国 际 电 信 联 盟

ITU-T

国际电信联盟
电信标准化部门

T.808

（01/2005）

T系列：远程信息处理业务终端

信息技术 — JPEG 2000图像编码系统：交互式工具、API和协议

ITU-T T.808建议书

国际电信联盟

国际标准 ISO/IEC 15444-9
ITU-T T.808建议书

信息技术 – JPEG 2000图像编码系统：交互式工具、API和协议

摘　要

本建议书|国际标准的目的是为了提供一个网络协议，允许从服务器到客户端进行交互式和渐进传输JPEG 2000编码的数据和文件。本协议允许客户端仅请求适用于客户端需求的部分图像（通过区域、质量或者分辨率级别）。本协议也允许客户端存取文件的元数据或其它内容。

来　源

ITU-T第16研究组（2005-2008）按照ITU-T A.8建议书规定的程序，于2005年1月8日批准了ITU-T T.808（2005年）建议书。相同的文本还以ISO/IEC 15444-9形式出版。

前　言

国际电信联盟（ITU）是从事电信领域工作的联合国专门机构。ITU-T（国际电信联盟电信标准化部门）是国际电信联盟的常设机构，负责研究技术、操作和资费问题，并且为在世界范围内实现电信标准化，发表有关上述研究项目的建议书。

每四年一届的世界电信标准化全会（WTSA）确定ITU-T各研究组的研究课题，再由各研究组制定有关这些课题的建议书。

WTSA第1号决议规定了批准建议书须遵循的程序。

属ITU-T研究范围的某些信息技术领域的必要标准，是与国际标准化组织（ISO）和国际电工技术委员会（IEC）合作制定的。

注

本建议书为简要而使用的“主管部门”一词，既指电信主管部门，又指经认可的运营机构。

遵守本建议书的规定是以自愿为基础的，但建议书可能包含某些强制性条款（以确保例如互操作性或适用性等），只有满足所有强制性条款的规定，才能达到遵守建议书的目的。“应该”或“必须”等其它一些强制性用语及其否定形式被用于表达特定要求。使用此类用语不表示要求任何一方遵守本建议书。

知识产权

国际电联提请注意：本建议书的应用或实施可能涉及使用已申报的知识产权。国际电联对无论是其成员还是建议书制定程序之外的其它机构提出的有关已申报的知识产权的证据、有效性或适用性不表示意见。

至本建议书批准之日止，国际电联尚未收到实施本建议书可能需要的受专利保护的知识产权的通知。但需要提醒实施者注意的是，这可能不是最新信息，因此大力提倡他们查询电信标准化局（TSB）的专利数据库。

目 录

图

表

引言

ITU-T T.800建议书 | ISO/IEC 15444-1 （JPEG 2000）是描述一个图像压缩系统的规范，该系统不仅对图像压缩，还对码流的存取提供了极大的灵活性。码流提供了多种定位和提取部分被压缩的图像数据的机制，为了重传、存储、显示或者编辑。这种存取允许存储和获取适于某个应用的被压缩的图像数据而不必解码。

本建议书|国际标准的目的是为了提供一个网络协议，允许从服务器到客户端进行交互式和渐进传输JPEG 2000编码的数据和文件。本协议允许客户端仅请求适用于客户端需求的部分图像（通过区域、质量或者分辨率级别）。本协议也允许客户端存取文件的元数据或其它内容。

任何想要使用本建议书|国际标准的组织应仔细考虑它的适用性。

国际电信联盟（ITU）、国际标准化组织（ISO）和国际电工委员会（IEC）提请注意一个事实，即据声称，使用本建议书|国际标准可能会涉及到对专利的使用。

ITU、ISO和IEC对有关这个专利权的证据、有效性和范围不表示意见。

这个专利权人已经向ITU、ISO和IEC保证他愿意在合理且非歧视的条件与期限下和全世界的申请人进行许可谈判。在这方面，已向ITU、ISO 和IEC注册了这个专利权人的声明。可从附件M中列出的公司中获取相关信息。

提请注意，本建议书|国际标准中的某些部分可能是未在附件M中提到的专利权的题目。ITU、ISO和IEC不负责对任何或者全部这种专利权的鉴别。

国际标准
ITU-T 建议书

信息技术 — JPEG 2000 图像编码系统：交互性工具、API和协议

1 范围

本建议书|国际标准以可扩展的方式定义了对符合ISO/IEC 15444中如下部分定义的JPEG 2000码流和文件进行远程访问和随意修改的语法和方法：

— ITU-T T.800建议书| ISO/IEC 15444-1:2004中对JPEG 2000码流和JP2文件格式的定义。

— ISO/IEC 15444其它部分中对JPEG 2000系列文件格式的定义。

本建议书|国际标准定义的语法和方法，称为JPEG 2000交互协议，“JPIP”，使用JPIP的交互型应用称为“JPIP系统”。

JPIP是指一个在客户端和服务器之间交互的一组结构化的序列所组成的协议，通过这种协议，可以以通信效率的方式交换图像文件的元数据、结构、部分或全部图像码流。本建议书|国际标准包括对被交换的语法和值的定义，以及如何使用现有的各种网络传输进行传送的建议。

使用JPIP，可以多种兼容的方式实现以下功能：

— 能力的交换；

— 会话中使用的能力的协商；

— 对多种容器，如JPEG 2000系列文件、JPEG 2000码流以及其它容器文件中的以下元素的请求和传递：

- 选择的数据分片段；
- 选择的和已定义的结构；
- 图像的部分或相关的元数据。

2 规范参考文献

下列ITU-T建议书和国际标准的条款，通过在本建议书中的引用而构成本建议书 | 国际标准的条款。在出版时，所指出的版本是有效的。所有的建议书和国际标准都面临修订，使用本建议书 | 国际标准的各方应探讨使用下列建议书和国际标准最新版本的可能性。IEC和ISO的成员维护当前有效的国际标准的登记表，ITU的电信标准化局维护当前有效的ITU-T建议书的清单。

– ITU-T Recommendation T.800 (2002) | ISO/IEC 15444-1:2004, *Information technology – JPEG 2000 image coding system: Core coding system.*

– ITU-T Recommendation T.801 (2002) | ISO/IEC 15444-2:2004, *Information technology – JPEG 2000 image coding system: Extensions.*

– ITU-T Recommendation T.802 (2005) | ISO/IEC 15444-3:2005, *Information technology – JPEG 2000 image coding system: Motion JPEG 2000.*

– ISO/IEC 15444-6:2003, *Information technology – JPEG 2000 image coding system – Part 6: Compound image file format.*

– IETF RFC 768 (1980), *User Datagram Protocol.* Available from World Wide Web: <http://www.ietf.org/rfc/rfc0768.txt>.

- IETF RFC 793 (1981), *Transmission Control Protocol*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc0793.txt>.
- IETF RFC 2046 (1996), *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2046.txt>.
- IETF RFC 2234 (1997), *Augmented BNF for Syntax Specifications: ABNF*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2234.txt>.
- IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI):* Generic Syntax. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2396.txt>.
- IETF RFC 2616 (1999), *Hypertext Transfer Protocol – HTTP/1.1*. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2616.txt>.

3 定义

下列定义适用于本建议书|国际标准。

3.1 JPEG 2000部分1中的定义

ITU-T T.800建议书| ISO/IEC 15444-1:2004中第3章和ITU-T T.801建议书| ISO/IEC 15444-2:2004第3章中的定义也适用于本建议书|国际标准。

3.2 HTTP中的定义

以下定义尽量和HTTP/1.1中一致。在有差别的情况下，应使用这些定义。

3.2.1 Connection 连接：为了通信在两个程序之间建立的传输层虚拟电路。

3.2.2 Entity 实体：作为请求或响应中的净荷被传送的信息。实体由实体－头域中的元信息和实体－主体中的内容组成。

3.2.3 Proxy 代理：中间程序，既作服务器又作客户端，代表其它客户端发起请求。可内部为请求提供服务，或者将请求传送给其它服务器，有可能经过翻译。

3.3 JPIP中的定义

本建议书|国际标准采用以下定义。在有些情况下，这些定义和其它标准和/或建议书中所使用的定义不同。

3.3.1 ache（client-side） 缓存（客户端侧）：客户端的缓存用于存储JPIP数据块。客户端可能缓存空间有限，需要不时的清除缓存在其中的JPIP数据块。

3.3.2 cacheable 可缓存的：若允许缓存存储一个响应消息的副本，用于应答后续的请求，这个响应被称为可缓存的。但即使一个资源是可缓存的，对于某个请求是否可以使用已被缓存的副本进行应答，还有另外的限制。

3.3.3 cache-model 缓存模型（服务器侧）：服务器对客户端缓存中可用的数据块部分的估算。当假设发送成功，或者收到传送数据的确认，或者缓存模型更新声明时，服务器会增加对客户端缓存估算的条目。

3.3.4 channel 通道：一种请求和应答的分组机制，在一个组里，同时只有一个请求/应答是激活的。多个同时的请求和应答需要多个通道。

3.3.5 client 客户端： 为了发送请求而建立连接的程序。

3.3.6 codestream image region 码流图像区域：码流图像区域是图像和由偏移量和区域尺寸定义的区域的交叉部分。码流图像区域可以是空的（即面积为0）。

3.3.7 data-bin 数据块：相同类型的可分部分传送的数据组成的一组字节。

3.3.8 incremental-codestream 增量码流：具有相同码流标识符的数据块（主头，分片头，分区或分片数据块）的集合的码流的表示。

3.3.9 JIIP index table JPIP索引表：提供文件或者码流各部分位置信息的文件格式框。

3.3.10 logical target 逻辑对象：对JPIP请求所指向的某个初始被命名的资源或者其中的一个字节范围的特定表示。这个特定表示可能是从初始命名的资源中转换而来。

3.3.11 message 消息：单个数据块中的一组字节以及标识那些字节和数据块的头。

3.3.12 raw-codestream 原始码流：单个元数据块的码流的表示。

3.3.13 request 请求：客户端发给服务器以获得图像的各部分或者元数据的一组域和值。

3.3.14 resource 资源：URI标识的网络数据对象或者服务，HTTP的对象。

3.3.15 response 响应：收到请求后服务器发送给客户端的字节。

3.3.16 server 服务器：为给请求提供服务，回送响应而接受连接的应用程序。任何程序可能既可以作客户端又可以作服务器，一般地，使用这些术语只是为了说明在一个连接中程序所执行的角色，并不是区分程序的能力。

3.3.17 session 会话：应用于同一个资源的所有请求和响应的集合，服务器为其维持一个缓存模型。

3.3.18 session-based 基于会话：服务器维持缓存模型的情况。

3.3.19 stateless 无状态：服务器在决定应答时不使用缓存模型的单个请求。

3.3.20 target 对象：JPIP数据的逻辑标识。主要对象的名字（往往是服务器上一个文件的名字）。

注 — JPEG 2000文件或者码流可有多种表示方法（如返回类型，分区尺寸）或不同的表示方式，每种都被标识为一个唯一的逻辑对象。

3.3.21 tile header 分片头：某个分片的所有部分分片的头。

3.3.22 view-window 视窗：客户端期望的图像数据的部分，由请求中以下域的组合来表示：区域尺寸，偏移量，帧尺寸，码流），码流上下文，抽样率，ROI以及图层。视窗往往比整个图像数据小。如果隐含视窗但未明确说明，应将视窗看作逻辑对象的所有图像数据。

3.4 符号

以下符号适用于本建议书| 国际标准。ITU-T T.800建议书| ISO/IEC 15444-1:2004第4章和ITU-T T.801建议书| ISO/IEC 15444-2:2004第4章中定义的符号同样适用于本建议书|国际标准。

c	分区所属的图像分量的索引（从0开始）
fx	客户端请求的视窗的x轴帧尺寸
fy	客户端请求的视窗的y轴帧尺寸
fx'	对于合适的码流分辨率的x轴帧尺寸
fy'	对于合适的码流分辨率的y轴帧尺寸
fx"	对于合适分辨率的修正jpx的x轴帧尺寸
fy"	对于合适分辨率的修正jpx的y轴帧尺寸
H_{cod}	图像标题（Image Header, ihdr）中记录的码流高度（参见ITU-T T.800建议书| ISO/IEC 15444-1:2004中附件I.5.3.1）
H_{comp}	JPX合成选项框规定的合成结果的高度（参见ITU-T T.801建议书	ISO/IEC 15444-2:2004中附件M.11.10.1）
H_{reg}	合成图层注册网关中显示的合成图层的高度
Hs_{inst}	切割高度
Ht_{inst}	合成高度
l	码流中分区的唯一标识符

N_L	分解的层数目
num_components	编码的分量数目
num_tiles	码流中的分片数目
ox	客户端请求的视窗的x轴偏移量
ox'	适当码流区域的x轴偏移量
ox"	适当区域的修正jpx的x轴偏移量
oy	客户端请求的视窗的y轴偏移量
oy'	合适的码流区域的y轴偏移量
oy"	合适的区域的修正jpx的y轴偏移量
r	分辨率级别
s	标识分片分量内分区的序列号A
sx	客户端请求的视窗的x轴尺寸
sx'	合适的码流区域的x轴尺寸
sx"	合适的区域的修正jpx的x轴尺寸
sy	客户端请求的视窗的y轴尺寸
sy'	合适的码流区域的y轴尺寸
sy"	合适区域的修正jpx的y轴尺寸
t	分区所属的分片的索引（从0开始）
W_{cod}	图像头（Image Header，ihdr）中记录的码流宽度（参见ITU-T T.800建议书| ISO/IEC 15444-1:2004中附件I.5.3.1）
W_{comp}	JPX合成选项规定的合成结果的宽度（参见ITU-T T.801建议书 | ISO/IEC 15444-2:2004中附件M.11.10.1）
W_{reg}	合成图层注册网关中显示的合成图层的宽度
Ws_{inst}	切割宽度
Wt_{inst}	合成宽度
XC_{inst}	相关指令指示的x轴切割偏移量（参见ITU-T T.801建议书 | ISO/IEC 15444-2:2004中附件M.11.10.2.1）
XO_{inst}	相关合成指令指示的x轴合成偏移量（参见ITU-T T.801建议书 | ISO/IEC 15444-2:2004中附件M.11.10.2.1）
XO_{reg}	x轴码流注册偏移量
XOsiz	从相关码流的SIZ标记段的参考网关开始的水平偏移量
XR_{reg}	码流注册框开始部分描述的x轴码流注册抽样因子（参见ITU-T T.801建议书| ISO/IEC 15444-2:2004中附件M.11.7.7）
Xsiz	相关码流的SIZ标记段的参考网格宽度
XS_{reg}	码流注册框开始部分描述的x轴注册区域（参见ITU-T T.801建议书| ISO/IEC 15444-2:2004中附件M.11.7.7）
YC_{inst}	相关指令指示的y轴切割偏移量（参见ITU-T T.801建议书| ISO/IEC 15444-2:2004中附件M.11.10.2.1）

$\mathbf{YO}_{inst}$	相关合成指令指示的y轴合成偏移量（参见ITU-T T.801建议书 | ISO/IEC 15444-2:2004中附件M.11.10.2.1）
$\mathbf{YO}_{reg}$	y轴码流注册偏移量
YOsiz	从相关码流的SIZ标记段的参考网格开始的垂直偏移量
$\mathbf{YR}_{reg}$	码流注册框开始部分描述的y轴码流注册抽样因子（参见ITU-T T.801建议书 | ISO/IEC 15444-2:2004中附件M.11.7.7）
Ysiz	相关码流的SIZ标记段的参考网格高度
$\mathbf{YS}_{reg}$	码流注册框开始部分描述的y轴注册区域（参见ITU-T T.801建议书 | ISO/IEC 15444-2:2004中附件M.11.7.7）

4 缩写

下列缩写适用于本建议书|国际标准。

ABNF	扩展巴科斯范式
DICOM	医学数字成像和通信
DWT	离散小波变换
EOR	响应结束
HTML	超文本标记语言
IP	互联网协议
JP3D	JPEG 2000第10部分：3-D和浮动点数据
JPIP	JPEG 2000交互协议
JPP	JPIP分区
JPSEC	JPEG 2000第8部分：安全的JPEG 2000
JPT	JPIP 分片部分
JPWL	JPEG 2000第11部分：无线
JTC 1	第1联合技术委员会
MTF	调制传递函数
PDF	便携文档格式
SC 29	第29小组委员会
SVG	可缩放矢量图形
TCP	传输控制协议
UDP	用户数据报协议
UUID	通用唯一标识符
VBAS	可变长字节对齐段
WG 1	第1工作组
XHTML	扩展超文本标记语言
XML	扩展标记语言

5 约定

5.1 ABNF规则

本建议书|国际标准使用RFC2234中定义的ABNF符号，包括核心ABNF语法规则：ALPHA（字母），CR（回车），CRLF（国际标准换行），CTL（控制字符），DIGIT（十进制数字），HEXDIG（十六进制数字），LF（换行），LWSP（线性空白字符）以及SP（空格符）。以下ABNF规则适用于本建议书|国际标准：

NZDIGIT = %x31-39 ; 1-9

UPPER = %x41-5A ; A-Z

LOWER = %x61-7A ; a-z

UINT = 1*DIGIT

NONZERO = *"0" NZDIGIT *DIGIT

UINT-RANGE = UINT ["-" [UINT]]

UFLOAT = 1*DIGIT ["." 1*DIGIT]

ENCODED-CHAR = "%" HEXDIG HEXDIG

UUID = 16(HEXDIG)

TOKEN = 1*(ALPHA / DIGIT / "." / "_")

本建议书|国际标准也定义了PATH，表示一个文件或路径名。通常，PATH值中可包含任何字符，尽管对于某个服务器体系来说，服务器应拒绝任何在该服务器上不合法的字符。另外，PATH还应根据不同的传输技术作正确的编码。

UINT-RANGE规定了整数的范围。范围中的第一个整数指这个范围的开始值。如果定义了两个值，则第一个和第二个值指这个范围所包含的开始和结尾。如果只定义了第一值和字符“-”，则这个范围包含所有大于或等于第一值的数值。

ABNF元素前面紧接一个数值是指复制跟在数字后面的参数，复制的次数由该数字定义，且每个副本之间不插入空格。

结构“1#”指一次或多次复制跟在其后的参数，每个副本之间用逗号隔开。

结构“1$”指一次或多次复制跟在其后的参数，每个副本之间用分号隔开。

5.2 文件格式的ABNF规则

```
compatibility-code = 4(ALPHA / DIGIT / "_" / ENCODED-CHAR)

box-type = 4(ALPHA / DIGIT / "_" / ENCODED-CHAR)

box-type-list = "*" / 1#(box-type)
```

box-type规定了框类型的四个字符。对于框类型中的每个字符，如果该字符是字母或数字（A..Z, a..z 或者 0..9），直接写入字符串中，如果该字符是一个空格符（0x20），应将其转译为下划线（"_"）。对于其它字符，应在它的位置写成3个字符串，即百分号("%")后跟两个代表框类型字符的十六进制数字。compatibility-code的编码方式和box-type相同。

box-type-list规定了框类型的列表。如果box-type-list的值是"*"，则本域指所有的框类型。

5.3 框的图形描述的要点（资料性）

每个框的描述后跟一个图，用以说明框中各个参数的顺序和关系。如图1就是这种图的一个例子。用矩形表示框中的参数。矩形的宽度和参数的比特数成正比。有阴影的矩形（斜纹）表示该参数是可变长的。标有上标的两个参数和中间的灰色区域表示有多个这种参数的排列。两组由灰色区域隔开的多个标有上标的参数组成的序列表示这组参数的排列（一组参数后再跟另一组参数）。可选的参数或者框用虚线矩形表示。

图后跟着对框中每个参数的意义的描述。如果是重复参数，则会定义参数的长度和排列的方式。例如，图1中，参数A、B、C和D分别为8、16、32和可变长度。符号E^0 和 E^{N-1}表示有N个不同的E^i参数排成

一列。一组参数F^0和F^{M-1}，以及G^0和G^{M-1}表示框中包括F^0，然后G^0，再跟F^1和G^1，一直到F^{M-1}和G^{M-1}（每个参数共M个实例）。D域是可选的，不一定出现在该框中。

另外，在描述超级框的图中，省略号（…）表示两个框之间的文件内容未明确定义。除非框定义中另外规定，任何框（或框序列）都可能出现在省略号处。

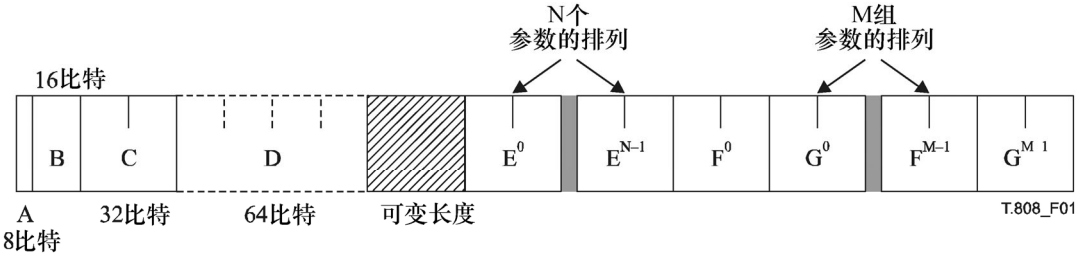

图 1－框的描述图的示例

例如，图2中显示的超级框必须包含一个AA框和BB框，且BB框必须在AA框后面。但在AA框和BB框之间可以是任何其它的框。对未知框的处理参见ITU-T T.800建议书 | ISO/IEC 15444-1:2004中的附件I.8。

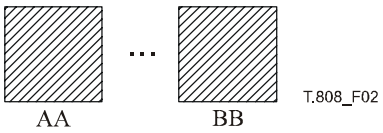

图 2－超级框的描述图的示例

6 概述

6.1 JPIP协议

本建议书|国际标准描述了客户端存取存放在支持JPIP服务器中的JPEG 2000压缩图像以及和图像相关的数据的语法和方法。本建议书|国际标准通过多种客户端/服务器传输实现了ITU-T T.800建议书| ISO/IEC 15444-1:2004提出的灵活性和功能。

JPIP定义了使JPEG 2000图像以及图像相关的数据能有效交换的交互协议。协议基于客户端的请求和服务器的响应，定义了客户端－服务器之间的交互过程，如图3所示。本建议书|国际标准定义了JPIP客户端的请求和JPIP服务器的响应。作为例子，图中列出了HTTP/1.1 (RFC 2616)、TCP (RFC 793)和UDP (RFC 768)作为承载JPIP的可能的传输协议。客户端用一个视窗请求定义它所请求的图像以及和图像相关的数据的分辨率、尺寸、位置、分量、图层和其它参数。服务器以基于分区的流、基于分片的流或整个图像传送图像以及和图像相关的数据作为响应。协议还允许客户端和服务器之间进行能力和限制条件的协商。客户端可能向服务器请求在JPIP索引表中定义的一个图像的信息，从而能够确定它的视窗请求中图像的具体参数（如字节范围的请求）。服务器的缓存模型是基于客户端定义的能力和会话状态的。

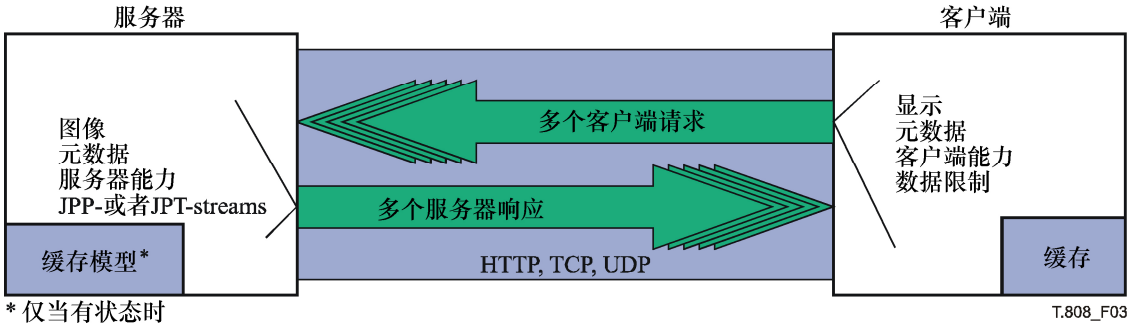

图 3—JPIP 协议概况

本协议可在不同在传输上使用，如图4所示。本建议书|国际标准包括关于在HTTP和TCP上使用JPIP协议的资料性附件，并提供了其它示例实现的建议。

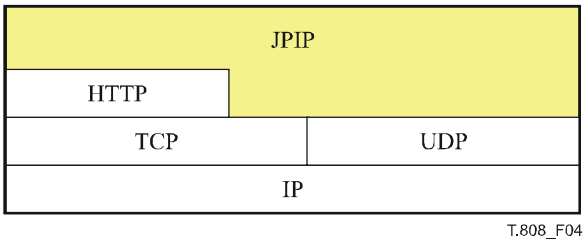

图 4—JPIP 协议栈

为支持目前的JPEG 2000标准，对JPIP协议的扩展的规定包括：ITU-T T.802建议书| ISO/IEC 15444-3、运动JPEG 2000、ISO/IEC 15444-6、复合文档以及JPEG 2000的未来部分（目前包括JP3D，JPSEC和JPWL）。

