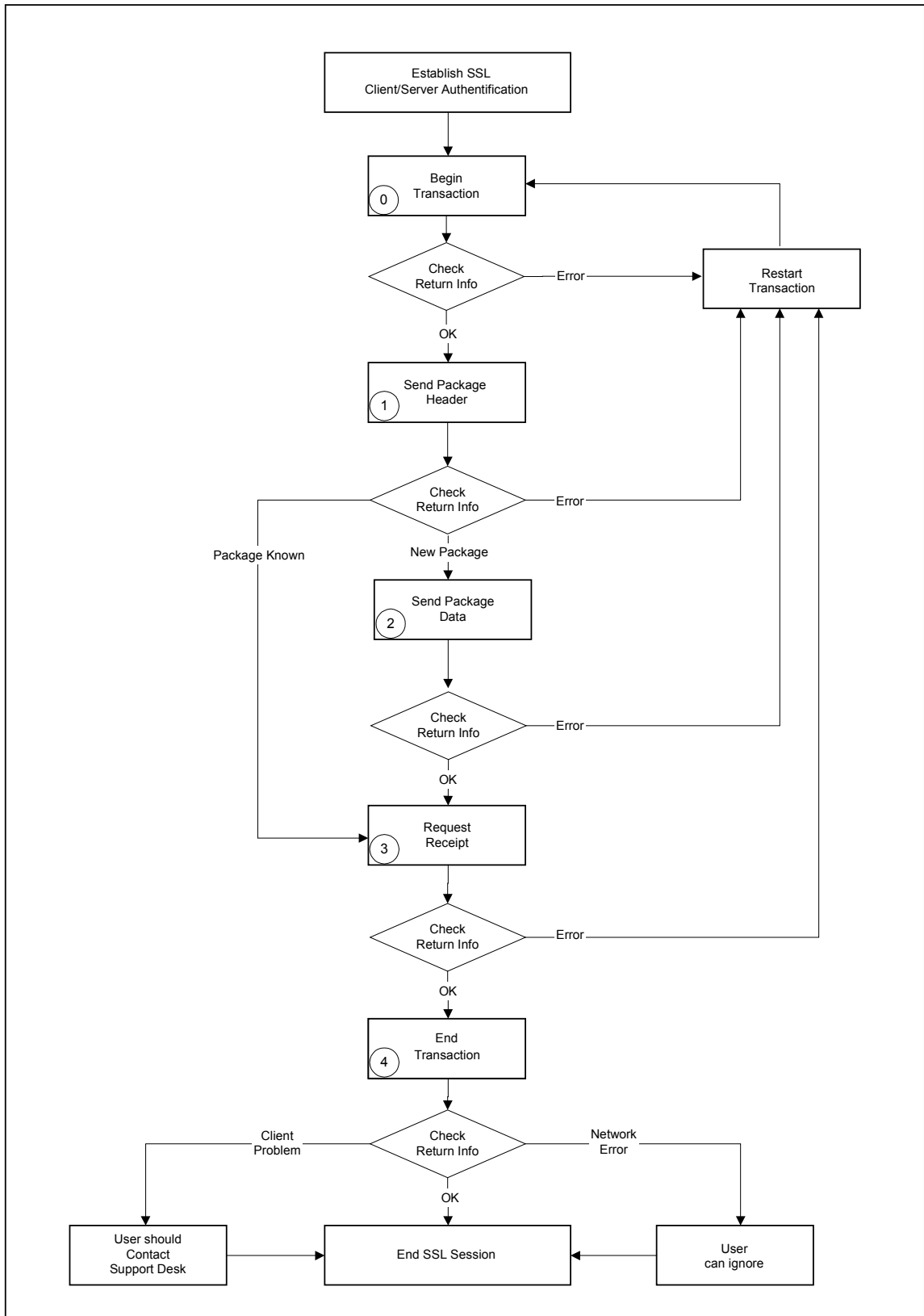


Figure 5 – Application level protocol behavior for application

5.1.3 *Application layer protocol for notification* [New]

At the highest level for notification, there are five events that the protocol requires a client and server to support¹⁰. These events are:¹¹

- (a) Begin Transaction
- (b) Get Package Header (for notification, or dispatch list, or application receipt list)¹²
- (c) Get Package Data (for notification, or dispatch list, or application receipt list)¹²
- (d) Send Receipt Check Notice (for notification, or dispatch list, or application receipt list)¹²
- (e) End Transaction

In between the Begin and End Transactions, there are two types of WASP and one type of C-WASP sent between the client and the server:

- (i) The client action package header contains essential information for initial processing to identify the request for notification. It is a WASP containing the package header in XML format. (This is applied to request to a server from a client.)
- (ii) The server response package header contains summary information (such as a dispatch-number and the number of notifications to be sent) on the notification to be notified. It is a WASP containing the package header in XML format. (This is applied to response to a client from server.)
- (iii) The package data contains the dispatched notification information. It is a C-WASP that consists of one or more WASP(s).

5.1.3.1 *Use of the SSL tunnel for notification*

Refer to Section 5.1.2.1, "Use of the SSL tunnel for application."

¹⁰ The Office may inform the applicant of the existence of notifications before these five events, by other means of communication, such as e-mail.

¹¹ This protocol may be used to transmit the dispatch list, the application receipt list, and the notification. Transmission of the dispatch list, the application receipt list, and the notification is supported at the discretion of the Office. The dispatch list contains dispatch numbers corresponding to notifications that the Office has sent to the applicant. The application receipt list contains application numbers corresponding to application documents that the Office has received from the applicant.

¹² The server uses the value of the "transaction-type" attribute (see section 5.1.4) to identify the type of document requested, e.g. notification, dispatch list, application receipt list.

5.1.3.2 Application level events for notification

Start SSL session (See Figure 6)

Step 0: Begin Transaction

Client action:

Get transaction Information.

Server response:

Return values in the *transaction_id* and *max_division_size* transaction management header elements.

transaction_id is a unique identifier assigned by the server associating all transactions involved in sending a notification.

max_division_size is the maximum number of bytes permitted by the server for the size of a division.

Step 1: Get Package Header

Client action:

Send request for package header (The WASP of package header for request of notification is contained.)

Server response:

- a) OK (The response contains the WASP of package header containing summary information (such as dispatch-number, number-of-notification) of notifications.)¹³
- b) Error (Abort, go back to step 0)

After receiving the last division of the WASP containing the package header, the server must verify the signature of the WASP. If the signature is invalid (for instance, due to a signature verification error or validation data expiration), the application response code (ARC) value is set to FFF6.

If the number of sendable notifications in package header of Server response is “0(zero)” (there is no sendable notifications), then go to Step 4.

Step 2: Get Package Data

Client action:

Send request for Package data

Server response:

- a) OK (The response contains the C-WASP consisted of one or more WASP(s))
- b) Error (Abort, go back to step 0)

¹³ If the C-WASP contains multiple WASP, the notice-info of each notification is set in the package header.

Step 3: Send Receipt Check Notice

Client action:

Send Receipt Check Notice

Server response:

- a) OK
- b) Error (Abort, go back to step 0)

Step 4: End Transaction.

Client action:

Send acknowledgment of completion including information about any client problem to the server.

Server response:

- a) OK
- b) Error (Client can ignore this response)

Close SSL session

In all cases of SSL Tunnel, the current protocol requires that, for each transaction, the client acknowledge the reception by sending Receipt Check Notice to the server.

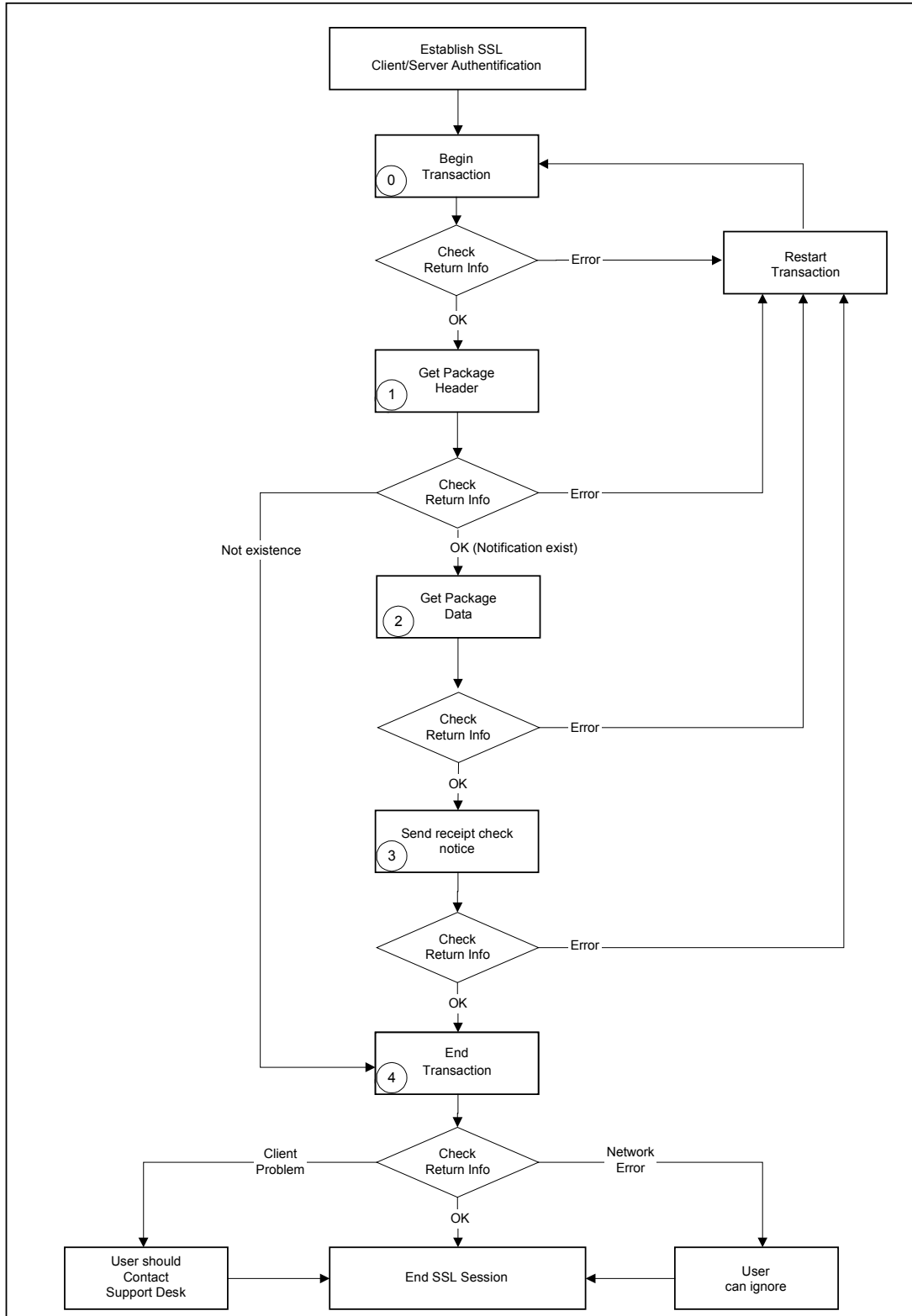


Figure 6 – Application level protocol behavior for notification [New]

5.1.4 Transaction management header elements

The following items, which are all fixed length, are included in all post and response messages. Unused parameters of header elements are set to space (ASCII '20').

Attrib. Name	division_hash
Values	ASCII upper case Hexadecimal representation of 160-bit hash value
Length	40 bytes (40 x 8bit characters)
Description	SHA-1 Hash of the current division.

Attrib. Name	protocol_version
Values	Unique
Length	4 bytes (4 x 8bit ASCII char)
Description	A unique identifier for the version of the protocol used to create the transaction data (e.g. 0100 for Version 1.0) First two bytes are for the major version number and last two for the release within this version.