6.2 目的

本建议书|国际标准定义了客户端和服务器所需的语法和方法。每个附件定义了客户端和服务器之间在几种传输上获得互操作性和功能性所需的一部分。每个附件可能对客户端是必需的，也可能对服务器是必需的，或者两者兼之。附件的描述如下。

— 附件A描述了对客户端和服务器都必需的基于分片和基于分区的流。服务器要求能产生适当的JPP- 和JPT-streams，理解上传的JPP- 和 JPT-streams。客户端要求能理解和正确解码这些流，且当要上传部分图像给服务器时，负责产生合适的流。

— 附件B描述了客户端/服务器会话的会话模型和缓存模型，这对客户端和服务器都是必需的。

— 附件C定义了客户端请求的语法。客户端应能产生适当的请求，且服务器应能理解并响应所有的适当请求。

— 附件D定义了服务器响应的语法。服务器应产生适当的响应，且客户端应能理解这些适当的响应。

— 附件E对使用JPIP上传的系统定义了上传部分图像的语法和方法。

— 附件F、G和H定义了JPIP客户端/服务器在几个不同传输协议上交互的方法和程序。

— 附件I定义了包含在JPEG 2000框中的索引信息，客户端和服务器使用这些信息可以更有效的存取图像以及和图像相关的数据。

— 附件J定义了如何通过注册扩展本标准。

— 附件K描绘了几个使用本建议书|国际标准的不同应用的例子。

7 一致性

客户端对本建议书|国际标准的一致性是指客户端的JPIP请求结构规范、正确，和本建议书|国际标准定义的JPIP客户端请求一致。客户端应支持所有标准化的请求类型。

服务器对本建议书|国际标准的一致性是指服务器的JPIP响应结构规范、正确，和本建议书|国际标准定义的JPIP服务器响应信令一致。服务器应支持所有标准化的请求类型。

虽然未定义一致性行为，但建议执行本建议书|国际标准，使得基于客户端的应用需求能以高效的JPIP请求来请求图像数据。

同样地，图像数据应在高效的JPIP服务器响应的基础上被服务，从而使不在客户端指定的感兴趣范围内被服务的数据以及客户端已经拥有的冗余数据数量最少。但未定义一致性行为。

根据不同的网络服务质量，服务器应用可能需发送附加或冗余的数据，这可能会降低效率。这种实现的决定和具体的应用有关，且为JPIP系统提供了高利用率。但本建议书|国际标准未定义启用这种操作的一致性。

附 件 A

JPP-stream和JPT-stream媒体类型

（本附件是本建议书|国际标准的组成部分）

A.1 引言

JPP-stream和JPT-stream是对以任意顺序呈现JPEG 2000码流和文件格式数据的有用的媒体类型。每种媒体类型由一个连续的消息序列组成，其中每个消息包括一个消息头以及后面的单个数据块的一部分。数据块包括JPEG 2000压缩图像描绘的各部分，因此构造一个完整描绘一个JPEG 2000文件或码流中信息的流是可能的。每个消息是完全自描述的，因此消息序列可以在任何点上终结，且消息也可能在最小的限制条件下进行重新排列而不丢失它们的意义。由于这些原因，JPP-stream和JPT-stream媒体类型对JPIP服务器非常有用，且JPIP协议就是用这些媒体类型进行设计的。本附件定义了JPP-stream和JPT-stream媒体类型，未参考JPIP协议。

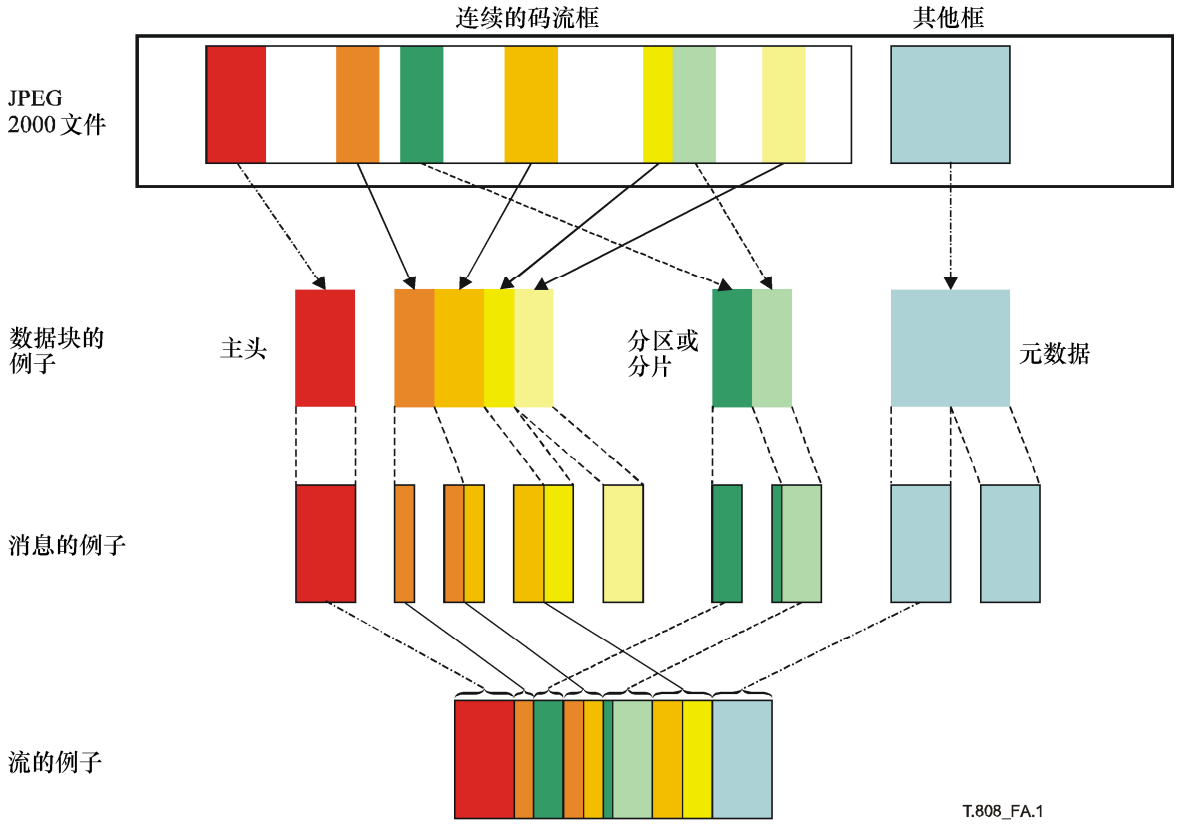

图 A.1－JPEG 2000 文件、JPIP 数据块、JPIP 流之间的关系示例（源自 G.J. Colyer 和 R.A. Clark, IEEE Trans. Consumer Electronics, 49 (2003), pp. 850–854）

图A.1是JPEG 2000文件的位流、JPEG数据块和JPIP流之间关系的示例。图中用红色表示主头，橙色和绿色阴影表示2个分区，蓝色表示元数据框。自描述的JPIP消息由这些数据块组成，且将其连接起来组成JPIP流。

JPIP流由一个或多个连续的JPIP消息组成，每个JPIP消息由一个头和一个主体组成。头提供了识别相关数据块的描述信息。主体就是数据块中的数据。除非提供其它的信令，消息一般由头和主体串接而成。

注 — 在本建议书|国际标准中，所有举例的二进制消息都由头和主体串接组成。如果提供了关于头和主体的其它信令，则具体的实现和传输以及应用程序有关。例如，对于基于无线的应用，可能实现具有可变错误保护的辅助信令。

A.2 消息头结构

A.2.1 综述

每个消息只描述一个数据块的一部分。消息头由一系列可变长度的字节对齐段（VBAS）的组成。每个VBAS由字节序列组成，除最后一个字节外，其它字节都有一个有效比特位（比特7）为1，如图A.2所示。VBAS中每个字节的低7位串联起来组成一个比特流，对于不同的VBAS具有不同的使用方式。

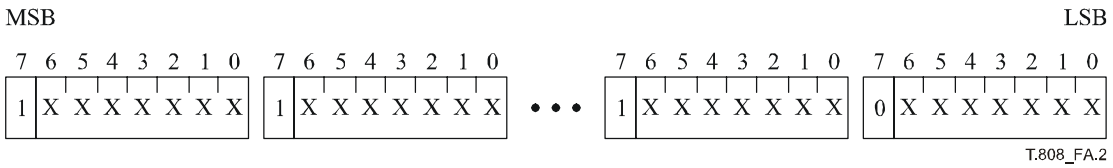

图 A.2－VBAS 的结构

消息头用于识别消息体中表示的特定的数据块和字节范围。消息头可以采用独立形式也可以采用非独立形式。独立形式是一种长形式，其中消息头是完全自描述的，对它们的解释和其它消息头无关。可选的较短的非独立形式的消息头利用前面消息头的信息，对它们的解码和前面的消息有关。应用若选择使用长形式的消息头，则这些消息可以以任意顺序重新排列。否则，应用若使用依赖前面消息头的较短形式的消息头，则这些消息较短，但如果在解码时没有以正确的顺序进行排列，将会得出错误的结果。由应用程序决定接收到的消息顺序是否可认为是可靠的，从而决定是否使用较短形式的消息头。

消息头由以下VBAS组成（可选的VBAS用中括号表示）

Bin-ID [, Class] [, CSn], Msg-Offset, Msg-Length [, Aux]

Bin-ID VBAS决定了Class和CSn VBAS是否存在。Class VBAS或者如果当前消息头中没有Class VBAS，由前面的Class VBAS决定Aux VBAS是否存在。

Bin-ID VBAS有几个作用。Bin-ID VBAS中第一个字节的比特6和5，在图A.3中用'b'标出，表示消息头中是否有Class和CSn VBAS。表A.1定义了该比特的值及其意义。

Bin-ID VBAS中第一个字节的比特4，在图A.3中用'c'标出，表示该消息是否包含相关数据块中的最后一个字节：'0'表示不包含数据块中的最后一个字节，'1'表示包含数据块中的最后一个字节。接收到该比特位置位的消息后，就能决定整个数据块的长度，但这并不意味整个JPP-stream或JPT-stream必须包括足够的消息从而集合该数据块中的所有字节。

Bin-ID VBAS中第一个字节的其它4个比特以及其它字节中的低位7位比特（图A.3中用'd'标出）组成“类内标识符”，用于唯一识别类内的数据块，方式见A.2.3中描述。

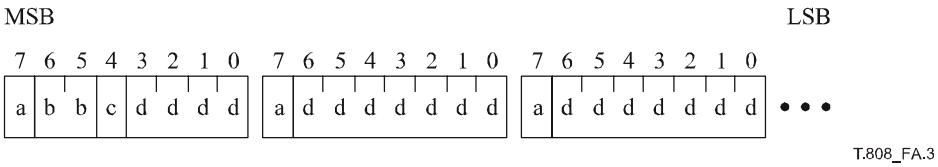

图 A.3－Bin-ID VBAS 的结构

表 A.1－Bin-ID 中附加 VBAS 的指示

指示位'bb'	意义
00	禁止
01	消息头中无 Class 和 CSn VBAS
10	消息头中有 Class VBAS，但没有 CSn
11	消息头中有 Class 和 CSn VBAS

Class VBAS如果存在，规定了消息的类标识符。消息的类标识符是一个非负整数，由VBAS中每个字节的后7位有效比特按big-endian字节顺序串联组成。如果Class VBAS不存在，消息的类标识符和相邻的前面的消息相同。如果Class VBAS不存在且前面没有消息，则消息的类标识符为0。有效的消息类标识符在A.2.2中描述。

CSn VBAS如果存在，规定了数据块所属的码流的索引（从0开始）。码流索引由VBAS中每个字节的后7位有效比特按big-endian字节顺序串联组成。如果CSn VBAS不存在，则码流索引和前面的消息相同。如果CSn VBAS不存在且前面没有消息，则码流索引为0。

Msg-Offset和Msg-Length VBAS都是非负整数，由VBAS中每个字节的后7位有效比特按big-endian字节顺序串联组成。Msg-Offset整数表示消息中的数据相对于数据块起始位置的偏移量。Msg-Length整数表示消息体中的字节数。

Aux VBAS是可选的。由Bin-ID VBAS中消息的类标识符确定它是否存在及其意义，见A.2.2中解释。Aux VBAS如果存在，表示为一个非负整数，由VBAS中每个字节的后7位有效比特按big-endian字节顺序串联组成。

注 — Aux VBAS中的信息不改变该消息体的长度。

A.2.2 消息的类标识符

本建议书|国际标准定义的消息的类标识符如表A.2中列出的非负整数。它们所涉及的数据块类型的解释如A.3中描述。消息的类标识符的所有其它值均保留，且当解码器不能识别该值的时候，应跳过相应的消息。

当且仅当类标识符为奇数时，Aux VBAS存在。这种特性使不能识别的消息头能被正确的解析，并跳过其内容。

扩展的分区数据块消息和非扩展的分区数据块消息的解析是完全一样的，且它们所涉及的分区数据块也完全相同。扩展的分区消息中包含一个Aux VBAS，表示完成的数据包（有效图层）的数量，该数量可通过将该消息中的字节和所有前面相同分区的字节串联起来得到。如果该消息还包含数据块的最后一个字节，则Aux VBAS表示在原来码流中和该分区相关的有效图层的总量。否则，Aux VBAS表示紧跟在消息最后一个字节后面的那个字节所属的有效图层。Aux VBAS中的信息可能对某些客户端有用。

表 A.2－对于不同数据块消息类的类标识符

类标识符	消息类型	数据块类型	流类型
0	分区数据块消息	分区数据块	JPP-stream
1	扩展的分区数据块消息	分区数据块	JPP-stream
2	分片头数据块消息	分片头数据块	JPP-stream
4	分片数据块消息	分片数据块	JPT-stream
5	扩展的分片数据块消息	分片数据块	JPT-stream
6	主头数据块消息	主头数据块	JPP-和 JPT-stream
8	元数据块消息	元数据块	JPP-和 JPT-stream

扩展的分片数据块消息和非扩展的分片数据块消息的解析是完全一样的，且它们涉及的分片数据块也完全相同。扩展的分片消息包括一个Aux VBAS，规定了最小的n值，使对于所有分量（$N_L - n$）为非负，且

当将该消息中的字节和所有前面相同分片的字节串联起来时，已经完成了($N_L - n$)及其所有更低的分辨率级别，其中N_L为分解的层数目，依不同分量具有不同的值。如果任何分量的所有分辨率级别都没有完成，则Aux VBAS的值等于1加上所有分量中最大的N_L值。当所有分量的分辨率都已经完成时，值达到0。由于分辨率不一定在分片中顺序出现，一些大于VBAS中值的分辨率级别可能已经完成，但这并不能根据消息头确定。Aux VBAS中的信息可能对某些客户端有用。

A.2.3 类内标识符

Bin-ID VBAS中第一个字节的低位4比特以及所有其它字节中的低位7比特按big-endian字节顺序串联起来，组成一个具有7*k*-3个比特的值，其中*k*指VBAS中的字节数。该值是一个无符号型整数，用于唯一识别类内和码流内的数据块。A.3描述了各种数据块类型和相应的类内标识符。

A.3 数据块

A.3.1 引言

数据块包含JPEG 2000文件或码流数据的各部分。数据块可基于图像元素，如基于分区的数据、基于分片的数据和头。也可以基于元数据。不管数据块的内容是什么，每个数据块作为单独的比特流进行处理。

A.3.2 分区数据块

A.3.2.1 分区数据块的格式

分区数据块仅出现在JPP-stream媒体类型中。每个分区数据块对应于一个码流中的一个分区。其类内标识符由方程式A-1定义。

$$I = t + (c + s \times \text{num\_components}) \times \text{num\_tiles} \qquad \text{(A-1)}$$

其中：

I 为码流中分区的唯一标识符；

t 为该分区所属的分片的索引（从0开始）；

c 为该分区所属的图像分量的索引（从0开始）；

s 为分片分量中识别该区域的序列号。

在每个分片分量中，依次给每个区域分配连续的序列号*s*。先给分辨率级别最低（即只含有LL子带样本）的所有分区编号，从0开始，随后是光栅扫描序号。每个连续分辨率级别的区域被依次编号，同样后面跟分辨率级别内的光栅扫描序号。

分区标识符为0指分片0中图像分量为0的LL子带上面左手边的区域。

每个分区数据块对应于由所有的码流数据包串联组成的字节串，这些码流数据包包括所有属于该区域的相关的数据包的数据包头。数据包数据包头可能被打数据包成属于主头或者分片头数据块的PPM或PPT标记段，在这种情况下，分区数据块中只包含数据包主体。在任何情况下，分区数据流应符合在具有下属层的进程顺序（CPRL, PCRL 或 RPCL）中一种的JPEG 2000码流中出现的连续的字节段。

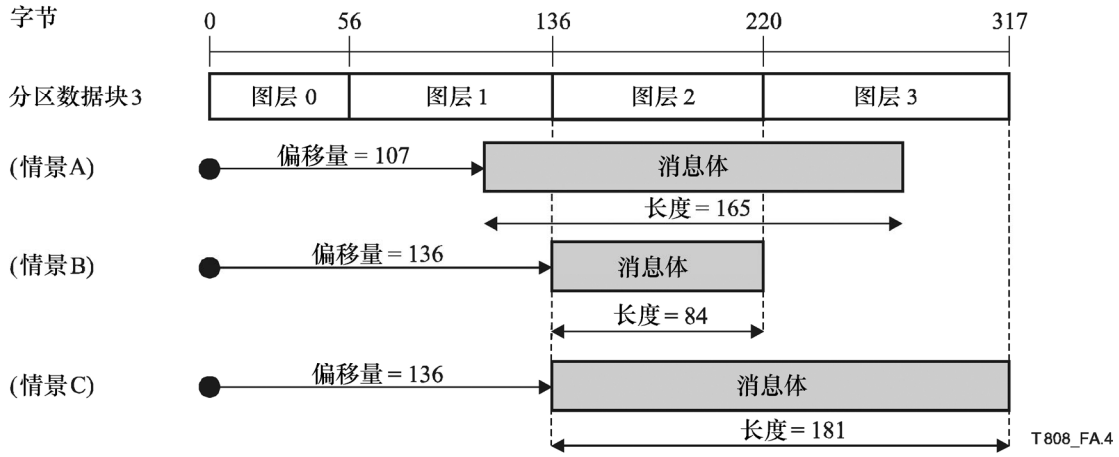

图 A.4－分区数据块的例子

A.3.2.2 分区数据块的例子（资料性）

图A.4显示了一个具有4个有效图层（或数据包）的分区数据块（类内标识符为3）的例子。

根据扩展还是非扩展的分区数据消息结构，情况A、B、C的消息头如下面所示。带下划线的数据表示Aux VBAS，标识由该消息完成的图层数量。

（情景A）

非扩展头： 00100011 01101011 10000001 00100101 xxxxxxxx …

首位0表示Bin-ID VBAS只有一个字节。接下来的两位（"01"）表示Class 和CSn VBAS 都不存在。随后的"0"表示本消息未完成数据块。第一个字节的剩余几位（"0011"）表示bin-ID是3。第二个字节中的第一位表示Msg-Offset VBAS只有一个字节。随后的7位（"1101011"）表示偏移量位107。第三个字节中的第一位表示本字节和至少下一个字节都是Msg-Length VBAS的一部分。第四个字节的首位0表示它是Msg-Length VBAS的最后一个字节。因此，所有第3、4字节中的低比特位串联起来确定了消息的长度。在本情景中，"0000001 0100101" = 165。

扩展头： 01000011 00000001 01101011 10000001 00100101 00000011 xxxxxxxx …

（情景B）

非扩展头: 00100011 10000001 00001000 01010100 xxxxxxxx …

扩展头: 01000011 00000001 10000001 00001000 01010100 00000011 xxxxxxxx …

(情景C)

非扩展头: 00110011 10000001 00001000 10000001 00110101 xxxxxxxx …

扩展头: 01010011 00000001 10000001 00001000 10000001 00110101 00000100 xxxxxxxx …

注意，在情景C中由于响应数据包含数据块的最后一个字节，Bin-ID VBAS指示了它是一个“完整的”消息。

A.3.3 分片头数据块

分片头数据块仅出现在JPP-stream媒体类型中。属于这种类型的数据块，其类内标识符包含该数据块所涉及的分片的索引（从0开始）。对分片n，这种数据块由标记和标记段组成。不包含SOT标记段。是否包含SOD标记是可选的。这种数据块可由一个合法的码流构成，通过将分片n的所有分片部分头中除SOT和POC以外的标记段串联起来。

A.3.4 分片数据块

分片数据块仅出现在JPT-stream媒体类型中。属于这种类型的数据块，其类内标识符包含该数据块所属的分片的索引（从0开始）。每个分片数据块对应于属于该分片的所有分片部分串联形成的字节串，因此，也包括它们的SOT、SOD和所有其它相关的标记段。

A.3.5 主头数据块

JPP-和JPT-stream都使用主头数据块。属于码流主头类型（完整的或不完整的变化）的数据块，其类内标识符应为0。这种数据块由主头中所有标记和标记段串联组成，以SOC标记开始。它不包含SOT，SOD和EOC标记。

A.3.6 元数据块

A.3.6.1 元数据块的介绍

JPP-和JPT-stream都使用元数据块。元数据块用于从包含一个或多个码流的逻辑对象传送元数据，码流的元素可被其它和JPP-stream或JPT-stream相关的数据块所引用。在本建议书|国际标准中，“元数据”这个术语指JPEG 2000系列文件中“框”的任意集成。在有元数据块类标识符的消息中，应忽略码流索引。

和其它数据块类型的数字ID不同，元数据块的ID和某个文件类型结构或字节偏移量并不是算术对应的。服务器可自由的为任何特定的元数据块选择任意的数字ID。有且仅有一个例外，即包含逻辑对象的根元数据块，其ID应为0。

注 — 分配机制和具体的实现相关，但资料性的建议服务器用连续的数字分配bin-ID。

A.3.6.2 将一个包含JPEG 2000文件的逻辑对象分割成元数据块

所有的元数据都可以毫无疑义的放在元数据块0中。在这种情况下，逻辑对象的所有框都属于元数据块0，且按照其原来的顺序排列。既然JPEG2000系列文件格式仅由框的序列组成，这意味着元数据块0由整个逻辑对象组成。然而，在大多数情况下，以便于管理的方式将逻辑对象分割成多块进行传输是非常有用的。这允许图像服务器有意的忽略客户端目前尚未请求的那部分逻辑对像。本节后面，JPIP定义了一种新的特殊的框类型，叫做“占位符（Placeholder）框”。占位符框用于标识逻辑对象的某个框的尺寸和类型，并指向另一个包含那个框内容的数据块。占位符还能标识逻辑对象的码流。这具有非常重要的意义，因为这样，用任意给定码流表示的压缩数据还可以通过其它数据块类型（头数据块、分区数据块或者分片数据块）进行传送。

形式上，元数据块0由逻辑对象的所有框组成，按照原来的顺序排列，但例外的是占位符可替代任何给定的框。占位符框包括被替代的框的原始头，以及包含那个框内容的元数据块的标识符，不包括自己的头。除了元数据块0的每个元数据块，应由某个框的内容组成，该框的头出现在引用该数据块的占位符中。这些框的内容本身可包括子框，子框可进一步由占位符替代。

以下颜色图案用于示例元数据块（图A.5）：

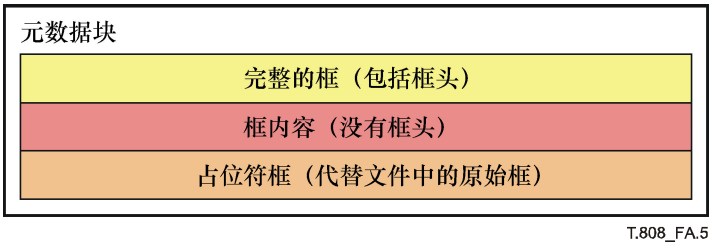

图 A.5－元数据块的颜色图案例子

例如，考虑具有以下框结构的一个简单JP2文件（图A.6）：

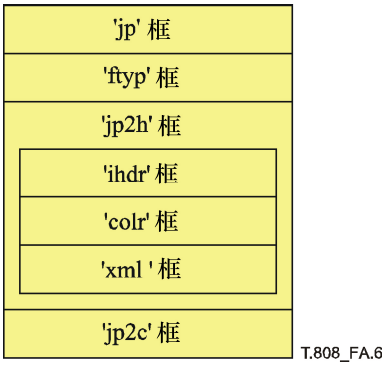

图 A.6－JP2 文件的例子

该文件可分成三个元数据块：一个表示原文件（数据块0）的顶层，一个表示JP2的头框，另一个表示码流。这种划分如图A.7所示。

任何元数据块的内容应该是它所表示的那个框或文件的内容，而包含在这些内容中的实际数据根据不同的框类型有概念性的变化。例如，图A.7中表示JP2的头框的内容的元数据块1中，由于JP2的头框是一个超级框，其内容是一系列完整的框的罗列。由于JP2的头框中没有其它的数据，因此，元数据块1中除了那些完整的框外，可能没有其它的数据。相反地，元数据块2中都是连续码流框的原始内容，没有框的头，这是由于那个框不是超级框。

图A.7这个例子中有一点特别引人注意，即有两种方式可以访问码流数据。第二个占位符块用于替代原文件中的连续的码流框（jp2c）。这说明元数据块2有该框的原始内容，也就是原始码流本身。为便于描述，建议书|国际标准中用“原始码流”这个术语表示。原始码流用于元数据块中。

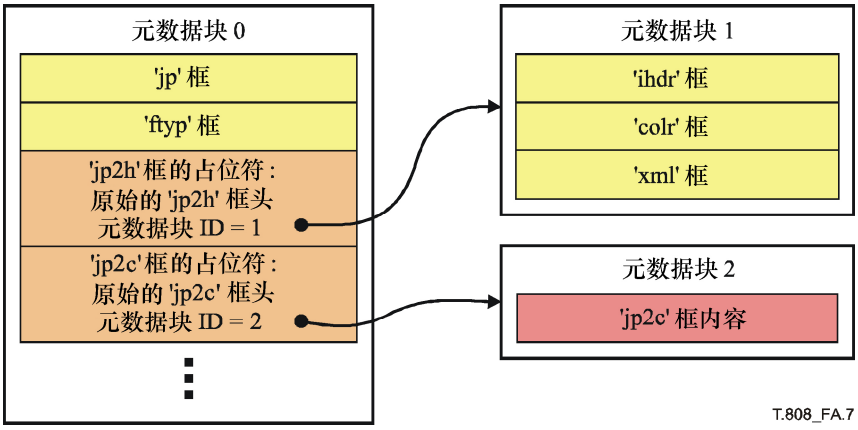

图 A.7－JP2 文件分割成三个元数据块的例子

占位符中也可能规定一个码流标识符。具有相同码流标识符的任何属于主头、分片头、分区和分片数据块类型的数据块，传送和元数据块2中同一个码流的压缩数据。为便于描述，本建议书|国际标准中，用“增量码流”这个术语表示。增量码流用于这些数据块中。

通常，占位符引用码流数据，可以通过引用一个单独的元数据块（原始码流），或规定一个码流标识（增量码流），或两者兼之。但即使使用了两种方式，客户端或图像显示代理中的JPP-stream或者JPT-stream数据只能有原始码流中的内容或者来自增量码流的数据。而且，如果同一个码流的原始和增量版本

都存在，并不能保证这两种表示具有一致的编码参数。只有和这两种表示相关联的重构图像样本，才能保证是一致的。

也可能用占位符框关联一个原始框中的多个码流。对这种关联的解释取决于被替代的框。关于这种情况的进一步讨论见A.3.6.4。

在图A.7这个简单的例子中，占位符框仅出现在文件的顶层，即元数据块0中。但前面已提到，占位符可用于替代任何元数据块中的任何框。这允许以分级的形式分解复杂文件。因此，一个源文件可根据占位符框的不同使用，封装成多种不同的元数据块结构。**但是，一个JPP-stream或JPT-stream 只能采取其中一种封装。**在客户端－服务器应用中，一般由服务器为文件确定一种合适的元数据块结构，给相应形成的流分配唯一的标识符，并在引用该唯一标识符的所有客户端的所有通信中使用相同的元数据块结构。

当占位符将一个框重新部署到新的元数据块中时，该框的头（Lbox，TBox和XLBox域）存储在占位符框中，不作任何修改。如果客户端或显示代理需要对应某些框相对它们源文件的偏移量，可使用占位符框中的原始框的头。该信息最终可以将源文件中的任何位置映射到特定元数据块中的特定位置，如果那个数据块的内容存在的话。这是很重要的，由于某些JPEG 2000系列文件包含引用其它框的框，通过它们在文件中的位置。

虽然确定如何以最好的方式将一个文件分割为数据块是相当自由的，但有一个约束条件。元数据块中的任何占位符框应替代该数据块中的顶层框。也就是说，如果一个子框被占位符替代了，则直接包含它的超级框应位于它自己的元数据块中。例如，图A.6所示的文件例子中，包含在JP2头框中的XML数据可能放在和其它框分开的数据块中。这允许服务器仅传送那些实际被请求用于图像的解码和显示的数据块，除非XML数据被明确请求。图A.8显示了一种适当的数据块结构。

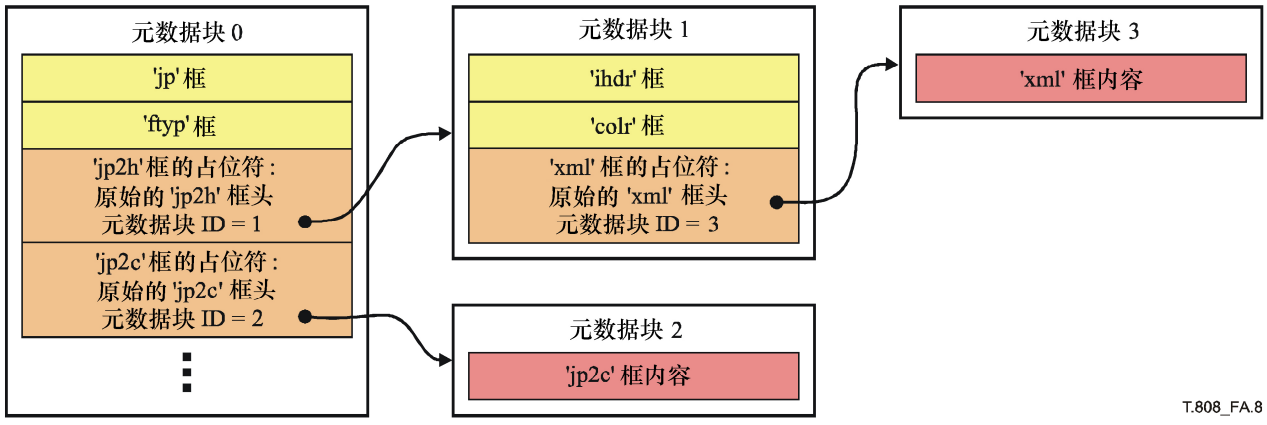

图 A.8－包含被引用的数据块的超级框

然而图A.9所示的将JP2头框放在元数据块0中是非法的：

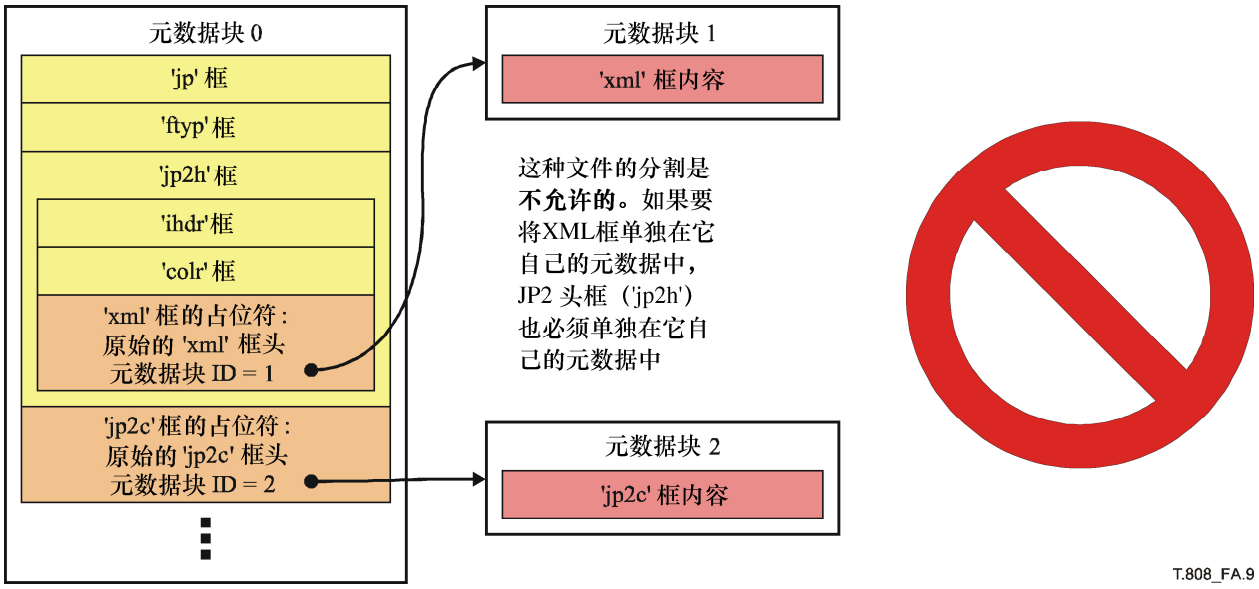

图 A.9－文件非法分割成数据块

注 — 这种限制条件可等价于如下表述。每当占位符替代了一个子框，则应有一个占位符框替代包含它的框。这种限制确保客户端或显示代理即使不理解某些框，仍能获得文件中原始框的长度和位置。

除了在单独的元数据块中提供框的原始内容外，JPP-和JPT-stream还允许提供框的其它表示方法，这种框在源文件中不明确出现。这些表示方法被称为“流等价”。例如，源文件可能包含一个交叉引用框，这个框的分片段列表框中收集了文件的一个或多个用于重组色彩空间规范框的分片段。虽然客户端或显示代理应能够根据相关的文件指针重新组建彩色空间规范框，但一种更方便的JPP-或JPT-stream表示方法可能是用一个占位符作为流等价，该占位符引用含有被重组的彩色空间规范框的数据块。为此，占位符需包括流等价的框头，以及容纳流等价框内容的元数据块的标识符。

下面的例子（如图A.10所示）说明了交叉引用框的流等价的使用。在这个情况中，容纳流等价内容的数据块也被引用作为容纳另外框的原始内容。然而在源文件包含交叉参考框的地方，这是一种普遍的情景，流等价不必指向一个连接到源文件层次的元数据块。流等价框的内容可能拼凑而成或者引用其它文件中的原来存在的内容。这允许分片段列表引用其它文件或者URL的交叉引用框，能全部封装在一个JPP-或JPT-stream中。

流等价可用在服务器能为了客户端的利益创建其它框内容的格式的任何情况中，并不仅仅提供到明确交叉引用的数据的接入。

除了指向实际的或等价的框数据外，占位符框还能指向一个或多个码流，其中被替代的框等价于那些码流。例如，连续码流框可能被占位符框替代，该占位符框引用那个连续码流框中的增量码流的ID。另一个例子是用定义一组码流ID的占位符框替代运动的JPEG 2000文件中的组块偏移量框。那些码流ID引用组块偏移量所指向的码流。

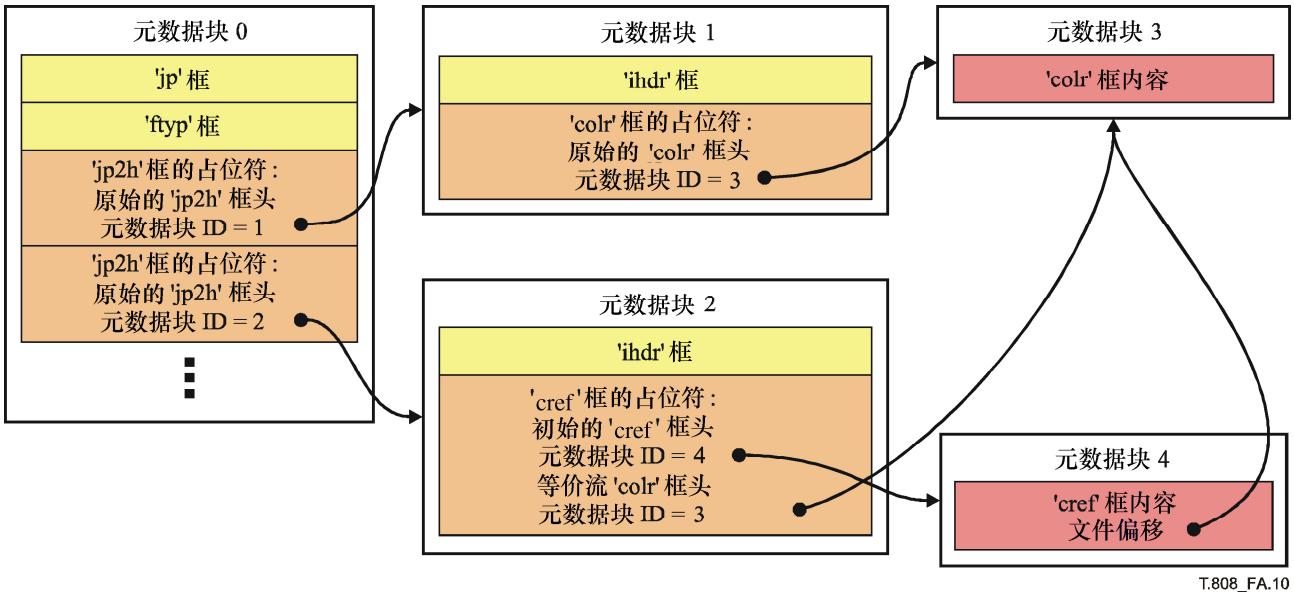

图 A.10－应用流等价的例子

A.3.6.3 占位符框的格式

图A.11显示了占位符框的格式，包括框头（和附件I以及本建议书|国际标准其它部分中大多数框的定义不同）。这种定义方式是为了强调占位符框的头中长度域的使用比其它框更严格。

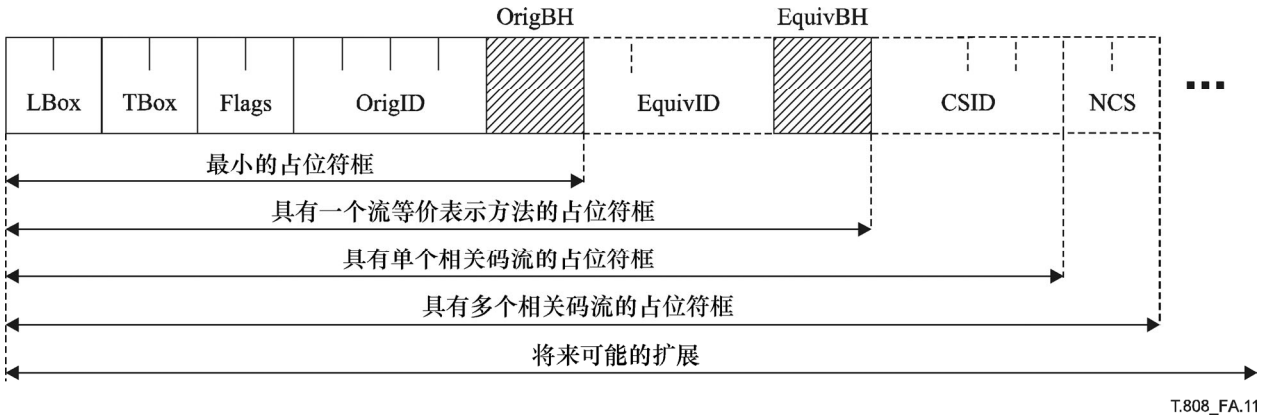

图 A.11－占位符框的结构

Lbox：这是对于框的标准的按big-endian字节顺序存储的4字节长度的域。占位符框的这个值不能为1，表示不应出现XLBox域。

Tbox：这是对于框的标准的4字节的框类型域。占位符框的类型值为'phld' (0x7068 6c64)。

Flags：这个域规定了占位符框的什么元素包含有效数据。这个域是4字节按big-endian字节顺序存储的整数，Flags域的合法值在表A.3中定义。

OrigID：这个域规定了本占位符引用的包含原始框的内容的元数据块ID。本域是一个8字节按big-endian字节顺序存储的无符号整数。

OrigBH：这个域规定了本占位符框引用的原始框的头（LBox,TBox以及XLBox（如果有））。如果原始框的头的Lbox域不等于1，本域的长度是8字节，否则为16字节。

EquivID：这个域规定了包含流等价格式的该框的内容的元数据块ID。本域是一个8字节按big-endian字节顺序存储的无符号整数。

EquivBH：这个域规定了本占位符引用的流等价框的头（LBox，TBox，以及XLBox（如果有））。如果等价框头的Lbox域不等于1，本域的长度是8字节，否则为16字节。

CSID：这个域规定了和被替代的框关联的第一码流的ID。这个ID和所有的头，以及用于增加传达和被替代的框关联的第一个码流的内容的区域和/或分片数据块相关联。本域是一个8字节按big-endian字节顺序存储的无符号整数。

NCS：这个域规定了和被替代的框等价的码流组中的码流的号码。这些码流的ID值从CSID域规定的值依次顺下去。本域是一个4字节按big-endian字节顺序存储的无符号整数。

ExtendedBoxList：这个域在图A.11中未明确显示。NCS域后可跟一系列来自服务器的包含扩展消息的框。跟在NCS后的任何框的存在，应通过Flags域中的一个比特进行定义。但本建议书|国际标准既没有定义扩展框也没有定义额外的比特标记。客户端应忽略任何在ExtendedBoxList中不理解的框。

表A.3中“x”值表示已定义的值，包括该比特被设置成“1”或“0”的情况。“y”值表示本标准未使用的比特，服务器应将其设置为“0”，客户端应忽略之。

并不是所有为占位符框中定义的域必须都在每一个占位符框中出现。如图A.11中箭头所表示的，如果没有等价或者增量码流ID，框可在OrigBH域后结束。同样地，如果没有增量码流ID，框可在EquivBH域后结束，如果只有一个增量码流ID，框可在CSID域后结束。

表 A.3－占位符框中 Flags 域的合法值

值	意义
yyyy yyyy yyyy yyyy yyyy yyyy yyyy xxx1	通过 OrigID 域定义的元数据块可访问这个框的原始内容。
yyyy yyyy yyyy yyyy yyyy yyyy yyyy xxx0	不能访问这个框的原始内容，且应忽略 OrigID 域的值。
yyyy yyyy yyyy yyyy yyyy yyyy yyyy xx1x	有流等价框，其内容在 EquivID 域定义的元数据块中。
yyyy yyyy yyyy yyyy yyyy yyyy yyyy xx0x	没有流等价框，应忽略 EquivID 和 EquivBH 域中的值。
yyyy yyyy yyyy yyyy yyyy yyyy yyyy 01xx	通过 CSID 域定义的单个增量码流访问这个框表示的图像。NCS 域的值应当做“1”处理而不管本域的实际值为多少。
yyyy yyyy yyyy yyyy yyyy yyyy yyyy 11xx	通过 CSID 和 NCS 域定义的一个或多个增量码流访问这个框表示的图像。
yyyy yyyy yyyy yyyy yyyy yyyy yyyy x0xx	本占位符不提供访问用增量码流表示原始框的图像，应忽略 CSID 和 NCS 域。
其它值	保留给 ISO 用。

A.3.6.4 用占位符对增量码流的引用

凡是有头、分区和/或分片数据块的地方，它们的码流ID应出现在适当元数据块的占位符框中。这一要求仅对于解开的JPEG2000码流是一个例外，这种码流不属于JPEG 2000系列文件格式。

出现在相应占位符框中的码流ID值应符合对所包含的文件格式的任何要求。例如，JPX文件形式上通过连续码流框或者片断表框给出现在文件顶层的每个码流分配一个序列号。逻辑对象中的第一个顶层码流的码流ID应为0，接着应为1，依次类推。

引用多个码流ID的占位符可能仅用在那些码流的意义通过所替代的框的类型中已明确定义的情况。

A.3.6.5 MJ2中使用占位符框

本建议书|国际标准仅定义了两种在运动的JPEG 2000（MJ2）文件中适用占位符的框类型。具体地，只有组组块偏移量框('stco')或组块大偏移量框('co64')可被识别多个码流ID的占位符框替代。

MJ2文件中的每个视频轨迹严格的包含一个组块偏移量框（'stco'或'co64'），与组块框的样本一起，用于标识属于该视频轨迹的所有连续码流框的位置。如果组块偏移量框被一个提供一个或多个码流ID的占位符替代，则视频轨迹中的每个连续的码流框需严格具有一个码流ID。如果视觉样本实体框（'mjp2'）规定了一个以2计算的域，那么占位符框应提供2*N*个码流ID，其中N为视频样本的数量（即N为帧数）。否则，占位符框应只提供*N*个码流ID。码流ID应按样本号（帧号）和每个样本中的域号码的顺序编号。

注 — 对于用JPP-stream 或 JPT-stream表示的MJ2文件，流中没有必要包含原始组块偏移量框、组块框('stsc')的样本或者抽样尺寸框('stsz') 的内容。如果流表示被转换成MJ2文件时，若需要，能重新生成这个索引信息。

A.4 解析和传送JPP-stream和JPT-stream的约定（资料性）

占位符框给客户端和服务器在如何分析或传送JPP-和JPT-stream时增加了灵活性以及某些潜在的含糊性。服务器可以选择使用很多策略中的任何一种，通过在合适的点引入占位符框，将JPEG 2000系列文件中的原始框分割成元数据块。服务器应用一致的方式进行分割，使得对于所有访问相同逻辑对象（可能通过一个唯一的对象ID标识）的客户端，每当访问时，和某个JPP-或JPT-stream关联的数据块具有相同的名字的内容。

然而更重要的是，占位符允许服务器构建单个JPP-或JPT-stream，其数据块提供相同原始内容的多种替代的表示方法。当占位符中有流等价，和/或当占位符中有增量码流ID时，就会出现这种情况。在这些情况下，原始框可能出现在元数据块中，也可能出现在另一个元数据块的流等价中，和/或出现在头、分区或者分片数据块表示的增量码流中。服务器可能分配表示一个原始框的所有数据块的内容，但为高效起见，服务器可能只分配足量的信息来传送原始内容，除非明确要求分发冗余的数据块。客户端侧JPP-或JPT-stream的解析器，当遇到一个原始框的多种表示方法时，可能只选择一种表示，忽略所有其它的。理想的客户端的规约应对服务器实际选择哪个数据块发送给客户端有重要的影响。

鉴于此，本建议书|国际标准建议如下的规约：

— 除非有可信任的理由，否则当流等价框和原始框这两种框类型都通过占位符告知给客户端时，服务器应假定客户端优先解析流等价框。

— 除非有可信任的理由，否则当增量码流和原始码流这两种框类型都通过占位符告知给客户端时，服务器应假定客户端优先用增量码流表示方法（头、分区或分片数据块）解析。

A.5 JPP-stream或JPT-stream互操作性的约定（资料性）

本规约描述了JPP-stream和JPT-stream的文件格式的交换，这里分别称为jpp-文件和jpt-文件。这种文件可能包含JPIP会话中接收到的JPEG 2000数据（如客户端的缓存），或其中的子集。由于JPP-stream和JPT-stream都是自描述媒体类型，另一个JPIP客户端读取和使用这个文件是可能的。

这些文件由JPT-stream 或JPP-stream消息串联而成。例如，它们可能通过将客户端从单个或多个会话中接收的所有这种消息简单的串联形成。一种改进的情况是客户端产生一个逻辑JPT-stream 或JPP-stream，每个数据块用一个消息头和消息表示。

建议这些文件用".jpp" 和".jpt"扩展名，且若合适，文件名包括对相关的JPIP `target`令牌或`target-id`令牌的引用。

本规约没有规定客户端缓存的实现或结构。例如，客户端可以使用一个数据库作为缓存功能的实现，而不是基于文件的缓存系统。

附 件 B

会话、通道、缓存模型和模型组

（本附件是本建议书|国际标准的组成部分）

B.1 会话中的请求和无状态请求

JPIP协议明确区分两种类型的请求：无状态请求和属于一个会话的请求。

会话的目的是为了减少客户端和服务器之间所需的显式通信。在一个会话中，服务器能记住前面请求中提供的客户端的能力和参数选择，使这些信息不需要在每个请求中都发送。更重要的是，服务器一般会维持一个关于它在对前面请求的响应中已发送给客户端的信息的日志，从而在对后来的请求的响应中不必再发送这些信息。会话过程中日志一直存在。除非有明确的命令，否则服务器认为客户端缓存着一个会话中的所有请求的响应，并建立客户端缓存的模型，仅发送那些客户端缓存中没有的压缩图像数据或者元数据的部分。

无状态请求不和任何会话关联，因此是完全独立的。应注意，“无状态”这个词仅适用于服务器，而不是客户端。就会话而言，客户端通常应缓存和同一个逻辑对象相关联的前面请求的响应。发出多个无状态请求给同一个对象的客户端通常应在每个请求中包括它们的缓存内容的信息，防止传送冗余数据。因此，会话的优点是更小更简单的请求和/或从服务器得到更少冗余的应答数据。无状态通信的优点是服务器不必保存请求之间的状态信息，这意味着同一个主机不必完全为从单个客户端发出的对同一个对象图像的所有请求进行服务。

B.2 通道和会话

和每个会话相关联的要素如下：

— 一个或多个逻辑对象（通常是图像文件），其内容在会话中不变。

— 和会话相关联的每个逻辑对象的一个图像数据返回类型。

— 若数据返回类型是"jpp-stream"或"jpt-stream"时，和会话相关联的每个逻辑对象应维持一个客户端缓存的模型。但注意，这个模型不一定完全反映客户端缓存的实际状态。管理缓存模型的规则在B.3中概述。

— 一个或多个JPIP通道。客户端通常可能在同一个会话中打开多个通道。每个JPIP通道可能和底层一个独立的传输通道（如一个独立的TCP连接）相关联，也可能不是。多个通道允许客户端可以同时发送对多个图像区域的请求，并期望服务器能同时响应这些请求。多个通道也允许在同一个目标图像内或多个对象间的不同类型的请求之间进行智能的带宽分配。

— 当多个通道和同一个逻辑对象相关联时，会话模型适用于所有的通道。尽管如果多个通道涉及同一个逻辑对象可能会有不好的影响，但在同一个会话中，仍可能有多个客户端打开多个JPIP通道。