Attrib. Name	transaction_type																												
Values	<table border="1"> <tr> <td>pbeg, ebeg,</td> <td></td> </tr> <tr> <td>pend, eend</td> <td></td> </tr> <tr> <td>ehdr, phdr,</td> <td></td> </tr> <tr> <td>edat, pdat,</td> <td></td> </tr> <tr> <td>erct, prct,</td> <td></td> </tr> <tr> <td>ephn, pphn</td> <td>Get package header for notification</td> </tr> <tr> <td>epdn, ppdn</td> <td>Get package data for notification</td> </tr> <tr> <td>ercn, prcn</td> <td>Send receipt check notice for notification</td> </tr> <tr> <td>ephd, pphd</td> <td>Get package header for dispatch list</td> </tr> <tr> <td>epdd, ppdd</td> <td>Get package data for dispatch list</td> </tr> <tr> <td>ercd, prcd</td> <td>Send receipt check notice for dispatch list</td> </tr> <tr> <td>epha, ppha</td> <td>Get package header for application receipt list</td> </tr> <tr> <td>epda, ppda</td> <td>Get package data for application receipt list</td> </tr> <tr> <td>erca, prca</td> <td>Send receipt check notice for application receipt list</td> </tr> </table> <p>ASCII lower case 7-bit ISO 646 e-stands for encrypted, p-plain text</p>	pbeg, ebeg,		pend, eend		ehdr, phdr,		edat, pdat,		erct, prct,		ephn, pphn	Get package header for notification	epdn, ppdn	Get package data for notification	ercn, prcn	Send receipt check notice for notification	ephd, pphd	Get package header for dispatch list	epdd, ppdd	Get package data for dispatch list	ercd, prcd	Send receipt check notice for dispatch list	epha, ppha	Get package header for application receipt list	epda, ppda	Get package data for application receipt list	erca, prca	Send receipt check notice for application receipt list
pbeg, ebeg,																													
pend, eend																													
ehdr, phdr,																													
edat, pdat,																													
erct, prct,																													
ephn, pphn	Get package header for notification																												
epdn, ppdn	Get package data for notification																												
ercn, prcn	Send receipt check notice for notification																												
ephd, pphd	Get package header for dispatch list																												
epdd, ppdd	Get package data for dispatch list																												
ercd, prcd	Send receipt check notice for dispatch list																												
epha, ppha	Get package header for application receipt list																												
epda, ppda	Get package data for application receipt list																												
erca, prca	Send receipt check notice for application receipt list																												
Length	4 bytes																												
Description	Attrib. of the transaction header that identifies the nature of the data transmitted. The value beginning with letter d or z is not available.																												

Note that the value beginning with the letter d or z is reserved for domestic application or the other transmission.

Attrib. Name	transaction_id
Values	Unique
Length	36 bytes
Description	A unique identifier assigned by the server associating all transactions involved in the submission of an application. For Begin Transaction this is blank (ASCII x'20').

Attrib. Name	reserved_use
Values	Reserved for domestic use (e.g. Server date and time YYYYMMDDHHMMSS)
Length	32 bytes
Description	This data area is available for the option by each RO. (e.g. To inform a client of the machine time of the RO server).

Attrib. Name	total_bytes
Values	Numeric ASCII with left-hand zero padding (e.g. 0000000123456789)
Length	16 bytes (16 x 8bit chars)
Description	The total size in bytes of the object being sent (WASP containing the Package Header, WASP containing the package data, and the WASP containing the receipt).

Attrib. Name	division_size
Values	Numeric ASCII with left-hand zero padding (e.g. 0000000123456789)
Length	16 bytes (16 x 8bit chars)
Description	The size in bytes of the data component (chunk) of the object being transferred.

Attrib. Name	division_offset
Values	Numeric ASCII with left-hand zero padding (e.g. 0000000123456789)
Length	16 bytes (16 x 8bit chars)
Description	Value representing the starting position of the data within the object being transferred. Division_offset starts at 0.

Attrib. Name	division_response_code		
Values	<i>Division RCs</i>		<i>Meaning</i>
	0000		OK
	FFFF		General Error
	FFFE		Resend Last
	FFFD		Wait
	FFFC		Protocol Sequence Error
	ASCII 4 x 8bit char		
Length	4 bytes		
Description	Server or client return code used to manage the division mechanism		

Attrib. Name	application_response_code		
Values		<i>Application RCs</i>	<i>Meaning</i>
		0000	OK
		FFFF	General Error
		0001	OK, Package Known
		0002	OK, New Package
		0003	OK, Not Existence
		1000	Pending
		FFFB	Client Problem
		FFFA	Network Error
		FFF9	Protocol Version Error
		FFF8	Hash Value of “division hash” in the Transaction Management Header is erroneous.
		FFF7	The hash values in package header and the WAD of package data do not match.
		FFF6	The signature is invalid (for instance, due to a signature verification error or validation data expiration). ¹⁴
	ASCII 4 x 8bit char		
Length	4 bytes		
Description	Server or client return code used to manage the application level events		

Attrib. Name	Encoding_method		
Values		<i>Application RCs</i>	<i>Meaning</i>
		UTF8	UNICODE UTF8
		SJIS	UNICODE Shift-JIS
		KS X	UNICODE KS X 1001
	ASCII 4 x 8bit char		
Length	4 bytes		
Description	Encoding scheme for error message translation.		

Attrib. Name	error_message
Values	UNICODE UTF8, UNICODE Shift-JIS, UNICODE KS X 1001
Length	256 bytes (256 x 8bits)
Description	Optional text explaining the reason for error response codes. If an error message is needed for both division and application response codes, these should be concatenated. Each server will choose one of the specified encoding schemes to translate the error message into human readable format.

¹⁴ This code is applied when the server cannot verify the authentication in Get package header.

5.1.5 Transaction management data elements

Attrib. Name	max_division_size
Values	Numeric ASCII with left-hand zero padding
Length	16 bytes (16 x 8bit chars)
Description	Maximum bytes allowed for a division.
Example	00000000000008192 (8Kbytes)

5.1.6 Server parameters

Attrib. Name	Server_timeout
Values	Numeric ASCII with left-hand zero padding (e.g. 0000000123456789)
Length	16 bytes (16 x 8bit chars)
Description	Time in seconds before the server can assume that a client has lost its network connection and the transaction can be abandoned.
Example	0000000000000120 (2 minutes)

Note that the value for the server_timeout at the protocol level is set at the discretion of the individual Office.

5.1.7 Client parameters

Attrib. Name	Client_preferred_division_size
Values	Numeric ASCII with left-hand zero padding
Length	16 bytes (16 x 8bit chars)
Description	Preferred number of bytes to be used for a division.
Example	0000000000004096 (8k)

Attrib. Name	Client_retry_limit
Values	Numeric ASCII with left-hand zero padding
Length	16 bytes (16 x 8bit chars)
Description	Number of times the client should resend a division before abandoning the transaction
Example	0000000000000005 (5 retries)

Note that the maximum number of Attrib. Client_retry_limit is NN (16 times). When a server retries more than 16 times, the transmission may be terminated.

Attrib. Name	Client_retry_wait
Values	Numeric ASCII with left-hand zero padding (e.g. 0000000123456789)
Length	16 bytes (16 x 8bit chars)
Description	The time in seconds the client should wait before issuing a retry
Example	0000000000000005 (5 secs)

Note that the value for the client_retry_wait at the protocol level is set at the application level.

5.1.8 Division mechanism

When sending data between the client and server, this data is divided into manageable chunks which, together with a transaction management header, are called divisions. Under the control of the client, the size of these divisions can vary during the life of the transactions. This provides a pacing mechanism that can be used to overcome Internet transmission problems.