和每个通道相关的要素如下：

— 单个逻辑对象（通常是图像文件）。

— 每个请求中应包括一个服务器分配的标识符。由于通道标识符足以将请求和它所在的会话关联起来，JPIP未定义单独的会话标识符。

— 客户端的能力和偏好的记录，该记录可以通过适当的请求域进行调整。

— 在服务器队列请求方面，应为每个JPIP通道提供单独的队列。

一个通道上的客户端请求和客户端响应是一对一的通信。不同的JPIP通道可能承载在同一个传输通道或者不同的传输通道上。如果用单独的传输通道传送请求，则用不同的JPIP通道的请求可能异步的到达服

务器。对多个通道如何提供服务主要取决于服务器的判断，而传送等级请求域、最大带宽以及带宽切分片偏好，应指导服务器。

B.3 缓存模型管理

如前所述，会话的一个基本功能就是服务器侧对客户端缓存的建模。除非有明确的告知，否则服务可能认为客户端缓存了一个会话中已发送的对请求的响应的所有信息：这些信息不必重传。但注意，服务器不一定要维持一个完整的甚至任何缓存模型：可能对请求传送冗余数据。

除了影响传送数据外，客户端请求中显式的缓存模型的操作声明可以更新服务器的缓存模型。在确定对客户端响应时应回送的数据前，先处理这些声明。缓存模型的操作声明有两种：添加和删除。

添加缓存模型操作声明用于增加服务器的缓存模型，在现存的模型中增加数据块或部分数据块。这为客户端提供了一种机制，告知服务器它在前面的会话中或者使用前面的无状态请求接收到的信息。服务器应尝试利用客户端请求中的任何添加缓存模型操作声明。但服务器不一定要维持一个完整的缓存模型，因此，服务器可以忽略或者部分忽略添加缓存模型的操作声明。

删除声明用于从服务器缓存模型中移去数据块或者部分数据块。客户端为了告知服务器它没有缓存或丢弃了某些服务器前面传送的数据，可发送删除缓存模型的操作声明。否则服务器认为客户端已经缓存了所有会话过程中发送的数据。服务器应在其所维持的任何缓存模型（完整的或其它的）中移去减除缓存模型声明中提到的所有信息。

基于会话的JPIP请求具有侧面的影响，即可能影响对未来请求的响应。这对于包含缓存模型的操作声明的请求来说也是正确的—缓存模型操作的影响是永久的。而且，一个JPIP通道上到达的请求，会侧面影响到对和同一个逻辑对象相关的属于不同JPIP通道上的所有请求的响应。这是基于会话中每个逻辑对象只有一个缓存模型的事实上的。

B.4 模型组的询问和操作

当和会话相关联的逻辑对象包含大量码流（如视频对象），或客户端保持长时间的连接时，对于实际的服务器实现来说，部分缓存建模逐渐变得具有策略性，客户端也逐渐不能缓存服务器发送的所有信息。为了避免这种情况下通信的低效，引入"mset"（模型组）的概念。"mset"是服务器模拟的客户端缓存中内容的码流的集合。

在任何请求中，客户端可指示服务器将其"mset"限制为一定组的码流。这为客户端提供了一种方便的机制，客户端丢弃缓存中的所有码流后，不用担心服务器对未来那些码流的请求产生不完整的响应。

对"mset"请求的服务器的响应指示了缓存模型维持着信息的实际的码流组。这使客户端能确定涉及各种码流的缓存模型的操作声明是否被服务器忽略。

在没有任何显式的"mset"操作或询问时，客户端可仅认为服务器的"mset"包括为它的请求产生的响应数据的所有码流。由于服务器通常有权利将客户端的请求限制在比原先指定的更少的码流中，因此不保证服务器的"mset"包括请求中涉及的所有码流，除非请求只涉及一个码流。这些问题将在C.8.6中进一步解释。

附 件 C

客户端请求

（本附件是本建议书|国际标准的组成部分）

C.1 请求语法

C.1.1 引言

本附件介绍了JPIP请求中所有可能的要素。每节都介绍了一组域以及这些域的可能的取值。通常，一个请求由多组域组成，但有些组之间是不兼容的。甚至在每一个组内，有些请求域之间也是不兼容的。一些其它方面合法的请求在某些情况（如会话）中是不合法的，尽管BNF语法中并没有指出。最后，即使一个合法的请求，服务器可能不支持所有的请求域或者它们的组合。

C.1.2 请求结构

JPIP请求由以下域组成：

— 对象标识域；
— 会话和通道管理域；
— 视窗请求域；
— 元数据域；
— 数据限制请求域；
— 服务器控制请求域；
— 缓存管理请求域；
— 上传请求域；
— 客户端能力和偏好域。

请求中要素的发送要遵从所选择的传输协议。例如，HTTP中，用BNF语法中列出的字符表达请求，用"&"字符连接多个参数，且请求可能是GET请求中查询域的一部分，或者是POST请求的主体。具体见附件F、G和H。

注 — 应避开URI的保留字符。例如，HTTP GET URL中的"request=a:b"，应表达为"request=a%3Ab"，其中':'是URL的保留字符，用'%3A'避开。

```
jpip-request-field = target-field
                   / channel-field
                   / view-window-field
                   / metadata-field
                   / data-limit-field
                   / server-control-field
                   / cache-management-field
                   / upload-field
                   / client-cap-pref-field
target-field       = target                    ; C.2.2
                   / subtarget                 ; C.2.3
                   / tid                       ; C.2.4
channel-field      = cid                       ; C.3.2
                   / cnew                      ; C.3.3
                   / cclose                    ; C.3.4
                   / qid                       ; C.3.5
```

```
view-window-field  = fsiz                    ; C.4.2
                   / roff                    ; C.4.3
                   / rsiz                    ; C.4.4
                   / comps                   ; C.4.5
                   / stream                  ; C.4.6
                   / context                 ; C.4.7
                   / srate                   ; C.4.8
                   / roi                     ; C.4.9
                   / layers                  ; C.4.10
metadata-field     = metareq                 ; C.5.2
data-limit-field   = len                     ; C.6.1
                   / quality                 ; C.6.2
server-control-field = align                 ; C.7.1
                   / wait                    ; C.7.2
                   / type                    ; C.7.3
                   / drate                   ; C.7.4
cache-management-field = model               ; C.8.1
                   / tpmodel                 ; C.8.3
                   / need                    ; C.8.4
                   / tpneed                  ; C.8.5
                   / mset                    ; C.8.6
upload-field       = upload                  ; C.9.1
client-cap-pref-field = cap                  ; C.10.1
                   / pref                    ; C.10.2
                   / csf                     ; C.10.3
```

C.1.3 请求域组合的限制

在一个请求中，每一类JPIP请求域不应出现多于一次。

通常，对图像数据的请求（视窗请求）可以和其它元数据的请求组合。但如何组合请求域有一些限制。

上传请求域不应和metadata-field、data-limit-field或者server-control-field组合。

C.2 对象标识域

C.2.1 逻辑对象的介绍

每个JPIP请求都是指向某个初始命名的资源或者该资源的某一部分的某种表示。这种资源可能是一个物理存储着的文件或者对象，也可能是服务器为请求实际创建的资源。

某种表示，不管是原始的编码格式还是转换码格式，是资源的某个字节范围还是整个资源，都被称为逻辑对象。逻辑对象通过三个请求域规定：对象ID、对象和子对象。

对象请求域规定了请求所指向的初始命名的资源。用PATH进行规定，其中PATH是一个简单的串或者URI。如果未定义对象域，且请求承载在HTTP上，JPIP请求应指向JPIP的请求URL中路径分量所规定的资源。这个初始命名的资源可能是一个物理存储在服务器上的实际的文件或者其它对象，也可能是服务器响应JPIP请求而创建的资源。

子对象请求域规定了请求所指向的初始命名资源（通过对象请求域规定）的某个字节范围。如果未定义子对象请求域，则请求指向整个原始资源。

对象ID请求域用于在客户端和服务器以前交换过该资源的数据的情况下，进一步规定资源的某种编码。例如，服务器可能基于提供的信息和以前请求的环境，给客户端提供过文件的转换码版本。如果那个

客户端在缓存中保存了以前传送的数据，则它将希望继续接收使用相同转换码的数据，从而能继续使用缓存中的数据。对象ID是服务器定义的标识串，标识以前和服务器相关联的那个初始命名资源的某个表示或者某个字节范围。

如果客户端同时规定了初始命名的资源（通过对象请求域或者JPIP请求的URL的路径分量）和Target-ID，服务器应验证它是否能以原先给那个资源指派Target-ID的相同的方式响应该请求。如果服务器不能用相同的方式进行响应，它应使用JPIP-tid响应头将新的Target-ID通知客户端，这种情况下客户端就知道它必须丢弃任何以前缓存的数据。

如果用JPP-stream或JPT-stream消息服务逻辑对象，则和一个会话中产生所有响应相关联的数据块应保持一致。当服务器或者一个相关的服务器还产生了一个对象ID时，在和所有产生的具有相同对象ID的响应相关联的数据块应保持一致，不管这些响应是否在同一个会话中产生。

如果该请求是会话的一部分，且服务器分配了一个通道ID，客户端可通过通道ID请求域规定通道ID，而不是通过对象、子对象和对象ID请求域。如果逻辑对象既通过对象、子对象和对象ID域的组合规定，又通过通道ID请求域规定，则服务器应响应错误。

以下例子显示了逻辑对象的定义：

例1：JPIP请求的URL为

"http://one.jpeg.org/imageserver.cgi?target=http ％ 3A ％ 2F ％ 2Fone.jpeg.org ％ 2Fimages ％ 2Fpicture.jp2&fsiz=200,200"

则逻辑对象是包含在服务器文档的根目录下的URI为“http://one.jpeg.org/images/picture.jp2,”内的所有字节。

例2：JPIP请求的URL为

"http://one.jpeg.org/imageserver.cgi?target=http ％ 3A ％ 2F ％ 2Fone.jpeg.org ％ 2Fimages ％ 2Fpicture.jp2&tid=4384-5849-af4d-3dca&fsiz=200,200"

则逻辑对象是包含在服务器文档的根目录下的URI为“http://one.jpeg.org/images/picture.jp2,”内的所有字节，且使用服务器定义的表示方法对象ID为4384-5849-af4d-3dca。

例3：JPIP请求的URL为

"http://one.jpeg.org/imageserver.cgi?target= http ％ 3A ％ 2F ％ 2Fone.jpeg.org ％ 2Fimages ％ 2Fpicture.jp2&subtarget=1038-13458&fsiz=200,200"

则逻辑对象是包含在服务器文档的根目录下的URI为“http://one.jpeg.org/images/picture.jp2,”内的部分字节，从1038字节开始到并包括13458字节。

例4：JPIP请求的URL为

"http://one.jpeg.org/imageserver.cgi?cid=1234-5849-af4d-3dca&fsiz=200,200"

则逻辑对象是服务器将其和通道ID为1234-5849-af4d-3dca相关联的资源。

例5：JPIP请求的URL为

"http://one.jpeg.org/images/picture.jp2?fsiz=200,200"

则逻辑对象是服务器文档的根目录下文件"images/picture.jp2"的所有字节。

例6：JPIP请求的URL为

"http://one.jpeg.org/images/picture.jp2?subtarget=1038-13458&fsiz=200,200"

则逻辑对象是服务器文档的根目录下文件"images/picture.jp2"的部分字节，从1038字节开始到并包括13458字节。

C.2.2 对象(target)

```
target = "target" "=" PATH
```

本域用于规定初始命名的资源（通常是服务器上文件的名字）。如果没有对象请求域，则初始命名资源通过其它方式确定。

C.2.3 子对象 (subtarget)

```
subtarget = "subtarget" "=" byte-range
byte-range = UINT-RANGE
```

本域可通过规定字节范围限制初始命名的资源。逻辑对象被解释为初始命名的资源的本域所指定的字节范围。

包含字节范围的上下界限，0表示目标文件的第一个字节。

C.2.4 对象ID (tid)

```
tid = "tid" "=" target-id

target-id = TOKEN
```

本域用于提供一个target-id串，该串是服务器以前产生的，用来完全标识被访问的逻辑对象，该串包括服务器实现的任何转换编码。逻辑对象的名字不一定是唯一的，且其内容不必对应一种编码。但target-id串和初始命名资源以及字节范围结合，应完全标识图象和它的编码。

如果target-id等于“0”，则逻辑对象通过对象、子对象以及JPIP URL的路径分量规定，且如果存在target-id客户端明确请求服务器告知其分配的target-id。服务器应在对客户端请求中包含target-id等于"0"的所有响应中包括一个对象ID。

target-id的长度不应超过255个字符。

C.3 和会话及通道相关的域

C.3.1 引言

一个请求是无状态的，除非当以下条件中的一个或两个都满足：

— 请求中包括一个有效的通道ID域；

— 请求中包括一个新建通道域（见下面），且服务器响应中包括一个具有新产生的channel-id 的新建通道的响应头。

关于会话和通道的讨论见B.2。

C.3.2 通道ID (cid)

```
cid = "cid" "=" channel-id

channel-id = TOKEN
```

— 本域用于将请求和某个JPIP通道关联起来，从而将请求和通道所属的会话关联起来。

C.3.3 新建通道 (cnew)

```
cnew = "cnew" "=" 1#transport-name

transport-name = TOKEN
```

本域用于请求一个新的JPIP通道。如果没有通道ID请求域，则请求一个新的会话。否则，请求和通道ID请求域所标识的通道在同一个会话中的新的通道。

串值表示客户端想要接受的一个或多个传输协议的名字。本建议书|国际标准仅定义了"http"和"http-tcp"传输协议名，尽管在其它地方可以任意定义其它的传输协议，如"udp"。在"http"传输上使用JPIP具体见附件F，而在"http-tcp"传输上使用JPIP具体见附件G。

如果服务器愿意用所指示的传输协议之一打开一个新的通道，它应使用新建通道响应头（见D.2.3）返回新通道的标识符令牌。这种情况下，当前的请求是这个新通道的第一个请求。

对于客户端来说，在同一个会话中打开到一个新的逻辑对象的通道是可能的。为此，客户端的请求应同时标识现有的通道ID和逻辑对象。当打开和现有的通道相关联的同一个逻辑对象的新通道时，不必明确规定逻辑对象。

如果服务器不愿打开一个新的通道，它不应返回新建通道的响应头，但应视同不包括新建通道请求域一样为该请求提供服务。这表示规定了现存通道ID的请求应作为该通道中的请求处理，而不包括通道ID请求域的请求应作为无状态请求处理。在新建通道请求中标识的逻辑对象和所提供的现存的通道ID关联的逻辑对象不同的情况下，服务器对请求的响应中应包含一个新的通道ID或者返回一个错误码。

例1："target=nice.jp2&cnew=http"，请求到图像"nice.jp2"的一个新的会话的第一个通道，请求用"http"传输。如果服务器不分配通道，该请求将视为无状态处理。

例2："cid=013ac8&cnew=http-tcp"，在和通道ID为013ac8相关联的同一个会话中请求一个新的通道。新通道用"http-tcp"传输，且和通道ID 013ac8指向同一个逻辑对象。这些通道共享一个缓存模型。如果服务器不分配通道，该请求视同忽略新建通道请求域处理。

例3："target=nice.jp2&cid=013ac8&cnew=http"，在和通道ID为013ac8相关联的同一个会话中请求一个新的通道。新的通道用"http"传输。和新的通道相关联的逻辑对象不同于和通道ID为013ac8相关联的逻辑对象，新通道使用一个单独的缓存模型。两个对象的缓存模型和同一个会话相关联。

C.3.4 关闭通道 (cclose)

```
cclose = "cclose" "=" ("*" / 1#channel-id)
```

本域用于关闭一个会话中的一个或多个通道。如果本域的值包含一个或多个channel-id令牌，这些通道应属于同一个会话。这种情况下，通道ID请求域不是必须的，但如果有，也应指示属于同一个会话的通道。

如果本域的值是"*"，则和会话相关的所有通道都将关闭。这种情况下，应在通道ID请求域中标识出该会话。

服务器应在实际关闭通道前在关闭通道请求所规定的通道上完成响应。

C.3.5 请求ID (qid)

```
qid = "qid" "=" UINT
```

本域用于规定请求ID的值。每个通道都有各自的请求队列，并有各自的请求ID计数器。在某个通道内接收到的请求（由请求ID值指示），当使用了请求ID域时，应按照请求ID值的顺序处理。对于不包含请求ID域的请求，服务器可以按先来先服务的原则进行处理。但对于和同一个通道相关联的请求，服务器只有处理完请求ID值小于n的所有请求后才能处理请求ID值为*n*的请求，除非*n*=0。客户端在同一个通道中不应生成具有相同请求ID值的请求，也不应生成比该通道中前面生成的请求ID值小的请求ID。

C.4 视窗请求域

C.4.1 将视窗请求映射为码流图像分辨率和区域

JPIP是为了响应来自客户端的请求，为其提供一个JPEG 2000图像的部分及相关的元数据。这通过一系列的请求和应答实现。对于图像的部分，被请求的数据在图像帧尺寸、区域、质量和/或分量方面可能比整个图像少。

在最简单的情况下，所涉及的图像的部分直接通过请求中标识的JPEG 2000码流的高分辨率参考网格定义，而不是特定的图像分量的抽样网格。而更一般地，客户端可能通过码流上下文请求域（见C.4.7）请求更高层的图像对象（如JPX合成图层或MJ2视频轨迹）。在这种情况下，为了确定每个被请求的相关码流的部分，被请求的图像的部分可能需要进行坐标变换。这些坐标变换在C.4.7中介绍，且应能根据随后的码流图像区域的描述理解这些坐标变换。

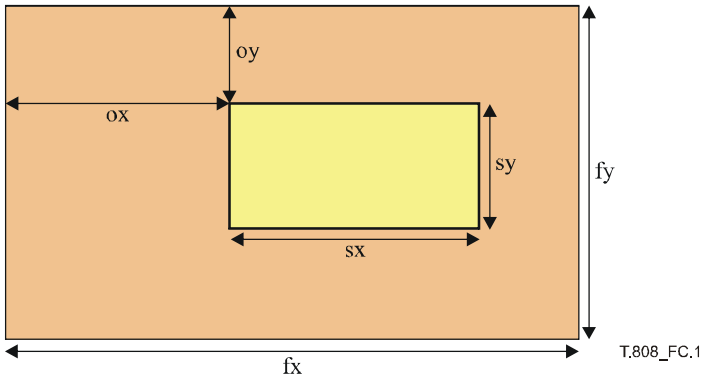

图 C.1－图像中想要的区域

码流图像区域通过3个二维参数进行描述，如图C.1所示。尺寸参数（sx和sy）和偏移量参数(ox和oy)规定了想要的码流图像区域的宽度和高度，以及该区域的左上角在帧尺寸(fx和fy)的整个图像中的位置。

例如：当客户端希望用整个图像填充一个640×480的显示时，可能产生下面的请求：

"fsiz=640,480&rsiz=640,480&roff=0,0"。注意不管图像的原始尺寸（甚至都不需知道图像的原始大小），都可以这样请求。

当JPEG 2000码流中没有可用的图像分辨率完全符合请求的帧尺寸时，返回的图像数据可能比请求的帧尺寸大或者小，甚至可能图像的高宽比都不同。服务器应确定一个合适的码流图像分辨率，由尺寸参数fx'和fy'表示，以及码流中的一个合适的区域，由参数sx'、sy'、ox'以及oy'表示，如图C.2所示。尽管客户端可作为帧尺寸请求域的一部分规定舍入的方向，但客户端应准备好处理返回的数据和请求的参数不完全符合的情况。

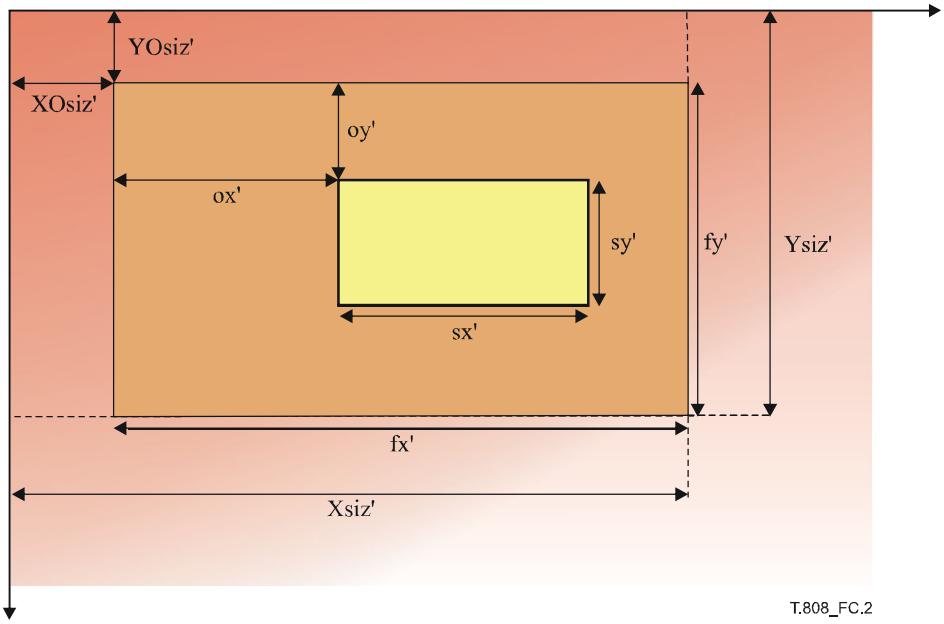

图 C.2－和二次抽样参考网格相关的想要的区域

如图C.2中所示，合适的码流图像分辨率的尺寸由fx'=Xsiz'-XOsiz'和fy'=Ysiz'-YOsiz'给出，其中XOsiz'、YOsiz'、 Xsiz'和Ysiz'根据等式C-1得到。

$$XOsiz' = \left\lceil \frac{XOsiz}{2^r} \right\rceil; \quad YOsiz' = \left\lceil \frac{YOsiz}{2^r} \right\rceil; \quad Xsiz' = \left\lceil \frac{Xsiz}{2^r} \right\rceil; \quad Ysiz' = \left\lceil \frac{Ysiz}{2^r} \right\rceil \tag{C-1}$$

其中：

r 是服务器根据帧尺寸请求域中给出的舍入参数，为和所请求的图像尺寸（fx 和fy）尽可能接近而确定的一个值。

这里，XOsiz、YOsiz、Xsiz 和 Ysiz从相关码流的SIZ标记段中取得。通常将*r*解释为舍去的最高DWT级数，且*r* 是一个不小于0的整数。但*r*的值并不仅限于用于压缩码流中的分片分量的DWT的级数。

一旦找到合适的帧尺寸fx'和fy'，和码流图像区域相关的区域尺寸sx'、sy'和偏移量ox'、oy'由等式C-2确定。

$$ox' = \left\lfloor ox \cdot \frac{fx'}{fx} \right\rfloor; \quad oy' = \left\lfloor oy \cdot \frac{fy'}{fy} \right\rfloor; \quad sx' = \left\lceil (sx+ox) \cdot \frac{fx'}{fx} \right\rceil - ox'; \quad sy' = \left\lceil (sy+oy) \cdot \frac{fy'}{fy} \right\rceil - oy' \tag{C-2}$$

例2：假设请求的帧尺寸是128 × 128，码流的高分辨率参考网格上的图像描述为XOsiz=127、Xsiz=648、YOsiz=0 和Ysiz=504。设码流中所有图像分量具有3级小波变换。则可能的码流图像的尺寸为：

$$\begin{aligned}
521\times504 \quad & \left(\left\lceil \frac{648}{1} \right\rceil - \left\lceil \frac{127}{1} \right\rceil \text{ by } \left\lceil \frac{504}{1} \right\rceil - 0 \right) \\
260\times252 \quad & \left(\left\lceil \frac{648}{2} \right\rceil - \left\lceil \frac{127}{2} \right\rceil \text{ by } \left\lceil \frac{504}{2} \right\rceil - 0 \right) \\
130\times126 \quad & \left(\left\lceil \frac{648}{4} \right\rceil - \left\lceil \frac{127}{4} \right\rceil \text{ by } \left\lceil \frac{504}{4} \right\rceil - 0 \right) \\
65\times63 \quad & \left(\left\lceil \frac{648}{8} \right\rceil - \left\lceil \frac{127}{8} \right\rceil \text{ by } \left\lceil \frac{504}{8} \right\rceil - 0 \right)
\end{aligned}$$

因此如果请求的是一个较大的帧尺寸（舍入方向是向上舍入），则返回的帧尺寸是260 × 252。如果请求的是一个较小的帧尺寸（舍入方向是向下舍入），则返回的帧尺寸是65 × 63。注意，在本例中，可能的码流图像帧尺寸一般不刚好是2的幂次方。