The initial size of the division data message is set to the smallest of either:

- (a) max_division_size returned by the server as a response to the Begin Transaction Request
- (b) client_preferred_division_size set in the startup parameters of the client

The client builds one or more divisions made up of the transmission management header and a data message. As each division is sent in the divided order to the server, the server checks for completeness of the transmission by calculating the hash value of the division.

5.1.8.1 Calculating the division hash value

The hash is calculated on the basis of all fields in the header as well as any data message. The hash, which is calculated using the SHA-1 algorithm, is placed as the first element of each division.

Before the server rejects a package as invalid, it should check the version of the protocol before checking the hash value in case a future version of the protocol should adopt a different hash algorithm.

The following fields of the HTTP Post or response message are therefore included in the hash calculation:

Name	Protocol Version	Transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application rc	Encoding method	error message	data message
Length	4	4	36	32	16	16	16	4	4	4	256	???

5.1.9 Event level protocol

Transactions described in this section are further illustrated in Figure 7 to 12 below.

5.1.9.1 *Begin transaction*

The Begin Transaction post message submitted by the client contains the highest protocol version supported by the client. If the server supports the version provided by the client, it should communicate with the client in accordance with the rules for that version of the protocol and use that version number in all response messages. If the server cannot support the protocol version specified by the client, the application response code should indicate protocol version error, and the version number specified in the response message should be the highest protocol version supported by the server. The client should support earlier versions.

Post Message

Name	Division hash	Protocol Version	Transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	pbeg	Blank	???	0	0	0	0	0	???	blank	None

Response Message

Name	division hash	Protocol Version	transaction type	Transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	16
Value	X	0100	pbeg	New id	???	16	16	0	0	0	???	???	???

Data Message: max_division_size (16 bytes)

5.1.9.2 *Send package header*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	???
Value	X	0100	phdr	tranid	???	X	Y	Z	0	0	???	blank	pkghdr

Data Message: WASP containing package header

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	phdr	tranid	???	0	0	0	a	b	???	blank	None

5.1.9.3 *Send package data*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	???
Value	X	0100	pdat	trandid	???	x	y	z	0	0	???	blank	pkgdata

Data Message: WASP containing package data

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	pdat	trandid	???	0	0	0	a	b	???	blank	None

5.1.9.4 *Get receipt*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	prct	trandid	???	0	0	0	???	0	???	blank	None

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	???
Value	X	0100	prct	trandid	???	x	y	z	???	0	???	blank	Receipt

Data Message: WASP containing receipt

5.1.9.5 *End transaction*

Post Message

Name	division hash	Protocol Version	Transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	pend	trandid	???	0	0	0	0	0	???	blank	None

Response Message

Name	division hash	Protocol Version	Transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	pend	trandid	???	0	0	0	a	b	???	???	None

5.1.9.6 *Get package header for notification*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	???
Value	X	0100	pphn	tranid	???	X	Y	Z	a	b	???	blank	pkghdr

Data Message: WASP containing package header

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	pphn	tranid	???	X	Y	Z	a	b	???	blank	pkghdr

Data Message: WASP containing package header

5.1.9.7 *Get package data for notification*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	ppdn	tranid	???	0	0	0	a	b	???	blank	None

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	???
Value	X	0100	ppdn	tranid	???	x	y	Z	0	0	???	blank	pkgdata

Data Message: C-WASP containing WASP

5.1.9.8 *Send receipt check notice for notification*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	prcn	tranid	???	0	0	0	0	0	???	blank	None

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	prcn	tranid	???	0	0	0	a	b	???	blank	None

5.1.9.9 *Get package header for dispatch list*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	???
Value	X	0100	pphd	trandid	???	X	Y	Z	a	b	???	blank	pkghdr

Data Message: WASP containing package header

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	pphd	trandid	???	X	Y	Z	a	b	???	blank	pkghdr

Data Message: WASP containing package header

5.1.9.10 *Get package data for dispatch list*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	ppdd	trandid	???	0	0	0	a	b	???	blank	None

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	???
Value	X	0100	ppdd	trandid	???	x	y	z	0	0	???	blank	pkgdata

Data Message: C-WASP containing WASP

5.1.9.11 *Send receipt check notice for dispatch list*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	prcd	trandid	???	0	0	0	0	0	???	blank	None

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	prcd	trandid	???	0	0	0	a	b	???	blank	None

5.1.9.12 *Get Package header for application receipt list*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	???
Value	X	0100	ppha	tranid	???	X	Y	Z	a	b	???	blank	pkghdr

Data Message: WASP containing package header

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	ppha	tranid	???	X	Y	Z	a	b	???	blank	pkghdr

Data Message: WASP containing package header

5.1.9.13 *Get package data for application receipt list*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	ppda	tranid	???	0	0	0	a	b	???	blank	None

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	???
Value	X	0100	ppda	tranid	???	x	y	z	0	0	???	blank	pkgdata

Data Message: C-WASP containing WASP

5.1.9.14 *Send receipt check notice for application receipt list*

Post Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	prca	tranid	???	0	0	0	0	0	???	blank	None

Response Message

Name	division hash	Protocol Version	transaction type	transaction id	Reserved use	total bytes	division size	division offset	division RC	application RC	Encoding method	error message	data message
Length	40	4	4	36	32	16	16	16	4	4	4	256	0
Value	X	0100	prca	tranid	???	0	0	0	a	b	???	blank	None

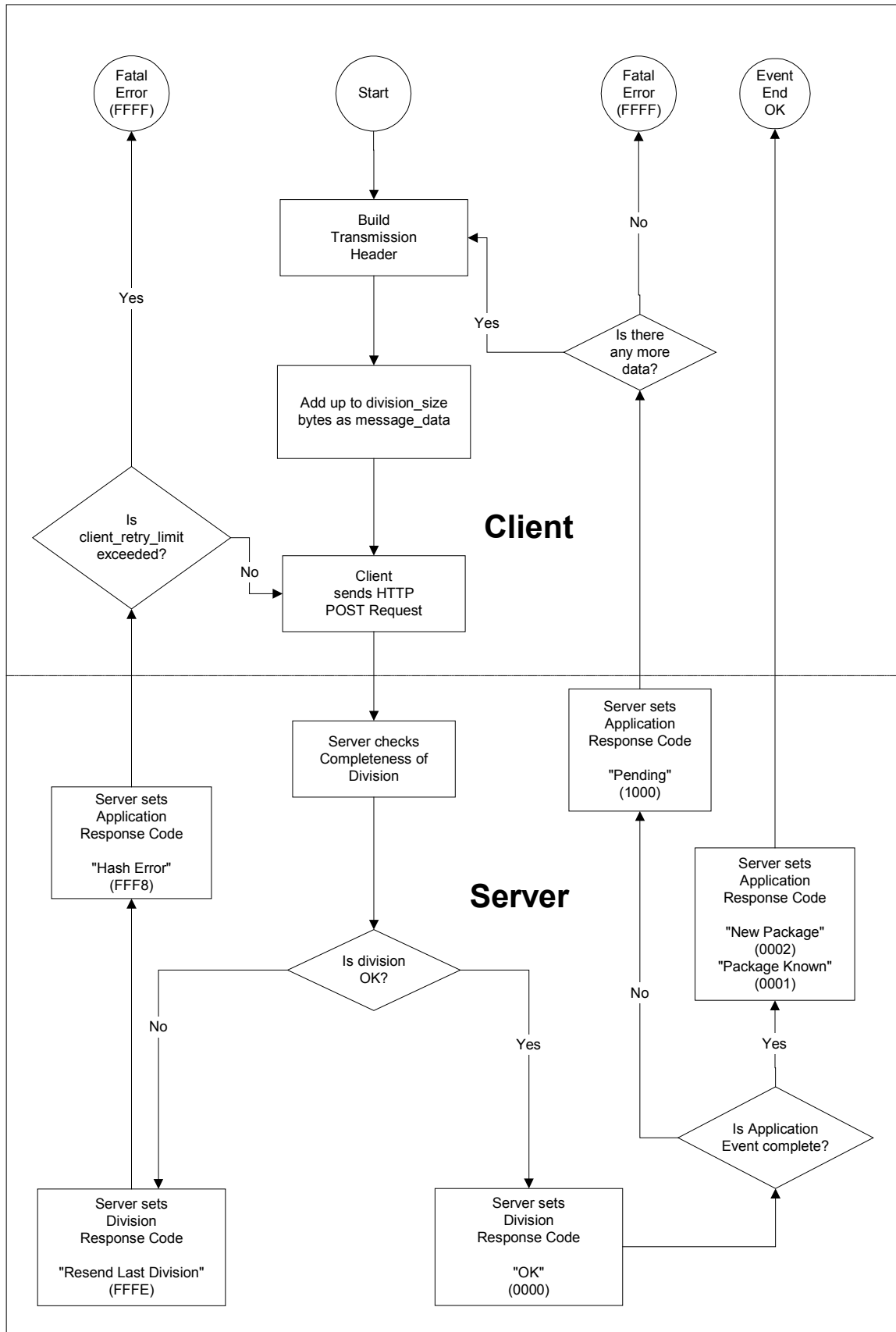


Figure 7 – Send package header behavior

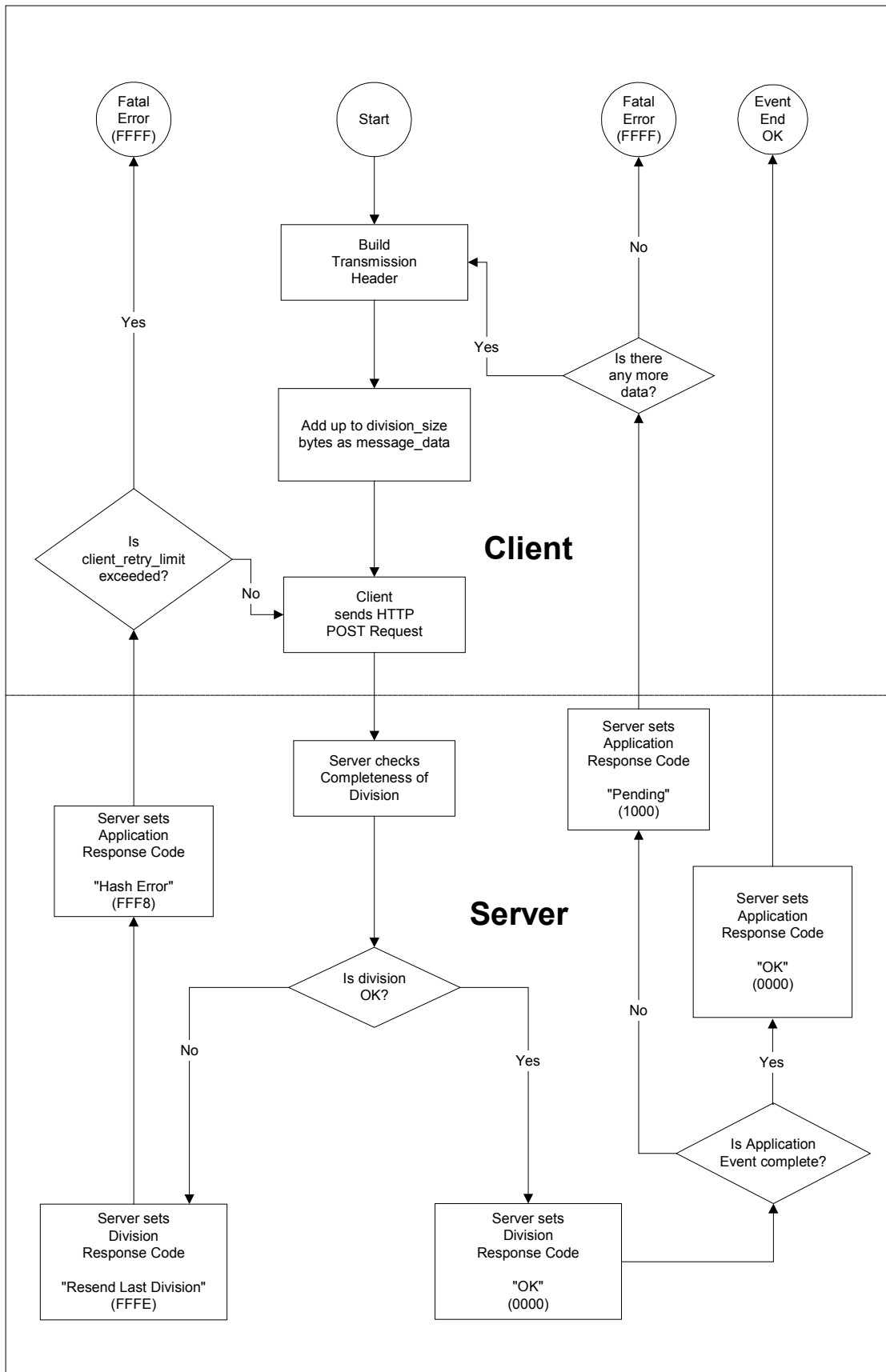


Figure 8 – Send package data behavior