图像分量的二次抽样，由XRsiz 和Yrsiz规定，不影响任何被请求的码流中的请求的图像区域或图像分辨率的解释。

例3：请求一个512 × 512图像的左上角的256 × 256的区域，可表示为fsiz=512,512&rsiz=256,256。

假设码流包含图像的分量1和2的二次抽样，而未包含分量0的二次抽样。具体值设为：Xsiz=1024, Ysiz=1024, XOsiz=0, YOsiz=0, and $XRsiz^0$=1, $YRsiz^0$=1, $XRsiz^1$=2, $YRsiz^1$=2, $XRsiz^2$=2, $YRsiz^2$=2。服务器不考虑所有三个分量中的最高分辨率级别，返回适当的分片或分区用以提供分量0的256 × 256 个样本，和分量1和2的128 × 128个样本。这样客户端具有显示整个图像左上角一半尺寸的数据，且仍被二次抽样了。如果客户端想要显示无二次抽样的色彩浓度分量，可以再发出的请求，如：

fsiz=1024,1024&rsiz=512,512&comps=1,2

服务器则会返回适当的数据提供分量1和2的256×256个样本，这和已经收到分量0的数据结合起来，可获得一个无二次抽样的一半尺寸大小的图像。

由于图像分辨率和图像区域根据每个请求的码流的参考栅格确定，如果所有三个分量都被二次抽样了，则服务器将对初始请求（fsiz=512，512&rsiz=256，256）仅提供所有三个分量的128×128个样本。

C.4.2 帧尺寸（fsiz）

```
fsiz = "fsiz" "=" fx "," fy ["," round-direction]

fx = UINT

fy = UINT

round-direction = "round-up" / "round-down" / "closest"
```

本域用于标识和请求的视窗相关联的分辨率。fx 和fy规定了想要的图像分辨率的尺寸。如果被请求的图像分辨率在那个被请求的码流中不存在时，round-direction值规定了如何为其选择可用的码流图像分辨率。依据C.4.1中所述的流程，请求的帧尺寸被映射为码流图像分辨率，另外可能还需根据码流上下文请求域（见C.4.7）中的请求作坐标变换。客户端若希望控制从某个图像分量接收到的精确的样本数量，可能必须增加请求的帧尺寸，如C.4.1中说明。round-direction的可选值在本建议书|国际标准的表C.1中定义。

表 C.1－舍入方向的可选值

舍入方向	意义
"round-up"	对于每个请求的码流，应选择宽和高都大于或等于规定尺寸的最小码流图像分辨率。如果没有，则应使用最大可用的码流图像分辨率。
"round-down"	对于每个请求的码流，应选择宽和高都小于或等于规定尺寸的最大码流图像分辨率。这是 round-direction 参数未规定时的默认值。
"closest"	对于每个请求的码流，应选择面积（其中面积＝fx×fy）最接近规定尺寸的码流图像分辨率。当有两个码流图像分辨率的面积和 fx×fy 差距相同时，应选择两者中较大的那个。

如果视窗请求中未规定帧尺寸请求域，且元数据请求域（见C.5.1）中未规定metadata-only，则被请求的视窗不包括压缩的图像数据和分片头，而包括当客户端包括帧尺寸请求域时将返回的所有其它的头（码流和文件格式）信息。关于视窗请求中隐式请求的文件格式信息（元数据）的进一步资料见C.5.1。

C.4.3 偏移量（roff）

```
roff = "roff" "=" ox "," oy

ox = UINT

oy = UINT
```

本域用于标识和被请求的视窗相关的空间区域的左上角（偏移量）；若不存在，偏移量默认为0。在服务器选择的实际码流图像分辨率下，码流图像区域相对于该图像的左上角的实际位移，根据C.4.1中描述的流程得到，另外可能还需根据码流上下文请求域（见C.4.7）中的请求作坐标变换。

偏移量域只能和帧尺寸请求域一起使用才有效。

如果用区域尺寸和/或偏移量定义码流图像域是空的（即没有面积），服务器的响应应不包括任何该码流的压缩图像数据。特别对于JPP-stream或JPT-stream类型的响应，应不包含任何引用该码流的分区、分

片或分片头数据块的消息。服务器可根据它的判断，选择返回的主头或者元数据块消息作为对没有帧尺寸请求域的请求的响应。

C.4.4 区域尺寸（rsiz）

```
rsiz = "rsiz" "=" sx "," sy

sx = UINT

sy = UINT
```

本域用于标识和被请求的视窗相关的空间区域的水平和垂直范围（尺寸），若不存在，区域将伸展到图像的最右下角。在服务器选择的实际码流图像分辨率下，码流图像空间的实际的尺寸，根据C.4.1中描述的流程计算得到，另外可能还需根据码流上下文请求域（见C.4.7）中的请求作坐标变换。请求的码流图像区域不一定完全包含在码流中，在这种情况下，服务器仅取可用的码流图像区域和请求的区域的交集。

区域尺寸请求域仅和帧尺寸请求域一起使用才有效。

码流图像区域可能是空的，如当sx或sy等于0的情况下。若为空，则服务器的响应不应包括该码流的任何压缩图像数据。特别对于JPP-stream或JPT-stream类型的响应，应不包含任何引用该码流的分区、分片或分片头数据块的消息。服务器可根据它的判断，选择返回的主头或者元数据块消息作为对没有帧尺寸请求域的请求的响应。

C.4.5 分量（comps）

```
comps = "comps" "=" 1#UINT-RANGE
```

本域用于标识包括在被请求的视窗中的图像分量；若不存在，应理解为该请求包括由Codestream请求域标识的所有码流的所有可能的图像分量，以及通过码流上下文请求域（见C.4.7）请求的所有码流的所有相关分量。这些“相关”分量是指那些通过码流请求域规定的图像实体（如JPX合成图层或MJ2视频轨迹）的复制品中包含的分量。

该请求域中的值表示感兴趣的图像分量的索引。图像分量索引从0开始，根据ITU-T T.800建议书| ISO/IEC 15444-1中描述的JPEG2000码流语法对所赋值进行解释。但应注意，这些是通过对压缩数据的解码和反小波变换获得的分量，而不是反RCT或ICT分量变换的应用。对符合ITU-T T.801建议书 | ISO/IEC 15444-2的码流，这里所指的分量是指那些空间分量，即那些通过对压缩数据进行解码和反小波变换得到分量，而不是任何反多分量变换、依赖分量变换或多分量小波变换的应用。

被请求的码流中的任何分量都不应忽略。

C.4.6 码流（stream）

```
stream = "stream" "=" 1#sampled-range

sampled-range = UINT-RANGE [":" sampling-factor]

sampling-factor = UINT
```

本域用于标识哪个或哪些码流属于被请求的视窗。如果该域不存在，且码流不能通过其它方式确定，默认值为标识符为0的单个码流。注意码流上下文请求域（见C.4.7），提供了请求码流的另一种方式。

对于JPEG 2000系列对象，码流索引放在相应的占位符框中，这些框出现在适当元数据块中，如A.3.6中描述。对于隐含码流标识符的文件格式，那些标识符应和这里使用的索引一致。

在标识一组码流时，若没有上限表示范围一直延伸到具有最大标识符的码流。若规定了上限，该上限规定了这个范围中最后的码流的绝对标识符。

不管是否规定了上限，码流的范围可由另外的sampling-factor来限制。sampling-factor，若存在，应是一个正整数*F*。这样，包括的范围为限制的范围内具有*L*+*Fk*的所有码流标识符，其中*L*是范围中的第一个码流标识符。客户端感兴趣的码流索引是*k*，且*k*是一个非零整数。

C.4.7 码流上下文（context）

```
context = "context" "=" 1#context-range

context-range = jpxl-context-range / mj2t-context / reserved-context

jpxl-context-range = "jpxl" "<" jpx-layers ">" [ "[" jpxl-geometry "]" ]

jpx-layers = sampled-range

jpxl-geometry = "s" jpx-iset "i" jpx-inum

jpx-iset = UINT

jpx-inum = UINT

mj2t-context = "mj2t" "<" mj2-track ">" [ "[" mj2t-geometry "]" ]

mj2-track = NONZERO ["+" "now" ]

mj2t-geometry = "track" / "movie"

reserved-context = 1*( TOKEN / "<" / ">" / "[" / "]" / "-" / ":" / "+" )
```

本域可用于通过“更高级别”的图像实体间接的请求码流。本建议书|国际标准定义了与JPX合成图层（一个JPX合成图层可包括一个或多个码流）和MJ2视频轨迹相关的上下文，但该机制具有可扩展性。

如果存在码流上下文请求域，被请求的视窗除了包括通过码流请求域请求的码流外，还包括每个和被请求的上下文相关的码流。

码流上下文请求域的主体包括一个或多个context-range值。每个context-range和一组服务器确定的码流相关。context-range也可标识为了确定每个和context-range相关的码流图像的分辨率和码流图像区域，应用于帧尺寸、区域尺寸和偏移量参数的坐标再映射变换。若服务器准备处理context-range，它应通过码流上下文响应头标识和该context-range相关的码流。

本建议书|国际标准定义了两种具体类型的context-range，用以满足JPX和MJ2文件格式的需求。context-range的第一种类型，jpxl-context-range用于标识一个或多个JPX合成图层。和jpxl-context-range相关的合成图层的索引，以sampled-range的形式提供，和码流请求域中抽样的码流范围的语法相同。其中jpxl-context-range由服务器处理，属于相应的合成图层的码流应在码流上下文响应头中标识。

jpxl-context-range可标识任意的坐标再映射变换，用于减小它的每个码流的码流图像分辨率和码流图像区域。该坐标再映射变换由两个非负整数确定，jpx-iset和jpx-inum。这两个整数合起来标识了逻辑对象范围内JPX分量（comp）框中的某个合成指令。所讨论的这个指令位于指令组（iset）框中，指令组（iset）框在合成框中顺序位置（从0开始）由jpx-iset值给出。jpx-inum值给出了该指令在那个指令组框中顺序位置（从0开始）。对这些索引的解释和在JPX合成框中可能出现的重复计数无关。

当服务器处理jpx-iset和jpx-inum时，应首先用等式C-3，将请求的帧尺寸和区域参数fx，fy，sx，sy，ox和oy，映射为修正的帧尺寸和区域参数fx"，fy"，sx"，sy"，ox"和oy"。这些修正的区域参数应对每个请求的码流进行单独的计算，然后当按照C.4.1描述的流程确定码流图像分辨率和码流图像区域时，代替fx，fy，sx，sy，ox和oy使用。

$$
\begin{aligned}
&\mathrm{fx}" = \left\lceil \mathrm{fx} \cdot \frac{\mathrm{XR}_{\mathrm{reg}}}{\mathrm{XS}_{\mathrm{reg}}} \cdot \frac{\mathrm{Wt}_{\mathrm{inst}}}{\mathrm{Ws}_{\mathrm{inst}}} \cdot \frac{\mathrm{W}_{\mathrm{cod}}}{\mathrm{W}_{\mathrm{comp}}} \right\rceil ; \quad \mathrm{fy}" = \left\lceil \mathrm{fy} \cdot \frac{\mathrm{YR}_{\mathrm{reg}}}{\mathrm{YS}_{\mathrm{reg}}} \cdot \frac{\mathrm{Ht}_{\mathrm{inst}}}{\mathrm{Hs}_{\mathrm{inst}}} \cdot \frac{\mathrm{H}_{\mathrm{cod}}}{\mathrm{H}_{\mathrm{comp}}} \right\rceil \\
&\mathrm{sx}" = \min\left\{(\mathrm{ox} + \mathrm{sx}), x_{\mathrm{lim}}\right\} - \max\left\{\mathrm{ox}, x_{\mathrm{min}}\right\} \\
&\mathrm{sy}" = \min\left\{(\mathrm{oy} + \mathrm{sy}), y_{\mathrm{lim}}\right\} - \max\left\{\mathrm{oy}, y_{\mathrm{min}}\right\} \\
&\mathrm{ox}" = \max\left\{\mathrm{ox}, x_{\mathrm{min}}\right\} - \left\lceil \left(\mathrm{XO}_{\mathrm{inst}} - \left(\mathrm{XC}_{\mathrm{inst}} - \frac{\mathrm{XO}_{\mathrm{reg}}}{\mathrm{XS}_{\mathrm{reg}}} \right) \cdot \frac{\mathrm{Wt}_{\mathrm{inst}}}{\mathrm{Ws}_{\mathrm{inst}}} \right) \cdot \frac{\mathrm{fx}}{\mathrm{W}_{\mathrm{comp}}} \right\rceil \\
&\mathrm{oy}" = \max\left\{\mathrm{oy}, y_{\mathrm{min}}\right\} - \left\lceil \left(\mathrm{YO}_{\mathrm{inst}} - \left(\mathrm{YC}_{\mathrm{inst}} - \frac{\mathrm{YO}_{\mathrm{reg}}}{\mathrm{YS}_{\mathrm{reg}}} \right) \cdot \frac{\mathrm{Ht}_{\mathrm{inst}}}{\mathrm{Hs}_{\mathrm{inst}}} \right) \cdot \frac{\mathrm{fy}}{\mathrm{H}_{\mathrm{comp}}} \right\rceil \\
&x_{\mathrm{min}} = \left\lfloor \mathrm{XO}_{\mathrm{inst}} \cdot \frac{\mathrm{fx}}{\mathrm{W}_{\mathrm{comp}}} \right\rfloor ; \quad y_{\mathrm{min}} = \left\lfloor \mathrm{YO}_{\mathrm{inst}} \cdot \frac{\mathrm{fy}}{\mathrm{H}_{\mathrm{comp}}} \right\rfloor \\
&x_{\mathrm{lim}} = \left\lceil \left(\mathrm{XO}_{\mathrm{inst}} + \mathrm{Wt}_{\mathrm{inst}}\right) \cdot \frac{\mathrm{fx}}{\mathrm{W}_{\mathrm{comp}}} \right\rceil ; \quad y_{\mathrm{lim}} = \left\lceil \left(\mathrm{YO}_{\mathrm{inst}} + \mathrm{Ht}_{\mathrm{inst}}\right) \cdot \frac{\mathrm{fy}}{\mathrm{H}_{\mathrm{comp}}} \right\rceil
\end{aligned}
\tag{C-3}
$$

注意由sx"，sy"，ox"和oy"定义的修正的视窗区域，可能会略微位于原来的区域左侧或者上面。也就是说，ox"和/或oy"可能是负的。当按照C.4.1描述的流程确定码流图像区域时，任何位于原来的区域左侧或者上面的视窗区域的部分，应被忽略。

如果没有jpx-iset和jpx-inum的值，用由等式C-4给出的修正的区域参数代替fx，fy，sx，sy，ox和oy的。和前面一样，当按照C.4.1描述的流程确定码流图像区域时，应使用这些修正的参数。

$$
\begin{aligned}
&\mathrm{fx}" = \left\lceil \mathrm{fx} \cdot \frac{\mathrm{XR}_{\mathrm{reg}}}{\mathrm{XS}_{\mathrm{reg}}} \cdot \frac{\mathrm{W}_{\mathrm{cod}}}{\mathrm{W}_{\mathrm{reg}}} \right\rceil ; \quad \mathrm{fy}" = \left\lceil \mathrm{fy} \cdot \frac{\mathrm{YR}_{\mathrm{reg}}}{\mathrm{YS}_{\mathrm{reg}}} \cdot \frac{\mathrm{H}_{\mathrm{cod}}}{\mathrm{H}_{\mathrm{reg}}} \right\rceil \\
&\mathrm{ox}" = \mathrm{ox} - \left\lceil \frac{\mathrm{XO}_{\mathrm{reg}}}{\mathrm{XS}_{\mathrm{reg}}} \cdot \frac{\mathrm{fx}}{\mathrm{W}_{\mathrm{reg}}} \right\rceil ; \quad \mathrm{oy}" = \mathrm{oy} - \left\lceil \frac{\mathrm{YO}_{\mathrm{reg}}}{\mathrm{YS}_{\mathrm{reg}}} \cdot \frac{\mathrm{fy}}{\mathrm{H}_{\mathrm{reg}}} \right\rceil \\
&\mathrm{sx}" = \mathrm{sx}; \quad \mathrm{sy}" = \mathrm{sy}
\end{aligned}
\tag{C-4}
$$

本建议书|国际标准中描述的context-range的第二种类型，mj2t-context，允许客户端从MJ2文件中请求某个轨迹。1是MJ2文件中允许的最小的轨迹标识符，因此mj2-track标识符必须是一个严格的正整数。如果mj2-track标识符中包括一个可选的"+now"下标，则mj2t-context由从获取时间等于请求收到的时间的码流开始的所有属于MJ2视频轨迹的码流组成。这对源是一个实况转播的视频流来说是非常有用的。其它情况下，服务器可能将"now"和它认为合适的任何码流相关联。如果不包括"+now"下标，mj2-context由所有属于MJ2视频轨迹的码流组成。

mj2t-context可规定一个坐标再映射变换，用于减小它的每个码流的码流图像分辨率和码流图像区域。若没有，由帧尺寸、偏移量和区域尺寸请求域规定的帧尺寸和区域参数应直接按照C.4.1中列出的流程解释。否则，根据"track"或"movie"令牌，请求一种或两种类型的坐标变换。

当规定了"track"，帧尺寸、偏移量和区域尺寸请求域用于标识在想要的显示尺寸下包含轨迹显示的最

小边界矩形内想要显示的尺寸和矩形区域。MJ2轨迹头（tkhd）框描述的几何变换应用于确定和轨迹相关的每个码流的相应的图像分辨率和区域。

当规定了“movie”，帧尺寸、偏移量和区域尺寸请求域用于标识整个复制电影想要的尺寸，以及在想要的尺寸下包含电影的最小边界矩形内想要的矩形区域。MJ2轨迹头（tkhd）框描述的几何变换应和电影头（mvhd）框描述的几何变换一起，用于确定和轨迹相关的每个码流的相应的图像分辨率和区域。

在服务器不能应用上面描述的`mj2t-context`几何变换的情况下，服务器在码流上下文的响应头中提供了一个修正的`mj2t-context`串。

注1 — 码流上下文请求域和码流请求域一起使用，可能会造成用帧尺寸、区域尺寸和偏移量请求域的不同的几何变换多次请求一个码流。若这样，会有效的请求该码流的多个分解的或重叠的图像部分。

注2 — 当在等式C-3中设$XS_{comp}=Ws_{inst}=Wt_{inst}=W_{reg}$，$YS_{comp}=Hs_{inst}=Ht_{inst}=H_{reg}$以及$XO_{inst}=YO_{inst}$，则当`sx"`，`sy"`，`ox"`和`oy"`不限制在$x_{lim}$，$x_{min}$，$y_{lim}$，$y_{min}$内时，等式C-4的表达式和等式C-3相等。

例1："context=jpxl<0-4:2>[s5i2]"

在这种情况下，服务器被请求返回JPX合成图层0、2和4使用的码流，根据分量框（JPX至多有一个分量框）中第六个指令框里的第三个指令所描绘的几何调整重新映射被请求的帧尺寸和图像区域。

例2："stream=0&context=mj2t<1+now>[track]"

在这种情况下，服务器被请求返回码流0和所有属于MJ2文件中的第一个轨迹的码流，从抽样时间等于当前时间的码流开始。此外，服务器被请求根据轨迹头框中描述的几何调整重新映射被请求的帧尺寸和图像区域，而不管任何其它在电影头框中描述的几何调整。

C.4.8 抽样率（srate）

```
srate = "srate" "=" streams-per-second

streams-per-second = UFLOAT
```

如果规定了这个域，属于视窗的码流是经过对那些码流请求域提及的码流和码流上下文请求域（见C.4.7）中上下文范围值扩展的码流的二次抽样得到的，这样就获得一个不比流/每秒的值大的平均抽样率。仅当码流具有关联的计时信息（比如它们属于符合MJ2文件格式的同一个逻辑对象）时才是可能的。

该请求域仅用于确定哪些码流应被认为属于视窗。服务器应扫描所有码流，丢弃必需的码流，确保码流的源时间之间的平均间隔不小于流/秒值的倒数。本建议书|国际标准未规定二次抽样的算法，也没有对术语“平均间隔”进行精确的解释。

如果没有源计时信息是可用的，则视窗将由码流请求域和码流上下文请求域标识的所有码流组成，但如果存在传送速率请求域，该请求可能会影响对它的解释。

C.4.9 ROI（roi）

```
roi = "roi" "=" region-name

region-name = 1*(DIGIT / ALPHA / "_")
            / "dynamic"
```

本域通过名字而不是坐标规定了想要的图像空间区域。`region-name`和某个图像空间区域的映射可来自几个地方，可在逻辑对象的ROI描述框中定义，也可在服务器本身的执行中定义。

`region-name`的值"dynamic"（动态ROI）被保留，用于描绘图像中的一个非恒定的区域，对于每一个请求，被独立映射成一个空间区域。当服务器确定应给那种请求分配什么空间区域时，服务器可利用关

于客户端的信息和其它请求参数。例如，如果服务器知道客户端的物理显示很小，它可能选择仅提供在较高的分辨率下图像的最重要区域，而不是在较低分辨率下图像的整个区域。服务器不要求支持动态ROIs。

如果存在ROI域，且服务器知道如何处理ROI请求，那么ROI域将比偏移量请求域和区域尺寸域的优先级高，服务器应忽略以上两个域。如果存在ROI域，但服务器由于某些原因不知道如何处理ROI请求，则服务器应忽略ROI域，使用偏移量和区域尺寸域。如果这些域都不存在，应使用这些域的默认值。

如果客户端既规定了帧尺寸，也规定了ROI，且服务器理解所规定的ROI，则帧尺寸请求域的值确定了ROI所请求的图像分辨率。

C.4.10 图层（layers）

```
layers = "layers" "=" UINT
```

本域可用于限制属于视窗请求的码流有效图层的数量。默认对所有可用的图层都感兴趣。该值规定了初始感兴趣的有效图层的数量。服务器不应试图增加任何超越相应图层边界的分区数据块。服务器不应试图增加任何超越所有位于相应图层边界外现存内容所在的点的分片数据块。由于分片内数据的顺序，服务器可能必须仅为JPT-stream请求返回超越被请求的图层边界的数据。

C.5 元数据请求域

C.5.1 在视窗请求中隐式请求的元数据

码流请求域和码流上下文请求域标识了一个或多个和被请求的视窗相关联的码流。如C.4.6中提到，即使这两个请求域都没有，视窗也至少和一个码流相关联。而且，如C.4.2中所述，即使帧尺寸请求域缺省，被请求的视窗也至少包括每个被请求的码流的主头。唯一的例外是在元数据请求域（见C.5.2）中规定了`metadata-only`的情况。除了这种情况，客户端为了使用被请求的码流所表示的图像，也隐性的请求文件格式所必须的任何元数据框。为了保证客户端和服务器分量之间的互操作性，本节规定了元数据的最小集合，服务器应认为这些元数据是在视窗中隐性请求的。当服务器意识到其它相关的元数据要素时，也会传送这些元数据。

对于JP2和JPX文件，应认为以下元数据要素是随着视窗一起被请求的：

a） 元数据块0的全部内容。

b） 每个以下框的全部内容，不管它们在文件顶层的何处出现：

1） JP2标签（"jP "）；

2） 文件类型（"ftyp"）；

3） 读者要求（"rreq"）；

4） 分量（"comp"）。

c） 以下每个超级框的所有直接的子框头：

1） 任何JP2头（"jp2h"）框；

2） 任何和被请求的码流相关的码流头（"jpch"）框；

3） 任何和码流上下文请求域请求的JPX合成图层相关的合成图层的头框。

d） 每个以下框的全面内容，不管这些框在以上提到的超级框中的何处出现：

1） 图像头（"ihdr"）；

2） 每个分量的比特（"bpcc"）；

3） 调色板（"pclr"）；

4） 分量映射（"cmap"）；

5） 通道定义（"cdef"）；

6） 分辨率（"res "）；

7） 码流注册（"creg"）；

8） 不透明性（"opct"）。

e） 对于JP2文件、JP2兼容文件以及JPX文件，一个或多个和码流上下文请求域请求的每个码流和JPX合成图层相关的彩色空间描述框（"colr"），如下：

1） 如果服务器能完全确定哪个框是首选的，服务器应仅发送那个框，即使这意味着不发送JP2或JP2兼容文件的第一个框（例如，如果第二个框是任意的ICC，且彩色空间偏好选择规定了客户端更喜欢任意的ICC）。如果服务器不能完全确定哪个框是首选的，它应发送第一个彩色空间描述框的全部内容。

2） 对于所有未发送的框，服务器应发送框内容的一部分，使客户端能确定它以后是否想请求另外的彩色空间规范。

- 对于列举框，服务器应至少发送框内容的前7字节（至少直到EnumCS域）。
- 对于厂商定义的彩色空间框，服务器应至少发送框内容的前19字节（至少直到VCLR域）。
- 对于受限的和任意的ICC彩色空间框，服务器应至少发送框内容的前3字节（至少发送METH，APPROX和PREC域）

服务器被请求返回包含以上提到的元数据的每个元数据块的初始前缀，从元数据块的第一个字节开始，一直到那个元数据块的所有被请求的元数据结尾。因此，服务器返回的实际的元数据数量可能取决于逻辑对象分割为元数据块的方式。关于这些问题的讨论，见A.3.6.2。

本建议书|国际标准没有提出关于什么组成视窗请求中隐性的MJ2元数据的建议，但这可能在将来的标准中定义。

C.5.2 元数据请求（metareq）

```
metareq = "metareq" "=" 1#("[" 1$(req-box-prop) "]" [root-bin] [max-depth])
                        [metadata-only]

req-box-prop = box-type [limit] [metareq-qualifier] [priority]

limit = ":" (UINT / "r")

metareq-qualifier = "/" 1*("w" / "s" / "g" / "a")

priority = "!"

root-bin = "R" UINT

max-depth = "D" UINT

metadata-only = "!!"
```

除要求客户端解码或理解被请求的图像数据的元数据（见C.5.1）外，本域规定了在这个请求的响应中还想要什么元数据。这个请求域内的值串是一列独立的请求，但服务器可将其作为一组请求进行处理，且请求之间可能有重叠。