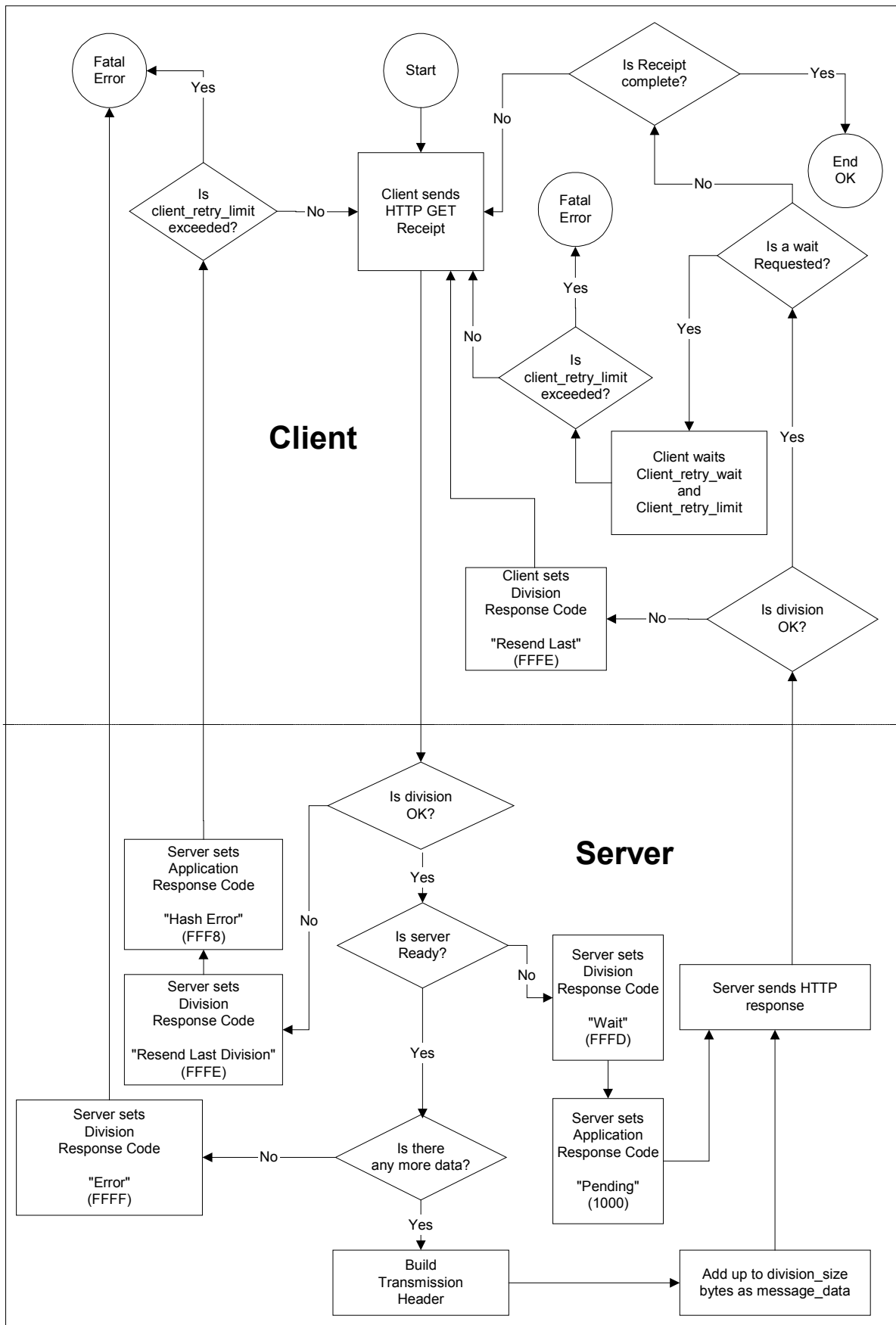


Figure 9 – Get receipt behavior

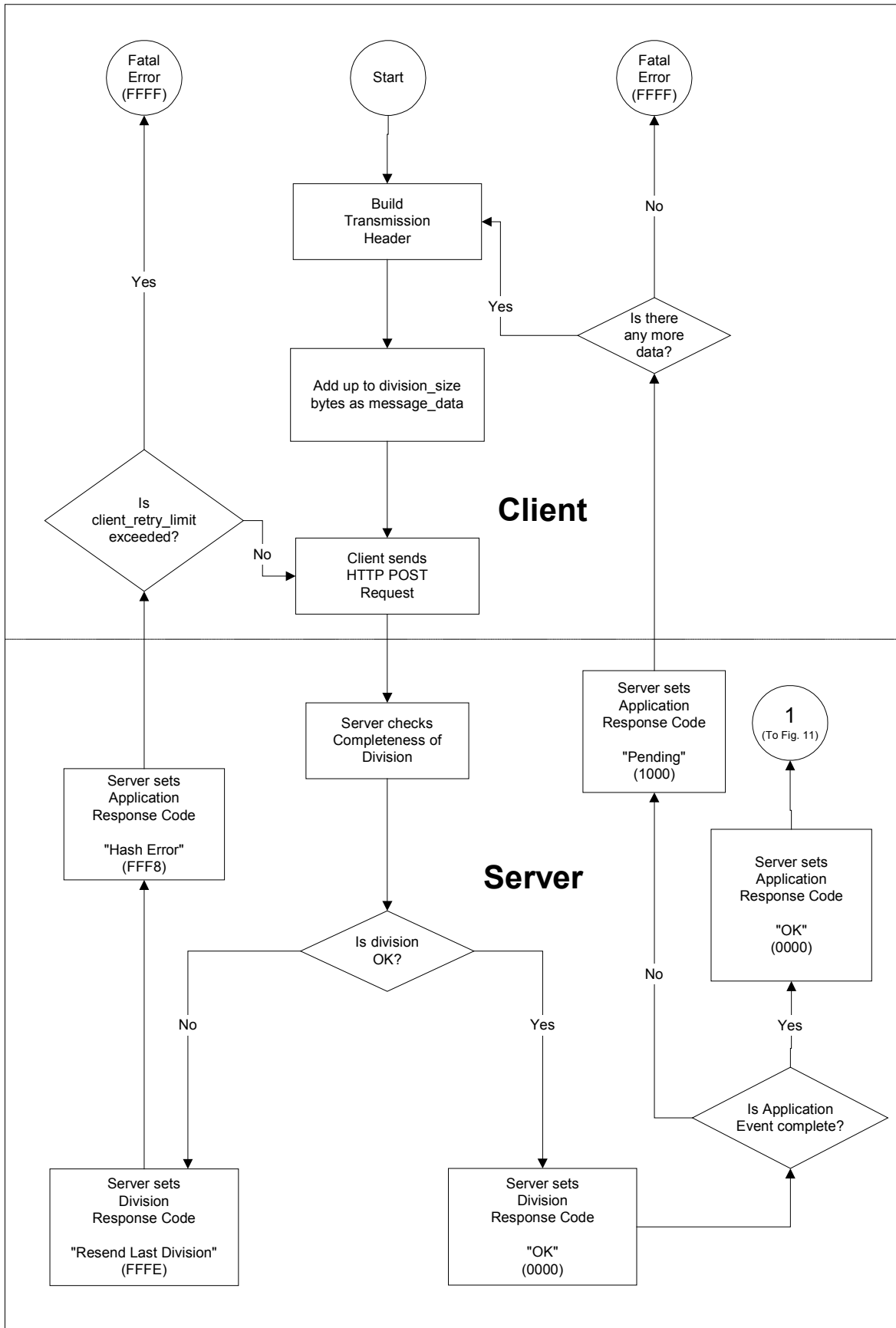


Figure 10 – Get package header behavior <upstream> [New]