每个请求和它的root-bin值所规定的数据块相关。如果root-bin值没有规定，则根是元数据块0。请求仅和该数据块内或者被某个数据块引用（通过占位符框）的数据有关。

如果规定了max-depth的值，只请求包含在根元数据块内以及文件层次中低于max-depth的框。如果max-depth的值没有规定，则该请求对文件层次的深度没有限制。

请求的req-box-prop部分规定了一列客户端感兴趣的框类型。特殊串"*"可用于代替框类型，这种情况表示隐含所有的框类型。每个框类型（或"*"）后，可跟三个属性的任意组合：limit值、metareq门限和priority标记。

limit规定了信息的类型，以及客户端请求哪种框类型的框内容。界限参数的格式是在冒号后接一个值（界限值），该值应是一个无符号整数或者符号"r"。

如果界限值是一个比0大的整数*n*，则除了框头，仅要求服务器返回那种框类型的*n*个字节的相应框的内容。如果界限值为0，则只请求那种类型框的头。如果limit没有规定，则客户端请求的是符合请求中其它方面的那种类型的所有框的全部内容，不管那种类型的框是否为超级框。同样，如果请求的是超级框且界限值是一个数值或者未规定的情况下，服务器被请求提供limit所请求的数据数量，不管那个超级框所包含的层次是否比根据root-bin和max-depth值所达到的更深，也不管超级框内的子框的类型。

如果界限值是"r"，则服务器被请求发送框类型所指示的框以及所有它的一直到请求中指出的最大深度的子孙的子框（不管它们的框类型是什么）的框头，而不是框内容。这实际上是对框层次的架构部分的请求。如果服务器不能确定一个框是否是超级框，它可能不能解析到框的子框，从而不能完全响应某些元数据请求。服务器应能识别它们需支持的文件格式所定义的所有超级框的状态。

虽然界限值"r"意味着客户端是请求由框头组成的框结构的架构，但将逻辑对象分割为元数据块，可能迫使服务器返回其它的数据，包括一些框的内容以及其它未请求的框的头和/或内容。这是因为服务器被请求返回从包含被请求的框字节的每个元数据块的开始到被请求的框字节的最后的所有字节。

metareq-qualifier的格式是"/"后面跟一个或多个标记"g"、"s"、"w"和"a"。每个标记标识了一个和请求匹配的框的上下文。关于这些上下文的解释，见表C.2。如果提供了多于一个的标记，应联合相应的上下文。如果metareq-qualifier没有规定，则应使用"g"，"s"和"w"的联合。为了澄清，应注意上下文"g"，"s"和"w"是互相独立的，但它们的联合通常比单个"a"小。

如果规定了priority标记，则客户端请求那些和请求中其它要素相匹配的box-type规定的类型的框传送的优先级比其它图像数据高。

不请求任何未在req-box-prop列表中规定的框类型的数据。

如果在元数据请求域的最后规定了metadata-only，客户端请求服务器的响应仅有元数据组成，不包含任何图像数据或码流头，不管是否在视窗请求域中使用了如帧尺寸等的请求域。对于JPP-stream和JPT-stream的返回类型，这意味着返回的JPIP消息都是元数据块消息。

例1："metareq=[*]R31D4"

在这种情况下，服务器被请求返回数据块31中所有框的全部内容。尽管规定了想要的深度的限制，但服务器应忽略这个限制，因为那些框中的内容未通过limit参数限制。

例2："metareq=[*:r,drep]R31D4"

"*:r"表示服务器被请求返回所有元数据块31中以及被该数据块内的占位符引用的任何数据块的框头，一直到数据块31的内容的4级深度，但不包括那些框的内容。另外"drep"的req-box-prop规定了服务器被请求返回元数据块31内以及被该数据块内的占位符所引用的任何"drep"框，一直到数据块31的内容的4级深度。

例3："metareq=[drep]R31D4"

在这种情况下，服务器仍被请求返回元数据块31内以及被该数据块内的占位符所引用的任何"drep"框的所有内容，一直到数据块31的内容的4级深度。但由于没有规定其它的框，服务器仅被请求发送尽量多的其他数据，这些数据对于规定任何"drep"框在和元数据块31中的框相关的文件层次中的位置是必要的。

不管元数据请求域对框作了什么规定，服务器可发送其它的数据，如果它确定这些数据对客

户端的解码或解释被请求的图像数据是必需的，或者服务器以前已经用不同的标准将逻辑对象分割成数据块，应发送附加的数据使该逻辑对象的所有元数据块一致、有意义。

表 C.2－元数据请求中限定词（qualifier）标记

标记	说明
"w"	这个元数据请求上下文包括所有被认为和属于视窗的一个或多个码流内某个空间图像区域相关联的框，这些框相关的空间区域、分辨率和图像分量和视窗的相交叉。例如，这种关联可能是通过JPX 文件中的"asoc"框确定的。
"s"	这个元数据上下文包括所有被认为和属于视窗的一个或多个码流或者和一个或多个被请求的码流上下文（如 JPX 合成图层或 MJ2 视频轨迹）相关联的框，这些框不仅仅和某个空间区域相关联。例如，这种关联可能是通过 JPX 文件中的"asoc"框确定的。
"g"	这个元数据上下文包括所有和被请求的视窗相关联的框，其中考虑被请求的码流和被请求的码流上下文，但不包括在"w"和"s"元数据请求上下文中的那些框。
"a"	这个元数据上下文包括所有逻辑对象中的框，没有例外（注）。
注 — 这个元数据上下文适合那些希望独立的询问视窗文件结构的请求。	

C.6 数据限制请求域

C.6.1 最大响应长度（len）

```
len = "len" "=" UINT
```

本域规定了客户端想要服务器在对这个请求的响应中发送的数据量的限制。单位为字节。如果不存在，服务器应给客户端发送图像数据，直到发送完所有相关数据、达到质量限制（见C.6.2）或者响应被一个新到来的未包括等待请求域为"yes"的（见C.7.2）请求中断。客户端如果只请求响应头，而没有响应数据时，应使用len=0。

C.6.2 质量（quality）

```
quality = "quality" "=" (1*2DIGIT / "100")          ; 0 to 100
```

本域可以用于将和图像相关的数据传输限制在一个质量级别（在0最低质量和100最高质量之间）。质量限制很难以可靠的方式进行阐明，且服务器可能通过响应"–1"值（见D.2.16）忽略这个请求。然而，允许客户端提供它所感兴趣的最高图像的质量的某个指示是有用的。质量因子可能试图接近通常用于控制JPEG压缩的ad hoc 质量。客户端应预期随着质量的提高返回的数据大小不单纯不减小，也就是说，提高质量值，通常相应的应增加返回的数据大小。

注 — 如果服务器支持这个请求，且两个不同的客户端对同一个对象提出相同的质量值的请求，例如"quality=80"，服务器在从数据块中返回数据上应有一致的执行策略。

C.7 服务器控制请求域

C.7.1 对齐（align）

```
align = "align" "=" ("yes" / "no")
```

本域规定了服务器响应数据是否应按自然界限进行对齐。缺省值为"no"。如果值是"yes"，在对这个请求的响应中传送任何JPT-stream和JPP-stream消息中，如果穿越了某个“自然界限”，应在随后的“自然界限”上终止。每种类型的数据块的自然界限如表C.3所列。如果一个消息包括界限前的最后一个字节和界限后的第一个字节，则被认为是穿越自然界限了。例如，如果一个分区数据块包括一个数据包的最后一个字节和下一个数据包的第一个字节，则它就穿越了自然界限。注意，被对齐的响应消息实际上并不一定终止在自然界限上，除非穿越了一个界限。这表示，例如，响应可能只包括分区的部分数据包，这在一个主要的字节限制阻止了所有数据包的传送的情况下可能是必要。

表 C.3－基于块类型的对齐界限

块类型	自然界限
分区数据块	数据包的结尾 (每个质量级别一个界限)
分片数据块	分片部分的结尾（每个分片部分一个界限）
分片头数据块	块的结尾（只有一个界限）
主头数据块	块的结尾（只有一个界限）
源数据块	高层数据块的框的结尾（每个框一个界限）

C.7.2 等待（wait）

```
wait = "wait" "=" ("yes" / "no")
```

本域用于表示服务器是否应完成对以前请求的响应。如果本域的值是"yes"，则服务器应在开始响应该请求前，完成对相同通道ID标识的通道资源上前面的请求的响应。

如果本域的值是"no"，服务器可以在完成前逐渐的结束对相同通道资源（用通道ID域标识）上的任何前面请求的处理，并开始对该新的请求进行响应。在文中，“逐渐的结束”是指服务器应至少完成当前的消息。

本域的缺省值是"no"。

C.7.3 图像返回类型（type）

```
type = "type" "=" 1#image-return-type
image-return-type = media-type / reserved-image-return-type
media-type = TOKEN "/" TOKEN *( ";" parameter )
reserved-image-return-type = TOKEN *( ";" parameter )
parameter = attribute "=" value
attribute = TOKEN
value = TOKEN
```

本域用于标识请求的响应数据的类型。服务器不愿提供任何请求的返回类型时应发送一个错误响应。

图像返回类型请求域的值应是一种媒体类型（RFC 2046中定义），或者表C.4中定义的保留的图像返回类型之一。

表 C.4－合法的图像返回类型

类型	解释
"jpp-stream"	附件 A 中定义的 JPP-stream。"jpp-stream"后可任选的跟";ptype=ext"，在这种情况下，请求返回的是所有分区数据块的消息头具有扩展形式的类型（见 A.2.2）
"jpt-stream"	附件 A 中定义的 JPT-stream。"jpt-stream"后可任选的跟";ttype=ext"，在这种情况下，请求是所有分片数据块的消息头具有扩展形式的类型（见 A.2.2）
"raw"	客户端请求逻辑对象的所有字节按原来的顺序传送。
其他值	保留供 ISO 使用。

如果type请求域不存在，则返回类型应根据其它方式确定。

在一个会话中，即所有请求包括一个通道ID请求域的情况下，返回的参数值应在对同一个逻辑对象的图像数据或元数据的请求的连续响应过程中保持不变。

注1 — 如果存在其它图像媒体类型（如jp2，jpeg，tiff，png），服务器能作为JPIP功能的转换码提供服务。

注2 — 对于"raw"码流返回类型，响应数据应全部由被请求的实体组成。因此，很多其它可能的客户端请求域没有意义，服务器将其忽略。

C.7.4 传送速率（drate）

```
drate = "drate" "=" rate-factor

rate-factor = UFLOAT
```

本域用于规定各种码流的传送速率。如果提供了本域，服务器应按照一个临时的顺序时间表传送属于视窗中的各种码流数据。这些属于视窗的码流，都由码流请求域或码流上下文请求域确定，且可能根据抽样率请求域进行了子抽样。

为了使该请求域有意义，应将时间信息和视窗中的各种码流关联起来。如果码流属于一个MJ2文件，则应由那个文件提供时间信息。MJ2文件规定了每个码流和标称播放时间之间的映射，这里称作“源时间”。

如果码流没有源时间信息，但存在抽样率请求域，服务器应认为视窗内码流的源时间的间隔为抽样率请求域中值的倒数。

如果码流没有源时间信息，也不存在抽样率请求域，服务器应认为视窗内码流的源时间的间隔刚好为1秒。

传送速率请求域提供了传送速率和源速率之间的缩放因子。如果rate-factor为1，则服务器应试图以建议的源时间的速率给客户端传送码流，应注意这些源时间不一定是规则的。更一般地，如果rate-factor是F，则服务器应试图以比建议的源时间快F倍的速率给客户端传送码流。

如果服务器不能以请求的速率传送每个码流的所有相关数据（如由于带宽的限制），则应传送每个码流的一部分数据，从而避免违反请求的传送速率。每个码流没有传送的那部分数据，可以根据客户端偏好请求域（见C.10.2）提供的`view-window-pref`值确定。如果偏好是"progressive"，或者缺省，则服务器应试图传送在传送速率限制下视窗上的完整的、最高的图像质量。如果`view-window-pref`的值为`"fullwindow"`，服务器可以以某种方式截短每个码流的表示。无论哪种情况，结果应类似于客户端以传送速率依次连续的请求每个相关码流所得到的结果。

如果服务器能以请求的速率传送每个码流的所有相关数据，应使请求的连接空闲以保证不超过传送速率。

如果未规定本域，且未规定`view-window-pref`的值是`"fullwindow"`，服务器应试图以统一渐进增加所有码流质量的方式排列相关的数据。

C.8 缓存管理请求域

C.8.1 模型（model）

C.8.1.1 概述

```
model = "model" "=" 1#model-item

model-item = [codestream-qualifier ","] model-element

model-element = ["-"] bin-descriptor

bin-descriptor = explicit-bin-descriptor    ; C.8.1.2
               / implicit-bin-descriptor    ; C.8.1.3

codestream-qualifier = "[" 1$(codestream-range) "]"

codestream-range = first-codestream-id ["-" [last-codestream-id]]

first-codestream-id = UINT

last-codestream-id = UINT
```

本域可用于基于会话和无状态的请求中。由于通道和服务器管理的会话是相关联的，因此基于会话的请求是指任何包括通道ID域的请求。"model"域包含一个或多个块描述符，每个块描述符标识了一个或者

一组数据块的缓存信息。对基于会话的请求，该缓存信息用于更新服务器关于客户端缓存的模型。和会话相关的每个逻辑对象只有一个缓存模型。对于无状态请求，在请求开始时服务器关于客户端缓存的模型是空的，但在服务器作出应答前由"model"域（如果存在）进行更新。当服务器认为处理的是一个无状态的请求时，服务器将丢弃所有缓存模型的信息。

为便于缓存模型信息的高效交换，规定了两种形式的块描述符值。被称为“显式”和“隐式”形式，具体在后续节中描述。客户端可用任何一种形式发送请求，且如果想要的话也可将两种形式的块描述符放在同一个"model"请求域中。

如果块描述符前带"-"号，表示减。否则，表示加。减块描述符告知服务器应将相关数据从服务器关于客户端缓存的模型中删除。从缓存模型中删除要素表示服务器不应认为客户端已经有这些要素。块描述符的值按顺序进行处理。

加块描述符（前面不带"-"号）告知服务器客户端在它的缓存中已经有的数据。服务器可以给它的缓存模型增加该信息，并认为客户端已经有这些数据。

"model"域可以涉及和由其它请求域（帧尺寸，区域尺寸，Offset等）确定的感兴趣的视窗不相关的数据块。如果这样，缓存模型的操作处理可能不影响对当前的请求的响应，但可能会影响将来的请求（除非请求是无状态的）。

当模型条款列表中包括一个码流限定，应从码流限定中列出的所有码流中（适当的）加或者减去所有后续的模型元素。码流限定可以散布在整个列表中，渐进的改变被后续的模型元素影响的码流集。前面没有码流限定的任何模型元素适用于通过码流请求域请求的第一个码流。如果码流请求域不存在，则不管是否存在码流上下文请求域，前面没有码流限定的模型元素值应指向码流0。如果last-codestream-id不存在，但限定存在，则应包括first-codestream-id和所有后续的码流。

一个会话中的请求不应包括任何涉及多于一个码流的码流限定r。

注1 — 服务器应试图开发附加的缓存模型处理声明，但在可能会引起传输效率降低时可自由的忽略某个或所有附加的缓存模型处理声明。客户端应意识到服务器很可能会忽略那些指向属于不在当前请求中服务的码流的数据块的附加的缓存模型处理声明。在包括多个码流时，为了减少这种不确定性，可以使用"mset"请求域来确定正在建模的码流组。

注2 — 基于会话的服务器缓存模型的处理，通常对当前和任何将来的请求的响应产生影响。而且，会话内所有和同一个逻辑对象相关联的通道共享同一个缓存模型。因此，使用同一个通道（通道ID域）到达的请求中的"model"域可能影响使用不同通道到达的请求的响应。注意如果用单独的TCP通道直接从客户端或者间接的从中间代理传输请求，使用不同JPIP通道（不同的通道ID值）的请求可能异步到达服务器，这是非常重要的。客户端应采取必要的行动，保证它们的缓存模型操作指令在这些考虑下保持有效。

C.8.1.2 显式

```
explicit-bin-descriptor = explicit-bin
                          [":" (number-of-bytes / number-of-layers )]
explicit-bin = codestream-main-header-bin
             / meta-bin
             / tile-bin
             / tile-header-bin
             / precinct-bin
number-of-bytes = UINT
number-of-layers = %x4c UINT              ; "L"
codestream-main-header-bin = %x48 %x6d    ; "Hm"
meta-bin = %x4d bin-uid                   ; "M"
tile-bin = %x54 bin-uid                   ; "T"
tile-header-bin = %x48 bin-uid            ; "H"
```

```
precinct-bin = %x50 bin-uid                ; "P"

bin-uid = UINT / "*"
```

块描述符的值是显式的是指数据块为如下的类型：M（元数据块），Hm（主头数据块），H（分片头数据块），P（分区数据块）或者T（分片数据块）。显式块描述符用一个唯一的整数标识符或者通配符"*"识别相关码流内的数据块。唯一的例外是码流的主头数据块时，块描述符是"Hm"。对于所有其它数据块类型，唯一的标识符等于JPP-stream 和/或JPT-stream中消息头中传送的in-class标识符的值（见附件A）。

通配符"*"应仅用于无状态请求中。在使用时，块描述符同时指向和视窗相关的相应类型（元数据、分区、分片头或者分片）的所有数据块。

每个块描述符用字节数进行限制。限制为*B*个字节的加块描述符，表示客户端的缓存中至少已经有该数据块的前*B*个字节；服务器可以在它的缓存模型中增加数据块的前*B*个字节。限制为*B*个字节的减块描述符，表示客户端至多有该数据块的前*B*个字节；服务器应在它的缓存模型中删除数据块的*B*字节后的字节。

例1：一个被限制的减块描述符如"-P23:10"表示服务器应从它的缓存模型中删除分区数据块23的所有除前10个字节外的数据。这并不意味着客户端缓存中有分区数据块23的前10个字节，且如果这些数据不存在的话，服务器不应认为这样而在它的缓存模型中增加这些字节。

区域块描述符还可以用图层数进行限制。限制为*L*个图层的加块描述符，表示客户端的缓存中至少已经有该区域的前*L*个图层（前*L*个数据包）；服务器可以在它的缓存模型中增加对应于这些图层的字节。限制为*L*个图层的减块描述符，表示客户端至多有该区域的前*L*个图层；服务器应在它的缓存模型中删除对应于区域中后续图层的字节。

没有number-of-bytes 或number-of-layers限制的块描述符表示整个显式的数据块。

例2："model=M0,Hm,H7:20,P3"表示客户端的缓存中至少有元数据块0、主码流头、分片头7的前20个字节以及分区3的所有字节。

例3："model=P3:256,P5:L2,-P6:20"表示客户端的缓存中至少有分区3的前256个字节、分区5的前2个图层（数据包），但没有分区6的第20字节后的数据（也可能没有前20个字节）。

例4："model=M*,-M5,-H*,-P*:L3"表示客户端的缓存中有除元数据块5外的所有元数据块，没有和视窗相关的分片头数据块、至多有和视窗相关的任何分区的前3个图层。注意，这里使用的通配符只有当"model"声明出现在无状态请求中时才允许。

例5："model=[30-200],Hm,H*,M*,P0,[0-29],-Hm,-H*,-M*,-P*"表示客户端有码流30到且包括200的所有头和元数据以及分区数据块0，但已经删除了所有前30个码流的头、元数据和分区数据块。

C.8.1.3 隐式

```
implicit-bin-descriptor = 1*implicit-bin [":" number-of-layers]

implicit-bin  = implicit-bin-prefix (data-uid / index-range-spec)

implicit-bin-prefix = %x74     ; t - tile
                    / %x63     ; c - component
                    / %x72     ; r -- resolution level
                    / %x70     ; p -- position

index-range-spec = first-index-pos "-" last-index-pos

first-index-pos = UINT

last-index-pos = UINT

data-uid = UINT / "*"
```

块描述符的值是隐式的是指数据块为如下的类型：t（分区所属的分片），c（分区所属的图像分量），r（分区所属的分片分量的分辨率级别）或者p（分片分量分辨率内分区的位置）。隐式块描述符用于通过索引标识分区数据块。所有索引从0开始。索引为0的分辨率级别r0，指分片分量中最低的分辨率级别（LL子带）。位置的索引p，在分片分量的分辨率程序中从左到右，从上到下按扫描线的方式进行编排索引，如ITU-T T.800建议书 | ISO/IEC15444-1中描述。

在无状态请求中，任意或者所有的分片、分量、分辨率级别或者位置的索引都可以用一个索引范围或通配符"*"代替。在每种情况下，块描述符都扩展到包括和视窗相关的所有索引范围的值。以上两种情况都不应用于会话中的请求。

在无状态请求中，任意或者所有的分片、分量、分辨率级别或者位置索引还可以用单个索引范围值代替。index-range-spec 中的first-index-pos值给出了范围中的第一个索引。last-index-pos值给出了范围中的最后一个索引，且应大于或等于first-index-pos的值。这两个索引值都是包含的。last-index-pos不可以省略。若给出了分片索引（"t"）的范围，该范围指分片的一个矩形阵列，其左上角为first-index-pos的值，右下角为last-index-pos。类似地，若给出了区域索引（"p"）的范围，该范围指分区位置的一个矩形阵列，其左上角和右下角分别由irst-index-pos和last-index-pos值给出。至于通配符，其范围不应用于会话中的请求。

隐式区域块描述符可以用图层数进行限制，其语法和解释同上面介绍的显式分区数据块用图层限制的情况。

例1："model=t0c2r3p4:L5"表示客户端有分片0中第3个分量的第4级分辨率级别下位置序列为5的前5个数据包。

例2："model=t10r0,t*r1:L4"表示客户端有分片索引号为10中分辨率级别0下的所有图层，以及所有和视窗相关在分辨率级别1下的分片的前4个图层。注意通配符仅适用于无状态请求。

例3："model=t0-10:L2"表示客户端有分片0到10的前2个图层。注意该范围仅适用于无状态请求。

例4："model=t*r0-2:L4"表示客户端有所有和视窗相关的分片的分辨率级别从0到2的前4个图层。注意通配符和范围仅适用于无状态请求。

C.8.2 缓存描述符可选项的总结（资料性的）

表 C.5－缓存描述符可选项总结

形式类型	通配符		索引范围	图层数（如":L3"）	字节数（如":256"）
	无状态	基于会话			
显式	允许	不允许	不允许	允许	允许
隐式	允许	不允许	仅对无状态允许	允许	不允许

C.8.3 JPT-streams中的分片部分模型（tpmodel）

```
tpmodel =  "tpmodel" "=" 1#tpmodel-item
tpmodel-item = [codestream-qualifier "," ] tpmodel-element
tpmodel-element = ["-"] tp-descriptor
tp-descriptor = tp-range / tp-number
tp-range = tp-number "-" tp-number
tp-number = tile-number "." part-number
tile-number = UINT
part-number = UINT
```

本域可以用于表示客户端想要从服务器缓存模型中加上或者减去的某个分片部分。就像"model"域，它既可用于基于会话的请求也可用于无状态请求。在无状态请求下，缓存模型在请求开始时是空的，且请求之间不连续，但它仍规定了用于识别已经在客户端缓存中的图像元素的一种有用的机制。

如果分片部分描述符前带"-"号，表示减。否则，表示加。加分片部分描述符表示客户端的缓存中已经有指定的分片部分或者分片部分范围；服务器可以给它的缓存模型增加这些元素。减分片部分描述符表示客户端的缓存中没有指定的分片部分或者分片部分范围；服务器应在它的缓存模型中删除这些元素。

分片部分编号中的第一个值是分片的索引（从0开始）；第二个值是分片内部分的编号（从0开始）。tp-range被认为是独立的包含从第一个分片编号到第二个分片编号的分片，以及从第一个分片部分编号到第二个分片部分编号的分片部分。因此4.0-5.1包括分片部分4.0、4.1、5.0和5.1，但不包括4.2和5.2。

"tpmodel"和"model"请求域可以出现在同一个请求中。但在这种情况下，服务器应在处理"tpmodel"域前先反映"model"域对它的 缓存模型的影响。

Codestream-qualifier值可散布在tpmodel-elements的列表之间，改变后续tpmodel-elements应用的码流集合，原理完全同"model"请求域。

注 — 和"model"请求域不同的是，"tpmodel"请求域中分片部分范围和码流范围（在codestream-qualifiers内）都允许，不管出现在基于会话还是无状态的请求中。

例1："tpmodel=4.0,4.1,5.0-6.2"表示客户端的缓存中已经有分片4的前两个分片部分，以及分片5和6的前三个分片部分。

例2："tpmodel=-4.0-6.254"表示客户端的缓存中没有分片4、5和6的分片部分。

例3："tpmodel=3.0,[131-133],4.0,[100],-0.0-65534.254"表示客户端的缓存中有请求中参考的码流0的分片3的分片部分0，加上131到包括133的每个码流的分片4的分片部分0，减去缓存中码流100的所有分片部分。

C.8.4 无状态请求的需要（need）

```
need = "need" "=" 1#need-item

need-item = [codestream-qualifier "," ] bin-descriptor
```