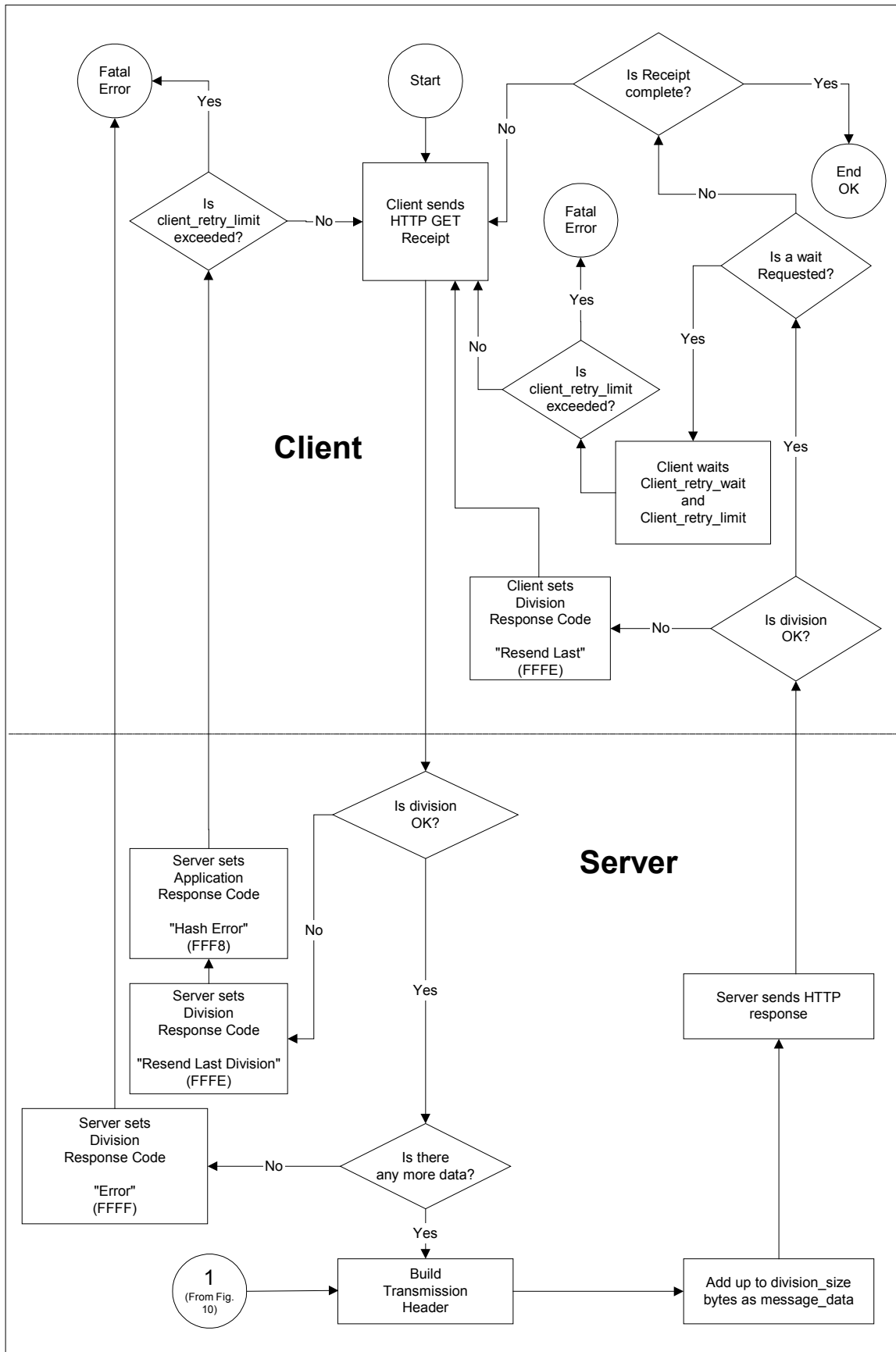


Figure 11 – Get package header behavior <downstream> [New]