本域仅可能出现在无状态请求中，即那些不包括通道ID请求域的请求。除了bin-descriptors前不应加"-"符号外，和模型请求域的语法相同。"need"请求域不应和"model"或"tpmodel"请求域出现在同一个请求中。

"need"请求域表示客户端可能感兴趣的数据块组（或数据块下标）。服务器不需发送可能不感兴趣的信息。不管可能感兴趣的数据块组有多大，服务器仅应该发送和视窗请求域或元数据块请求域相关联的信息。

"need"域对服务器请求的影响可用一个临时缓存模型概念进行解释。临时缓存模型在处理请求之前初始化（置空），在生成响应之后被立即丢弃。如果请求中有"need"域，先将所有可能的数据加入缓存模型中，再从缓存模型中删除"need"域中bin-descriptors涉及的所有元素。然后服务器处理被请求的视窗，用该缓存模型确定不需要发送给客户端的元素。

Codestream-qualifiers可散布在bin-descriptors的列表之间，改变后续bin-descriptors应用的码流集合，原理完全同"model"和"tpmodel"请求域。

例1："need=M1,H0:20,P0"表示客户端需要所有元数据块1，分片头数据块0的从第20个字节开始的数据以及所有分区数据块0。

例2："need=P1:256,P5:L2"表示客户需要分区数据块1中第256个字节后（或者从字节256开始）的数据，以及分区数据块5中第2个图层后的数据。

例3："need= H*,P*:L3"表示客户端需要所有和视窗相关的分片头数据块以及所有和视窗相关的分区数据块的第3个图层后的数据。

例4："need=t/or0,t*r1:L4"表示客户端需要分片索引号为10中分辨率级0下的所有层，以及和视窗相关的在分辨率1的所有分片的第4层以下的各层。

例5："need=t*r0-2:L4"表示客户端需要在所有和视窗请求相关的分片和分量中，分辨率级别在0到2（0、1和2）下的所有分区数据块的从图层4开始的所有图层。

例6："need=[120-131],r0,[140;143-145],r0-1"表示客户端需要码流120到包括131的分辨率级别0，以及码流140和143到包括145的分辨率级别0和1。

C.8.5 无状态请求的分片部分需要（tpneed）

```
tpneed = "tpneed" "=" 1#tpneed-item

tpneed-item = [codestream-qualifier "," ] tp-descriptor
```

本域仅可能出现在无状态请求中，即那些不包括通道ID的请求域的请求。除了tp-descriptors前不应加"-"符号外，和tpmodel请求域的语法相同。"tpneed"请求域不应和"model"或"tpmodel"请求域出现在同一个请求中。

"tpneed"请求域表示客户端可能感兴趣的分片部分组。服务器不需发送可能不感兴趣的信息。不管可能感兴趣的数据块组有多大，服务器仅应该发送和视窗请求域或元数据块请求域相关联的信息。

"tpneed"域对服务器请求的影响可用一个临时缓存模型概念进行解释。临时缓存模型在处理请求之前初始化（置空），在生成响应之后被立即丢弃。如果请求中有"tpneed"域，先将所有可能的数据加入缓存模型中，再从缓存模型中删除"need"域中bin-descriptors涉及的所有元素以及"tpneed"域中所有分片部分。然后服务器处理被请求的视窗，用该缓存模型确定不需要发送给客户端的元素。

Codestream-qualifiers可散布在tile-parts的列表之间，改变后续tile-parts应用的码流集合，原理完全同"model"和"tpmodel"请求域。

C.8.6 会话中请求的模型组（mset）

```
mset = "mset" "=" 1#sampled-range
```

本域用于两个目的：第一，告知服务器客户端准备缓存服务器发送的数据的码流组。第二，它规定了客户端学习服务器准备建立客户端缓存模型的码流一种机制。特殊地，如果"mset"请求中提供的码流索引的集合和服务器正准备提供缓存建模的码流组存在任何差别，服务器应提供一个模型组响应头，如D.2.18中讨论。

"mset"请求域的参数串是由逗号隔开的码流索引范围组成，这些索引可能按照C.4.6中码流请求域相关的规约被子抽样。

除了"mset"请求中提到的码流外，服务器还可以建立一个和它对当前请求的所有响应相关的码流的缓存模型。这是客户端请求（见C.4.7中码流和码流上下文）所标识的码流集合，除非服务器通过码流响应头标识了缩减的码流组（见D.2.9）。如果没有规定"mset"请求域，客户端不应认为服务器为那些不和它的响应相关的码流提供了缓存模型，但服务器可能为其它码流建立模型。如果规定了"mset"请求域，服务器应丢弃任何已经有的、关于那些不在"mset"请求中以及与它的响应数据相关的码流组中所提及的所有码流的模型信息。而且，"model"或"tpmodel"请求域规定的任何缓存模型操作的影响，应仅限于这些码流。

服务器可以根据它的判断，减少"mset"中的码流数目，这种情况下，服务器应提供一个"mset"响应头，标识已被建模的实际的码流组；而且，这个已被建模的码流组至少应包括和服务器响应数据相关的所有码流（即那些客户端请求中请求的或者服务器码流响应头所标识的（如果有））。在这种情况下，这些声明应用于那些包含在服务器标识的"mset"中的码流。服务器不能标识比客户端"mset"请求中提出的码流组更大的码流组，这些码流的集合和服务器的响应数据相关。

注意对于不同的请求，服务器可以改变它的"mset"，因此需要跟踪和/或紧紧限制服务器的"mset"的客户端，可以采用在每一个请求中都包括"mset"请求域。

C.9 上传请求参数

C.9.1 上传（upload）

```
upload = "upload" "=" upload-type
upload-type = image-return-type                    ; C.7.3
```

本域规定了服务器正在给服务器上传新的图像或元数据。upload-type的值可以是类型请求域中可能使用的任何有效的image-return-type值。上传数据的信息见附件E。

C.10 客户端能力和偏好请求域

C.10.1 客户端能力（cap）

```
cap = "cap" "=" 1#capability-group
capability-group = processing-capability
                 / depth-capability
                 / config-capability
processing-capability = compatibility-capability
                      / vendor-capability
compatibility-capability = "cc." compatibility-code
vendor-capability = "vc." vendor-code [":" vendor-value]
vendor-code = 1*(LOWER / DIGIT / "." / "-")
vendor-value = TOKEN
depth-capability = "depth:" UINT
config-capability = "config:" UINT
```

本域规定了客户端的能力。对基于会话的请求（即包括通道ID请求域的请求），客户端发送的能力域应仅影响和请求相关的通道，且应被永久考虑。对于同一个通道的后续请求，客户端不需要重新发送能力。

当在已有的通道中创建一个新的通道时，它的客户端能力被继承。对于无状态请求以及在未规定或继承能力的通道内的请求，客户端的能力可以通过其它方法确定或预测。通道的能力可以被包含客户端能力请求域的任何请求改变。

如果客户端能力请求域规定了一个或多个processing-capability选项，服务器应认为客户端没有其它的processing-capability选项。如果客户端能力请求域中没有提供processing-capability选项，服务器应继续使用任何前面关于处理能力的知识。本建议书|国际标准定义的processing-capability选项见表C.6。

表 C.6—processing-capabilities 元素的合法能力

能力	意义
compatibility-capability	客户端支持文件类型框内兼容列表中包含 compatibility-code 的所有文件。例如，为了指出客户端支持所有的 JP2 文件，客户端会在能力请求域中发送"cc.jp2_"。能力码值为"jp2c"用于说明支持原始 JPEG 2000 码流。
vendor-capability	客户端支持 vendor-code 中定义的厂商能力。vendor-code 是一个串，规定了定义特征的厂商域名的倒转，后接厂商特征名。例如，如果 example.com 定义了"distance"，则该特征 vendor-code 的值为"com.example.distance"。vendor-value 规定了一个可选值，由具体的厂商特征定义。

如果规定了depth-capability参数，指出了客户端能用于解压缩图像的最大抽样位深度（精度）。如果客户端对不同的图像分量支持不同的位深度，本域应指明客户端具有的最大位深度能力的分量的位深度。

注1 — 如果客户端支持12比特的亮度和8比特的色度，depth-capability值应为12。

注2 — 具有仅处理每个抽样N比特能力的客户端，通常还能处理SIZ标记指示大于*N*位深的码流。但服务器可使用该标记确定适当的方式传送被请求的图像数据。

如果提供了config-capability参数，它的范围是0-255，表示成8比特字，每个比特是一个配置的标记。配置标记的解释由表C.7提供。

表 C.7－config-capability 参数的合法值

值	意义
1xxx yyyy	客户端能处理彩色图像数据。
0xxx yyyy	客户端不能处理彩色图像数据，希望服务器发送灰色的被请求的图像区域。
x1xx yyyy	客户端有和终端用户交互用的定点设备
x0xx yyyy	客户端没有和终端用户交互用的定点设备
xx1x yyyy	客户端有和终端用户交互用的键盘
xx0x yyyy	客户端没有和终端用户交互用的键盘
xxx1 yyyy	客户端有声音输出能力
xxx0 yyyy	客户端没有声音输出能力
Other values	保留供 ISO 使用

表C.7中"x"表示的值表示规定的值在这一比特上可以为"1"或者"0"。"y"表示的比特在本建议书|国际标准中不使用，客户端应设置为"0"，服务器应忽略它。

C.10.2 客户端偏好（pref）

C.10.2.1 概述

```
pref = "pref" "=" 1#(related-pref-set ["/r"])
related-pref-set = view-window-pref            ; C.10.2.2
                 / colour-meth-pref            ; C.10.2.3
                 / max-bandwidth               ; C.10.2.4
                 / bandwidth-slice             ; C.10.2.5
                 / placeholder-pref            ; C.10.2.6
                 / codestream-seq-pref         ; C.10.2.7
                 / other
other = TOKEN
```

本域规定了对服务器行为的客户端偏好。对基于会话的请求（即包括通道ID请求域），客户端发送的偏好域应仅影响和请求相关的通道，且应被永久考虑。对于同一个通道的后续请求，客户端不需要重新发送偏好。每个偏好在每个偏好请求域中不应出现多于一次。

当在已有的通道中创建一个新的通道时，它的偏好被继承。对于无状态请求以及在未规定或继承偏好的通道内的请求，客户端的偏好可以通过其它方法确定或预测。如果客户端想要改变它的偏好，应再次发送整个受影响的related-pref-set。

除非另外声明，每个related-pref-set规定了一个个人偏好令牌的顺序列表，偏好程度从高到低。如果可能，服务器应遵从客户端在该请求域中提出的偏好。如果related-pref-set后接"/r"修饰语（必需的），服务器要么支持related-pref-set列表中的一个偏好，否则应响应一个错误。在后一种情况，服务器应返回一个偏好不可得响应头，标识任何具有"/r"修饰语但不能支持的related-pref-set。偏好不可得响应头的更多信息见D.2.20。

例如，考虑以下客户端偏好请求：

```
pref=fullwindow/r,color-ricc:2;color-icc
```

这个偏好请求要求服务器返回整个请求的视窗，而不管视窗可能有多大（"fullwindow"偏好的讨论见C.10.2.2）。由于使用了"/r"修饰语，服务器应返回一个错误响应，除非它能支持该偏好。另外，客户端更喜欢使用受限的ICC而不是任意的ICC轮廓，提供的受限的ICC至少是"exceptional quality"。关于彩色空间偏好的讨论见C.10.2.3。

服务器应忽略任何它不理解且后面没有紧接"/r"的`related-pref-set`的值。如果不理解的值后紧接"/r"，则服务器应返回偏好不可得响应头，标识它不能执行的偏好。

令牌其他的值作为ISO备用。

C.10.2.2 视窗处理的偏好

```
view-window-pref = "fullwindow" / "progressive"
```

本建议书|国际标准定义了两个可选项，用以规定在不能完全按照请求中的声明进行服务的情况下服务器的行为，根据是否按照响应数据质量渐进的顺序。这两种选项在表C.8中规定。

表 C.8－视窗处理偏好

选项	意义
"fullwindow"	服务器应尊重视窗请求参数但允许以非质量渐进的顺序返回数据。
"progressive"	服务器可能为了保持响应数据的质量渐进 quality-progressive 属性修改视窗请求参数。在服务器修改了视窗请求参数的情况下，被修改的视窗应是原来被请求的视窗的一个子集。

如果客户端偏好请求域中既没有规定"fullwindow" 也没有规定"progressive"，服务器应认为客户端的偏好是"progressive"。

注意对"progressive"的解释的传送可能受传送速率请求域出现的影响，具体在C.7.4中解释。

C.10.2.3 彩色空间方法的偏好

```
color-meth-pref = 1$(color-meth [":" meth-limit])

color-meth = "color-enum" / "color-ricc" / "color-icc" / "color-vend"

meth-limit = UINT
```

本建议书|国际标准定义了四个选项，规定服务器应以什么形式返回彩色空间规范数据。一个JPEG 2000文件可能包含多个对单个码流或合成图层的彩色空间规范。这允许文件编写者在提供能共同使用的解决方案的同时，能规定最优化的彩色空间规范。

但并不是所有的读者支持所有的彩色空间方法，且提供某些彩色空间方法的数据量可能很大。在这些情况下，服务器应只发送客户端想要的彩色空间规范。

如果客户端偏好请求域没有包含任何彩色空间方法偏好，则根据能力域中的信息定义支持的彩色空间方法，且不定义偏好。

每个彩色空间方法偏好由两部分组成：某个彩色空间方法以及可选的对该偏好的限制。彩色空间方法的合法值在表C.9中规定。

表 C.9－客户端彩色空间方法的偏好

方法	意义
"color-enum"	客户端更喜欢使用列举法的彩色空间规范
"color-ricc"	客户端更喜欢使用受限的 ICC 法的彩色空间规范
"color-icc"	客户端更喜欢使用任意 ICC 法的彩色空间规范
"color-vend"	客户端更喜欢使用厂商定义的彩色空间规范

可选项meth-limit值规定了那个特定的彩色空间方法的APPROX值的限制。当用这些偏好选择彩色空间规范时，服务器应把等于或小于meth-limit中的APPROX值的彩色空间方法规范看成其实际的APPOX值为1（精确的）。这允许客户端规定某些点，在这些点上色彩保真度在目前的应用中对某个彩色空间方法不重要。例如，在一个只关心图像数据排列的页面设计应用中，页面上关于色彩保真度的其它元素可能根本不在乎的中，可将meth-limit设为4，表明彩色空间方法的精确度不重要。另一个在低质量屏幕上显示图像的应用，可设置meth-limit为3，表明只要色彩精确度合理就能满足要求。本域的字符应解释为无符号的十进制整数。合法值在ITU-T T.801建议书 | ISO/IEC 15444-2的表M.24中关于APPROX域的定义中给出，且在那个建议书| 国际标准中进行扩展和修正。

当选择哪个彩色空间规范框传送给客户端时，服务器应采用以下的算法，如图C.3所示。

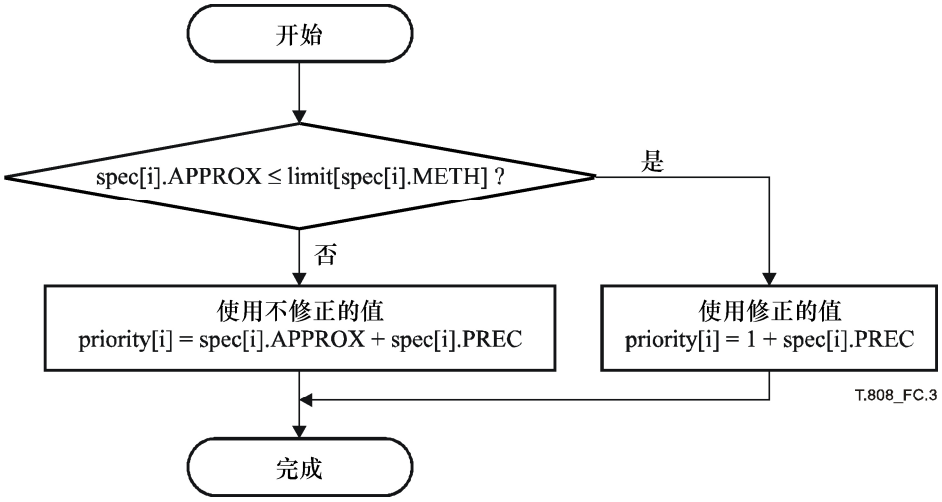

图 C.3－彩色空间规范框的选择程序

对于每个使用客户端所支持的方法的彩色空间规范框，其中：

- spec[]是一个数组，包含所有给定的逻辑对象的彩色空间规范框。
- spec[i].APPROX是逻辑对象中第i个彩色空间规范框的APPROX域的值。
- spec[i].METH是逻辑对象中第i个彩色空间规范框的METH域的值。
- spec[i].PREC是逻辑对象中第i个彩色空间规范框的PREC域的值。
- limit[]是一个数组，包含请求域中规定的meth-limit值，用彩色空间规范框中METH域的合法值进行编序。
- priority[]是一个数组，计算得到的给定的逻辑对象中每个彩色空间规范框的优先值。priority[i]和spec[i]相对应。

如果服务器知道客户端不支持某个特定的彩色空间规范框，在服务器在选择偏爱的彩色空间规范框时应忽略那个框。当计算出每个支持的彩色空间规范框的priority[]值后，服务器应选择具有最小优先值的框。这

时，如果该逻辑对象中有多好框具有同一个优先值，服务器应按照以下的偏好顺序选择彩色空间方法。

1） 列举法；

2） 厂商定义的方法；

3） 受限的ICC法；

4） 任意ICC法；

如果将一个文件分割成多个元数据块，服务器可能不管客户端对彩色空间规范的偏好，返回比通过这个算法得到的单个色彩框更多的彩色空间规范框。

C.10.2.4 最大带宽

```
max-bandwidth = "mbw:" mbw

mbw = UINT ["K" / "M" / "G" / "T"]
```

该偏好指示了客户端想要的每个逻辑对象数据的最大发送速率。如果mbw值以"K"结尾，则该值的单位是千比特/秒，其中1千比特 = 1 024比特。如果mbw值以"M"结尾，则该值的单位是兆比特/秒，其中1兆比特 = 1024^2比特。如果mbw值以"G"结尾，则该值的单位是吉比特/秒，其中1吉比特 = 1024^3比特。如果mbw值以"T"结尾，则该值的单位是兆兆比特/秒，其中1兆兆比特 = 1024^4比特。否则，该值的单位是比特/秒。服务器或者网络的能力可进一步限制JPIP业务的可用最大带宽。

C.10.2.5 带宽切分片

```
bandwidth-slice = "slice:" slice

slice = NONZERO
```

该偏好可能用于标识应分配给该通道的可用的带宽的分数。slice值应严格大于0。带宽分数通过将每一个通道的切分片值除以所有通道切分片值的总和得到。如果没有规定，通道的切分片值默认为1。

例如，可能用一个较低的slice值请求一个背景视窗，而用一个较高的slice值请求一个前景视窗。如果会话中包含和不同逻辑对象相关的多个通道，切分片值将影响分配给那些不同对象（图像）的可用带宽的比例。

C.10.2.6 占位符偏好

```
placeholder-pref = "meta:" placeholder-branch

placeholder-branch = "incr" / "equiv" / "orig"
```

本域可能用于指示占位符框的首选处理。当在JPP-stream或JPT-stream的元数据中出现占位符框时，可能有多达三种不同的框内容的表示方法：原始框，流等价框和增量的码流（用索引表示）。这些可能性在A.3.6和A.4中进行解释。如A.4中的解释，建议的默认假设是，如果有增量码流这种表示方法，客户端将更喜欢接收增量码流，否则，如果有流等价框这种表示方法，将更喜欢接收流等价框。客户端还可以用这里描述的机制指示替代的偏好。占位符偏好的合法值在表C.10中规定。

表 C.10－占位符偏好

方法	意义
"orig"	客户端更喜欢接收原始框，若有。否则，客户端更喜欢接收码流等价框，若有。
"equiv"	客户端更喜欢接收接收码流等价框，若有。否则，客户端更喜欢接收原始框，若有。
"incr"	客户端更喜欢接收接收码增量码流数据块，若有。否则，客户端更喜欢接收接收码流等价框，若有。这和建议的默认策略相同。

规定多个占位符偏好的值是不合法。

C.10.2.7 码流排序

```
codestream-seq-pref = "codeseq:" codestream-seq-option

codestream-seq-option = "sequential" / "reverse-sequential"
                      / "interleaved"
```

该偏好可用于表示客户端想要服务器如何传送在一个请求中请求的多个码流。码流排序偏好的合法值在表C.11中规定。

表 C.11－码流排序偏好

方法	意义
"sequential"	客户端更喜欢按照帧序列（例如一个运动 JPEG 2000 文件中的多个帧）顺序接收多个码流
"reverse-sequential"	客户端更喜欢按照帧序列（例如一个运动 JPEG 2000 文件中的多个帧）反序接收多个码流
"interleaved"	客户端更喜欢以交叉方式（例如交叉 JPX 文件中的多个合成图层）接收多个码流

规定多个码流排序偏好的值是不合法。

C.10.3 对比灵敏度（csf）

```
csf = "csf" "=" 1#csf-sample-line

csf-sample-line = csf-density [";" csf-angle] ";" 1$sensitivity

csf-density = "density" ":" UFLOAT

csf-angle = "angle" ":" UFLOAT

sensitivity = UFLOAT
```

本域可用于提供关于对比灵敏度的信息。该信息既反映了视觉灵敏度的影响，也反映了显示设备的调制转换功能的影响，而根据一个假定的调制转换功能将更容易描述。当图像按帧尺寸请求域中标识的帧尺寸复制后，假设通过一个调制转换功能（MTF）为$m(\omega_1, \omega_2)$的设备，之后由一个人看到，他的人类视觉系统具有完全均衡的对比灵敏度功能。MTF $m(\omega_1, \omega_2)$通过一个样本集合描述。样本是径向对数间隔的，沿着一个或多个方向轴。服务器可能为了恢复MTF，用它认为合适的任何方法更改这些样本，这反过来又可用于调整通过JPP-stream或JPT-stream 消息传送给客户端的数据块的字节范围的顺序。

每个csf-sample-line描绘了由$\omega_1 = \pi d^n \cos\psi$、$\omega_2 = \pi d^n \sin\psi$确定的MTF抽样$m(\omega_1, \omega_2)$，其中，$n$是抽样索引，从$n = 0$开始，$n = 0$是csf-sample-line中的第一个csf-density抽样，ψ是CSF样本行的方向，用度表示（如果没有csf-angle值，缺省为0），d是抽样密度，它应不大于1.0。ω_1值表示径向的水平频率，其中$\omega_1 = \pi$是水平奈奎斯特频率。ω_2值表示径向的垂直频率，其中$\omega_2 = \pi$是垂直奈奎斯特频率。

MTF抽样值只有相对意义，其绝对值没有具体的含义。

附 件 D

服务器响应信令

（本附件是本建议书|国际标准的组成部分）

D.1 响应语法

D.1.1 引言

本附件介绍了JPIP响应中所有可能的要素。各主要节描述了状态码及相关联的原因分析（Reason Phrase）、响应头和可能的取值，以及响应数据。一般情况下，一个响应由多个响应头组成。

D.1.2 响应结构

JPIP响应包括以下要素：

— 状态码（status-code）；

— 原因分析（reason-phrase）；

— JPIP响应头（jpip-response-header）；

— 响应数据（response data）。

响应中的这些要素应该遵从具体采用的传输协议。例如，在采用HTTP时，状态码和原因分析出现在状态行字段，JPIP响应头出现在HTTP的响应头字段，响应数据（如果有）则出现在HTTP的实体中。

```
Status-Code = 3DIGIT
Reason-Phrase = *<TEXT, excluding CR and LF>
jpip-response-header =
                      / JPIP-tid                    ; D.2.2
                      / JPIP-cnew                   ; D.2.3
                      / JPIP-qid                    ; D.2.4
                      / JPIP-fsiz                   ; D.2.5
                      / JPIP-rsiz                   ; D.2.6
                      / JPIP-roff                   ; D.2.7
                      / JPIP-comps                  ; D.2.8
                      / JPIP-stream                 ; D.2.9
                      / JPIP-context                ; D.2.10
                      / JPIP-roi                    ; D.2.11
                      / JPIP-layers                 ; D.2.12
                      / JPIP-srate                  ; D.2.13
                      / JPIP-metareq                ; D.2.14
                      / JPIP-len                    ; D.2.15
                      / JPIP-quality                ; D.2.16
                      / JPIP-type                   ; D.2.17
                      / JPIP-mset                   ; D.2.18
                      / JPIP-cap                    ; D.2.19
                      / JPIP-pref                   ; D.2.20
```

原因分析字符串应该为状态码提供一个文本解释。以下状态码可能对于JPIP应用而言已经足够了。

D.1.3 状态码和原因分析

D.1.3.1 概述

状态码是3位整数，用于标识对请求的理解和满足结果。JPIP采用了HTTP/1.1中状态码和原因分析的一个子集，JPIP客户端应等待以下的码值。如果是基于HTTP的JPIP客户端也可以看到其它的状态码。

D.1.3.2 200（成功）

如果服务器接受客户端的视窗请求，则可以用此状态码进行响应。此外，服务器可以通过在响应消息中的其他头字段中进行指示，修改被请求的视窗。

D.1.3.3 202（已接受）

如果视窗请求可以被接受，但队列中有后续视窗请求（因为wait=no）出现，从而取代了当前的视窗请求，这种情况下，服务器产生该状态码。在服务器能够对第一个请求作出处理并开始传输响应消息之前，如果第一个请求变得不相关，就应使用202状态码。这种情况在实际中经常发生，因为在服务器对较早的请求作出响应或者准备好中断正在进行的处理之前，交互用户可能多次改变他/她感兴趣的区域。