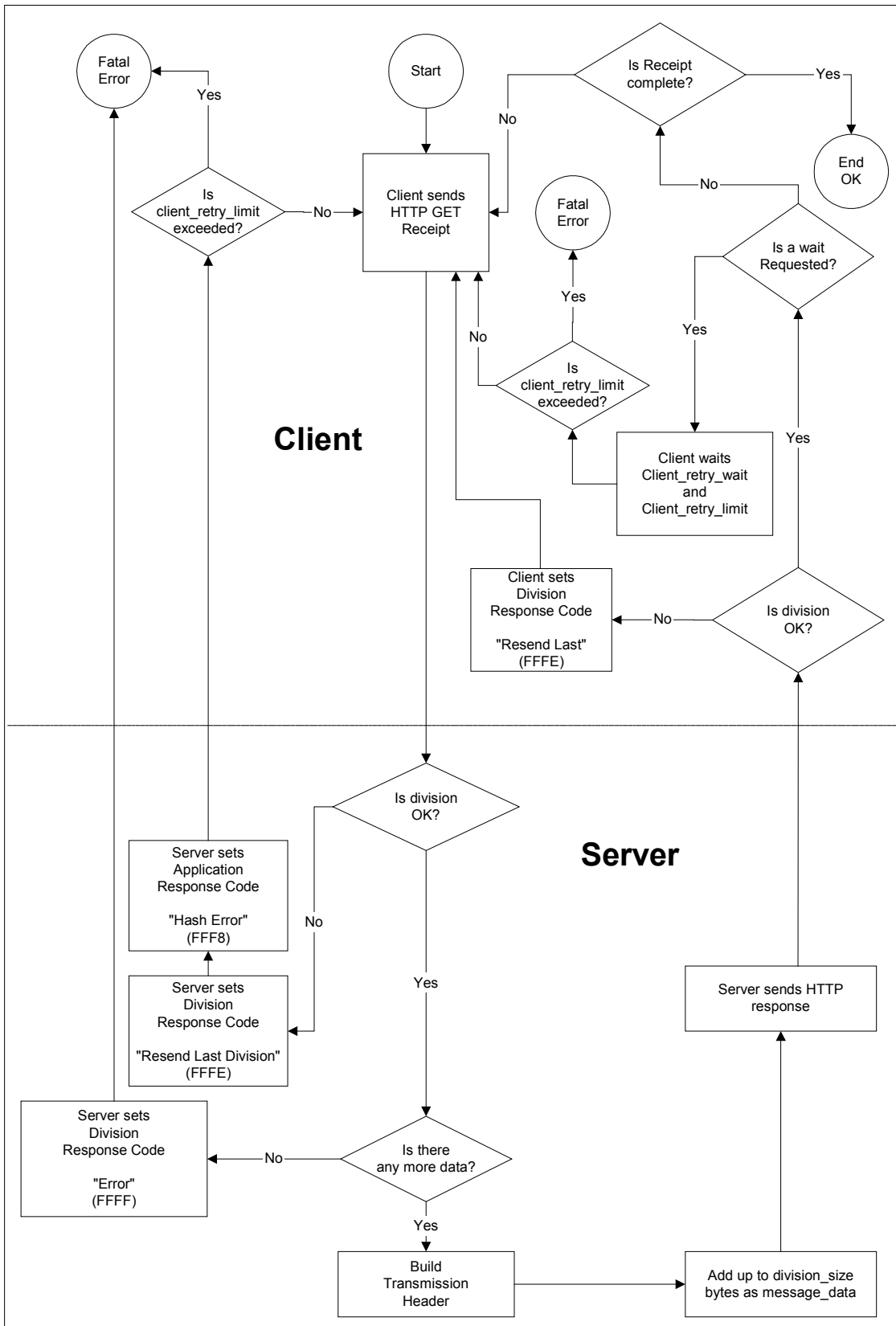


Figure 12 – Get package data behavior [New]

5.2 *Package/transmission combinations*

[No change to the introductory text]

5.2.1 *Applicant-Office (international phase) sector*

IA documents may be filed by on-line means (using PKI) over the public Internet, over a private network, or transmitted off-line (using PKI or non-PKI) on physical media. The option of on-line filing of an IA utilizing a non-PKI method is not presently permitted, except under possible transitional reservations permitted by AIs Section 703(f) (see section 7.1.1 as to the consequences of non-PKI filing under such a transitional reservation).

Figure 13 shows a matrix of the various submission mechanism/packaging combinations that are permissible in the Applicant-Office (international phase) sector as specified under this standard. In summary, for each submission mechanism:

- (a) On-line/Internet: The SEP must be used. TCP/IP used to exchange data, in realtime, over the Internet
- (b) On-line/secure: The SEP, WASP or C-WASP must be used. This is defined as a telecommunication connection established to exchange data, over a network which includes: 1) a private network; 2) the Internet using channel level encryption (e.g. SSL); 3) a Virtual Private Network (VPN) connection over the Internet.
- (c) Off-line/physical media: either the SEP, WASP, C-WASP or WAD package must be used. Physical media (e.g., diskette, CD-ROM, DVD, etc.) is used to store IA data with no real-time data exchange.

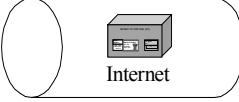








	Signed and Encrypted Package	Wrapped and Signed Package Compound WASP	Wrapped Application Documents
On-line / Internet	 Internet	 Not permissible	 Not permissible
On-line / secure	 Secure	 Secure	 Not permissible
Off-line / media			

Figure 13 – Package/transmission combinations permitted in the Applicant-Office (international phase) sector

5.2.2 Office-Office sector

All Office-Office sector data exchange must be conducted utilizing PKI-based data exchange. IA documents may be exchanged by on-line means over the public Internet or over a private network (such as Tri-Net or WIPONET), or transported on physical media.

Figure 14 shows a matrix of the various submission mechanism/packaging combinations that are permissible as specified under this standard. In summary, for each data exchange mechanism:

- (a) On-line/Internet: The SEP must be used. TCP/IP used to exchange data, in realtime, over the Internet
- (b) On-line/secure: The SEP or WASP must be used. This is defined as a telecommunication connection established to exchange data, over a network which includes: 1) a private network (e.g. WIPONET, Tri-Net); 2) the Internet using channel level encryption (e.g. SSL); 3) a Virtual Private Network (VPN) connection over the Internet.
- (c) Off-line/physical media: The SEP or WASP must be used. Physical media (e.g. diskette, CD-ROM, DVD, etc.) is used to store IA data with no real-time data exchange.

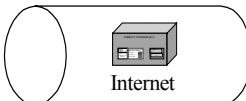








	Signed and Encrypted Package	Wrapped and Signed Package	Wrapped Application Documents
On-line / Internet	 Internet	 Not permissible	 Not permissible
On-line / secure	 Secure	 Secure	 Not permissible
Off-line / media			 Not permissible

Figure 14 – Package/transmission combinations permitted in Office-Office sector

5.2.3 *Designated Office sector*

The SEP, WASP, or WAD package may be used when exchanging IA documents under the designated Office sector. Figure 15 shows a matrix of the various submission mechanism/packaging combinations that are permissible. In summary, for each data exchange mechanism:

- (a) On-line/Internet: the SEP must be used. TCP/IP used to exchange data, in realtime, over the Internet
- (b) On-line/secure: the SEP, WASP, or WAD must be used. This is defined as a telecommunication connection established to exchange data, over a network which includes: 1) a private network (e.g. WIPONET, Tri-Net); 2) the Internet using channel level encryption (e.g., SSL); 3) a Virtual Private Network (VPN) connection over the Internet.
- (c) Off-line/physical media: either the SEP, WASP, or WAD package must be used. Physical media (e.g., diskette, CD-ROM, DVD, etc.) is used to store IA data with no real-time data exchange.

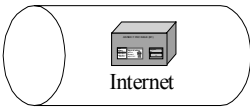








	Signed and Encrypted Package	Wrapped and Signed Package	Wrapped Application Documents
On-line / Internet	 Internet	 Not permissible	 Not permissible
On-line / secure	 Secure	 Secure	 Secure
Off-line / media	 Internet	 Secure	 Secure

Figure 15 – Package/transmission combinations permitted in designated Office sector

6. to 9. [No change]

[End of document]