D.1.3.4 400（错误请求）

如果请求的格式不正确，或者在查询串中含有无法识别的字段时，服务器就产生该状态码。

D.1.3.5 404（未找到）

如果服务器不能找到和对象ID相一致的被请求资源，则产生该状态码。这可能由于企图进行未授权的存取，更多的则是由于超过了时间限制。如果由于连接不好客户端错过了时间窗口，它可能发现对象ID已不再有效。

D.1.3.6 415（不支持的媒体类型）

如果服务器无法对图像返回类型请求域中指定的单个图像类型提供服务，则可以使用此状态码标示。

D.1.3.7 501（不可实现）

如果服务器不能提供本建议书|国际标准中的一部分服务，而该部分服务又是请求必需的，在这种情况下，就可能使用该状态码。

D.1.3.8 503（服务不可用）

如果在通道ID请求域中指定的通道ID不正确，则使用该状态码。

D.2 JPIP响应头

D.2.1 对JPIP响应头的介绍

在对客户端请求作出响应时，服务器可能修改请求的某些方面。如果服务器修改了请求，那么修改后的参数应该通过响应头进行标识。响应头的名字来源于被修改的请求域的名字，通过在请求域名字前面加上前缀“JPIP-”得到。除非另外说明，如果响应头中规定的参数已经在客户端的请求中作了规定，则服务器可以用相同的方式响应，例外的情况是现在响应消息中不包括这些响应头。此外，服务器可能发送的JPIP响应头用于通知客户端未在请求域中规定的参数值，客户端在将来的请求中可以利用这些参数值。

JPIP-qid响应是一个例外，不论客户端的请求中是否包含请求ID，JPIP-qid都会被发送，JPIP-qid的值通常和qid相同。

响应头中的参数和原来的请求域中的参数用相同的BNF元素标识，而且两者的含义和格式也完全相同。

唯一的例外情况是新建通道和质量响应头。

D.2.2 对象ID（JPIP-tid）

```
JPIP-tid = "JPIP-tid" ":" LWSP target-id
```

如果服务器的唯一对象标识符和对象ID请求域中提供的标识符不同，或者客户端没有规定对象ID请求域，则服务器需要发送该响应头。Target-ID是任意的、由服务器分配的长度不超过255的字符串。如果对象ID请求域规定的值为"0"，则服务器被强制发送对象ID响应头，指示实际的对象ID。如果服务器不能分配一个唯一的标识符给被请求的逻辑对象，从而不能保证多个请求或会话间的完整性，则把对象ID响应头的值设为0。如果服务器提供的对象ID和请求中规定的值不同，当响应此请求时，应忽略所有的model、tpmodel、need和tpneed请求域。

D.2.3 新建通道（JPIP-cnew）

```
JPIP-cnew = "JPIP-cnew" ":" LWSP "cid" "=" channel-id
            ["," 1#(transport-param "=" TOKEN)]

transport-param = TOKEN
```

当且仅当服务器响应新建通道请求域、分配一个新的通道时，才会发送该响应头。值串由逗号隔开的name=value对组成，其中第一对标识了新通道的通道ID令牌。

本建议书|国际标准定义了下列transport-param令牌（见表D.1）。

表 D.1－transport-param 的合法值

值	意义
"transport"	该参数中应分配新建通道请求域中提供的可接收的传输名字类表中的一个值。如果在请求域中提供了多个传输名字，响应头应标识出通道使用的实际传输。
"host"	该参数标识了正在管理该新通道的 JPIP 服务器的主机名字或者 IP 地址。除非和请求实际被发送给的主机不同，否则不需要返回这个参数。
"path"	该参数标识了在该通道上构造未来请求时用到的 URL 的路径分量。除非和实际发送的请求中用到的路径名不同，否则不需要返回这个参数。
"port"	该参数标识了正在管理新通道的 JPIP 服务器上正在监听请求的数字的端口号（十进制）。如果和那些原始请求发送到的主机和端口号相同，不需要返回该参数。如果和请求实际被发送给的主机不同，且使用相关传输的默认端口号，也不需要返回该参数。
"auxport"	该参数和需要第二个物理通道的传输一起使用。如果用"http-tcp"，辅助端口用于连接辅助的 TCP 通道。具体见附件 G。如果原始请求也包括利用辅助通道的通道，且具有相同的辅助端口号，则不需要返回该参数。否则，只有当辅助端口号和选择的传输的默认值不同时，才需要返回该参数。

D.2.4 请求ID（JPIP-qid）

```
JPIP-qid = "JPIP-qid" ":" LWSP UINT
```

如果客户端的请求中包括请求ID qid，则服务器需要发送该响应头。JPIP-qid的值应该和qid相同。如果客户端请求不包括请求ID，则服务器也不能包括请求ID响应头。

注 — 服务器的请求ID，JPIP-qid应总是和客户端的请求ID相同。由于只有当客户端使用了请求ID而不是服务器修改该值时才发送该响应头，因此该请求ID不同于一般的情况。

D.2.5 帧尺寸（JPIP-fsiz）

```
JPIP-fsiz = "JPIP-fsiz" ":" LWSP fx "," fy
```

如果响应数据中帧尺寸和通过帧尺寸请求域请求的不同，则服务器发送该响应头。

D.2.6 区域尺寸（JPIP-rsiz）

```
JPIP-rsiz = "JPIP-rsiz" ":" LWSP sx "," sy
```

如果响应数据中的区域尺寸和请求的不同，则服务器发送该响应头。

D.2.7 偏移量（JPIP-roff）

```
JPIP-roff = "JPIP-roff" ":" LWSP ox "," oy
```

如果响应数据中的区域偏移量和请求的不同，则服务器发送该响应头。

D.2.8 分量（JPIP-comps）

```
JPIP-comps = "JPIP-comps" ":" LWSP 1#UINT-RANGE
```

如果响应数据中的分量和通过分量请求域请求的不同，则服务器发送该响应头。如果任何被请求码流中均不包含被请求的图像分量，该响应头可以不发送。

D.2.9 码流（JPIP-stream）

```
JPIP-stream = "JPIP-stream" ":" LWSP 1#(prefixed-range / sampled-range)
prefixed-range = "<" ctxt-id ":" ctxt-elt ">" sampled-range
ctxt-id = UINT
ctxt-elt = UINT
```

服务器通过发送该响应头来通知客户端它将提供数据的一个或多个码流，例外的情况是当提供服务的这些码流响应了码流请求域直接和码流上下文请求域间接请求的所有码流时，则不需要发送该响应头。服务器应该使用prefixed-range语法来标识那些对被翻译的码流上下文请求域进行响应的码流。这种情况下，ctxt-id用于标识来自码流上下文请求域的具体context-range，对它的翻译可以产生相关的码流范围。进一步，ctxt-elt的值用于标识由ctxt-id确定的context-range里的特定元素，对它的翻译可以产生出相关的码流。

ctxt-id的值为0表示码流上下文请求域中的第一个context-range用于产生跟在前缀后面的码流范围。类似地，ctxt-id的值为1表示码流上下文请求域中的第二个context-range用于产生后面的码流范围，以此类推。

ctxt-elt的值为0表示在相关context-range里的第一个上下文用于产生跟在前缀后面的码流范围。

例：

用户端请求：

stream=0&context=jpxl<2-7:2>[s0i0],jpxl<3-5>[s1i3]

服务器响应：

JPIP-context: jpxl<2-7:2>[s0i0]=0,1;jpxl<9-10>[s1i3]=0

JPIP-stream: 0,<0:1>1,<1:0>0,<1:1>0

这表示服务器以下面的数据响应用户端的请求：

1） 直接应用于码流0的视窗（通过"stream=0"请求）；

2） 根据合成指令组0中的合成指令0翻译成JPX合成图层4的视窗，其应用于码流1上；

3） 根据合成指令组1中的合成指令3翻译成JPX合成图层9的视窗，其应用于码流0上；以及

4） 根据合成指令组1中的合成指令3翻译成JPX合成图层10的视窗，其应用于码流0上。

D.2.10 码流上下文（JPIP-context）

```
JPIP-context = "JPIP-context" ":" LWSP 1$(context-range "=" 1#sampled-range)
```

如果服务器能够处理码流上下文请求域提供的任何一个context-range，则发送该响应头。该响应头描述了正在处理的context-range以及与之相关所有码流索引。服务器可能忽略那些原来由码流上下文请求域提供但当前并未处理的context-range。此外，服务器也可能修改原来在码流上下文请求域中提供的context-range。允许两种修改：

a） 服务器可能限制原来被请求的图像元素（如合成图层等）的集合。

b） 服务器可能丢弃不支持的几何变换修正（如一个mj2t-context串中的“track”或“movie”修正）。

D.2.11 ROI（JPIP-roi）

```
JPIP-roi = "JPIP-roi" ":" LWSP
           "roi" "=" region-name ";"
           "fsiz" "=" UINT "," UINT ";"
           "rsiz" "=" UINT "," UINT ";"
           "roff" "=" UINT "," UINT ";"
region-name = 1*(DIGIT / ALPHA / "_")
```

在响应客户端对ROI的请求时，服务器应通过ROI响应头指定实际提供服务的ROI范围。如果服务器不能满足ROI请求，只需在回复时，将ROI响应头简单设成“JPIP-roi:roi=no-roi”。除了ROI外，服务器还通过帧尺寸、区域尺寸和偏移量响应头指定正在服务的图像区域作为反馈。

如果服务器能提供ROI服务，但由于某些原因需要对返回的部分图像的大小进行调整，则服务器可以发送ROI响应头对ROI进行描述，同时发送帧尺寸、区域尺寸和偏移量响应头对返回的部分ROI进行描述。

D.2.12 图层（JPIP-layers）

```
JPIP-layers = "JPIP-layers" ":" LWSP UINT
```

如果提供服务的图层数少于layers请求域中指定的值，服务器应发送该响应头。既然视窗通常以质量渐进方式进行服务，并不强制要求服务器（实际上可能无法）确定传送的响应数据跨越的图层数。然而，如果被请求的图层数超过了视窗中所有码流所能提供的图层数，服务器至少应标识可用的最大图层数。对于接受对齐请求域（见C.7.1）的服务器，在其提供服务的图层数小于图层请求域中指定的值时，应该发送JPIP-layers响应。

D.2.13 抽样率（JPIP-srate）

```
JPIP-srate = "JPIP-srate" ":" LWSP UFLOAT
```

如果发送到客户端的码流平均抽样率预计和抽样率请求域请求的值不同，且该抽样率已知，则服务器应发送该响应头。如果源码流无时间信息，则不应发送该响应头。

D.2.14 元数据请求（JPIP-metareq）

```
JPIP-metareq = "JPIP-metareq" ":" LWSP
               1#( "[" 1$(req-box-prop) "]" [root-bin] [max-depth] )
               [metadata-only]
req-box-prop = box-type [limit] [metareq-qualifier] [priority]
```

如果服务器修改了元数据请求请求域中提供的max-depth、limit、metareq-qualifier或priority的值，则应发送该响应头。

D.2.15 最大响应长度（JPIP-len）

```
JPIP-len = "JPIP-len" ":" LWSP UINT
```

如果最大响应长度请求域中指定的字节界限太小，以至于不能容纳一个非空响应，且该字节界限不为零，则服务器应发送该响应头。如果返回该响应头，则JPIP-len用于通知客户端一个合适的最大响应长度值len，用于后续的请求。如果len=0，服务器应只用响应头对请求作出响应，而不包含响应数据。

D.2.16 质量（JPIP-quality）

```
JPIP-quality = "JPIP-quality" ":" LWSP (1*2DIGIT / "100" / "-1")
```

一旦完成客户端的请求，服务器就发送该响应头来通知客户端与返回图像相关的质量值。如果当前请求被另一个请求中断（没有设置“wait=yes”），则此质量值可能不准确。该质量值仅仅依赖于被请求的视窗，而且和质量请求域的解释方法相同。如果服务器忽略了客户端的请求，应返回值“–1”。

D.2.17 图像返回类型（JPIP-type）

```
JPIP-type = "JPIP-type" ":" LWSP image-return-type
```

除非另外的机制标识返回图像数据的MIME子类型，否则服务器应发送该响应头。另外的机制包括：

— HTTP“Content-Type”头；

— 响应和某一会话相关的请求，而该会话的返回图像类型已被告知。

D.2.18 模型组（JPIP-mset）

```
JPIP-mset = "JPIP-mset" ":" LWSP 1#sampled-range
```

如果客户端请求中包含模型组请求域，且该模型组请求域指定的码流组和由服务器实际准备维持缓存模型信息的码流组有任何不同，则服务器应发送该响应头。由服务器维护缓存模型信息的那些码流组应包括和服务器响应数据相关的所有码流（或者在客户端请求中指定，或者通过服务器发送的码流响应头指定）。除了那些码流，服务器的“mset”可能没有客户端的模型组请求域指定的值大。

D.2.19 需要的能力（JPIP-cap）

```
JPIP-cap = "JPIP-cap" ":" LWSP 1#capability-code
```

为对逻辑对象作出一致解释，该响应头指定了客户端应支持的特定特征。合法的能力和表C.6中为能力 请求域定义的值相同。

D.2.20 偏好不可得（JPIP-pref）

```
JPIP-pref = "JPIP-pref" ":" LWSP 1#related-pref-set
```

当且仅当客户端偏好请求域包含一个带有"/r"修饰语（必需的）的`related-pref-set`，且服务器不愿支持时，应提供该响应头。在这种情况下，还应返回一个错误值作为响应状态码。值串由一个或多个不能支持的`related-pref-sets`组成，完全以客户端偏好请求中出现的形式进行重复。

尽管是值得要的，但该响应头不必列出所有不能被支持的`related-pref-sets`。因而，允许服务器读取客户端偏好请求域，直到碰到标有`"/r"`且不能支持的`related-pref-set`为止。更多关于该响应头什么时候要使用的信息，见C.10.2.1。

表 D.2－定义的原因码

原因码	原因	解释
1	图像已经完成	服务器已经发送了所有可用的图像信息（不仅仅和被请求的视窗相关的信息）给客户端。该原因码对基于会话的请求有特殊的意义。对一个基于会话的请求，该原因码表示客户端已经接收了所有能发送的对和该逻辑对象相关的基于会话的请求进行响应的数据。可能的一种例外请求是包括缓存管理请求域的请求，其后续任何基于会话的请求的响应是没有响应数据，且 R=1 EOR。
2	窗口已经完成	服务器已经发送了所有和被请求的视窗相关的可用的信息。该原因码对基于会话的请求有特殊的意义。对一个基于会话的请求，该原因码表示客户端已经接收了所有能发送的对该请求响应数据，且响应数据没有被请求中的数据限制域（len 或 quality）或后续请求的处理所限制。可能的一种例外请求是包括缓存管理请求域的请求，其后续接收到的请求的响应是没有响应数据，且 R=2 EOR。
3	窗口改变	为了给没有规定 Wait=yes 的一个新的请求提供服务，服务器正终止它的响应。
4	字节门限已经达到	由于最大响应长度请求域中规定的字节门限已经达到，服务器正终止它的响应。
5	质量门限已经达到	由于质量请求域中规定的质量门限限制已经达到，服务器正终止它的响应。
6	会话门限已经达到	由于会话资源如时间的门限已经达到，服务器正终止它的响应。不应再用和那个会话相关的通道 ID 发送请求。
7	响应门限已经达到	由于某个门限如时间门限已经达到，服务器正终止它的响应。如果发送了会话内的请求，以后的请求还能用和那个会话相关的通道 ID 发送。
0xFF	未规定的原因	由于某个未规定的原因服务器正终止它的响应。
其他值		保留供 ISO 使用。

D.3 响应的数据

对于JPP-或JPT-stream以外的图像返回类型，包括原始码流，响应数据应包括全部被请求的实体。对于JPP-或JPT-stream图像返回类型，响应数据由附件A中定义的一连串消息组成，并以一个EOR（响应结束）消息作为结束。附件A中未定义EOR消息，不是JPP-或JPT-stream媒体类型的正式组成部分。

EOR消息由一个头部和一个主体组成。EOR消息头包括一个字节的标识符，0x00，后面跟着一个字节的原因码，R，然后是一个VBAS字节计数器，用于指示EOR消息主体的字节数。本建议书|国际标准没有为EOR消息主体的内容提供标准解释。

注意，附件C中定义了最大响应长度请求域，而EOR消息主体并不对与该请求域相关的字节计数限制造成影响。

注意，EOR消息意味着服务器已经发送完与客户端请求相关的数据块中所有的相关内容。这并不一定是这些数据块中的全部内容。因为当达到了客户端指定的某一界限，响应就会被终止。如果没有指定任何界限，那么EOR消息将意味着已经提供了相关数据块的全部内容。

目前定义的原因码见表D.2。

附 件 E

上传图像到服务器

（本附件是本建议书|国际标准的组成部分）

E.1 引言

图像会预先以多种方式置于服务器，对这些方式的介绍已经超出了本建议书|国际标准的范围。本附件的目的在于描述一种允许将图像各部分上传到服务器的机制。

E.2 上传请求

E.2.1 请求结构

上传请求由附件C中定义的一个或多个请求域和一个请求主体组成。

E.2.2 上传请求域

一次上传的请求域应包括一个上传请求域。对象、子对象和对象ID请求域（见C.2.2，C.2.3和C.2.4）也可能用到。对于一个完整图像媒体类型的上传，帧尺寸、偏移量和区域尺寸请求域（见C.4.2，C.4.3和C.4.4）用于指示上传部分在整个图像中的位置。对于JPT-stream和JPP-stream的上传，数据块编号（还有分片或分区编号）以及主头指示了编码数据的位置，视窗请求域不是必需的。

E.2.3 上传请求主体

E.2.3.1 概述

上传请求主体由支持的图像类型中的一种组成：JPP-stream、JPT-stream或完整图像媒体类型。主体包含客户端请求服务器处理的数据。本建议书|国际标准不支持原始图像数据的上传。

E.2.3.2 JPT-stream

请求主体由客户端想要服务器替换的所有数据块（头数据块、元数据块和分片数据块）组成。如果客户端没有上传主头数据块，则分片数据块的编码方式要和当前的主头兼容。

E.2.3.3 JPP-stream

请求主体由客户端想要服务器替换的所有数据块（头数据块、分片头数据块、元数据块和分区数据块）组成。如果客户端没有上传主头或分片头数据块，则分区数据块的编码方式要与当前的主头和分片头兼容。

E.2.3.4 完整图像上传

请求主体包括表示客户端希望修改的那些样本的一个完整图像媒体类型。

在完整图像上传的情况下，请求可包括帧尺寸、区域尺寸和偏移量请求域。帧尺寸请求域应是图像参考网格的尺寸。在完整图像上传时，压缩方式并不需要和服务器上的逻辑对象兼容。如果上传图像的大小超出了区域尺寸请求域规定的范围，服务器应对其进行界限修正，从而满足区域尺寸请求域规定的范围。

E.3 服务器响应

E.3.1 概述

服务器应用附件D中的状态码和原因分析对上传请求进行响应。一些对图像上传有用的返回码和原因分析在下面小节中介绍。

E.3.2 201（已创建）

在收到一个上传请求后，如果服务器上已定义了一个新的资源，则服务器应发送该状态码。服务器在回复该请求前应该已经完成了创建。如果存在延时，服务器应返回202（已接受）来代替201（已创建）。

服务器在响应中应发送一个包括被更新资源的对象ID域的头。

不需要返回主体。

E.3.3 202（已接受）

如果上传创建了一个新资源但服务器还没有准备好提供服务，则服务器使用该状态码。服务器也可使用该状态码表示对当前资源的更新。

E.3.4 400（错误请求）

如果请求的格式不正确，或者查询中包含和上传不相容的请求域，或者在查询串中含有无法识别的域时，服务器就产生该状态码。

E.3.5 404（未找到）

如果服务器不能找到和对象ID相一致的被请求资源，则产生该状态码。

E.3.6 415 (不支持的媒体类型)

当支持上传但不支持请求的特定上传类型（如完整图像，JPT-stream或JPP-stream）时，可能产生该状态码。

E.3.7 501（不可实现）

如果服务器不支持上传或不支持与上传相关的某一选项时，可以采用此状态码。

E.4 服务器上的数据合并

E.4.1 更新图像

接收到上传数据后，服务器应创建逻辑对象的一个新版本并提供给客户端，客户端可以通过新的或旧的URL访问此新版本。但是，服务器不应使用旧的对象ID请求域来提供对任何合并或更新数据的访问。

如果客户端在上传请求中包含了对象ID请求域，但其指定的对象ID和服务器上资源的当前对象ID不一致，则服务器不应更新图像。这种不一致可能显示出用户编辑的图像版本是早先的版本，服务器上的图像已经被编辑过。服务器可以拒绝接受没有包含对象ID请求域的上传。这是一种可以防止不同用户同时对一个对象进行编辑的方法。服务器在提供编辑能力时，可以通过其它方法来关注这种对象闭锁问题。

JPIP客户端可能上传一个新图像的一部分通过是把对象ID设成0，或者用新的URL，或者用服务器没有的对象。服务器应为上传产生一个对象ID。客户端可以利用前面上传时服务器返回的对象ID，继续上传此新图像的其它部分。

E.4.2 JPT-stream

服务器如果接受了分片数据块数据，首先应移除所有对应的旧分片数据块数据，然后将新的分片数据块数据包含在码流中。如果分片的数目、维数或位置发生了变化，则不能进行更新：上传不能改变图像的结构。特别地，服务器不应接受对主头中包含PPM标记段的码流的分片数据块上传，除非客户端在上传时提供了一个新的主头。任何PLM或TLM标记段都应被删除或更新。对于一个新图像，JPT-stream主头数据块应被上传。

一个分片数据块的分片部分如何构成未作规定。客户端不必提供一个分片的所有分片部分，也不需要完成最后一个分片部分。服务器应对文件格式中受到影响的任何部分进行更新（如码流框的长度）。

当合并数据时，分片的数目或大小都不应发生变化，未被上传替换的数据和上传前具有相同的意义。

E.4.3 JPP-stream

服务器如果接受了分区数据块消息，首先应移除对应的旧分区数据块，然后将新的分区数据块数据包含进去。如果区域的数目，或区域标识符的含义，或者区域的位置或大小发生了变化，则不能对头进行更新。对于一个新图像，JPP-stream分片头数据块和主头数据块应被上传。

一个分区数据块的分区数据包如何构成未作规定。客户端不必提供一个分区的所有数据包，甚至不需要完成最后一个数据包。

当合并数据时，分区的数目或大小都不应发生变化，未被上传替换的数据和上传前具有相同的意义。

E.4.4 JPP-stream和JPT-stream元数据块

元数据块能被上传，替换已有的元数据块的内容。由于服务器具有将元数据分配到元数据块的划分控制能力，客户端应遵从服务器的元数据块结构。除了整个删除外，客户端不应改变元数据块中的占位符。当上传整个元数据块时，客户端能在原来的元数据块后面添加或者在原来的元数据块的框之间插入新的元数据方式，增加新的元数据。服务器应管理占位符和元数据块结构。这包括更新所有指向任何已经改变或改变所影响到的已逝元数据框的占位符。服务器应删除任何客户端已经删除的占位符所指向的元数据框。服务器在接受上传后，但在创建新的资源前，可以重新构造元数据。如果上传后文件中还有未使用的部分，应使用Free框填充那些部分。

E.4.5 完整图像上传

在可接受的完整图像上传的情况中，服务器应解压缩（若需要）上传的子图像，解压缩服务器上整个图像的某部分，在（解压缩的）空间域中替换那些像素并重新压缩所有受更新操作影响的分片或分区。

注 — 该技术需要服务器上更多的计算，但它排除了客户端以不兼容的方式（如小波变换的错误层数）使用压缩图像数据的可能性。

附 件 F

在HTTP上使用JPIP

（本附件是本建议书|国际标准的组成部分）

F.1 引言

本附件定义了在HTTP上使用JPIP进行请求和应答的方法。附件C中的JPIP请求参数被封装成合法的HTTP请求结构。附件D中的服务器响应（包括状态码，头，消息和响应编码）被封装成合法的HTTP响应。所有的请求和应答都按照HTTP标准进行编码。

注意本附件中的文本和实例描述的是在HTTP上使用JPIP。预计相同的绑定也能用于安全的HTTP上。

F.2 请求

F.2.1 请求的介绍

附件C定义了请求域。JPIP请求通过HTTP传送时，会作为一个HTTP的“GET”请求的查询串或者一个HTTP的“POST”请求的主体出现。因为一些HTTP系统限制在“GET”请求中提供的询问串的长度，所以对于长的JPIP请求，优先选则“POST”请求。

注1 — RFC 2616第5章中定义的HTTP请求如下：

```
Request = Request-Line                ; HTTP Section 5.1
          0*(( general-header         ; HTTP Section 4.5
            / request-header          ; HTTP Section 5.3
            / entity-header ) CRLF)   ; HTTP Section 7.1
           CRLF
           [ message-body ]           ; HTTP Section 4.3
```

注2 — HTTP中的Request-Line和Request-URI的定义如下：

```
Request-Line = Method SP Request-URI SP HTTP-Version CRLF
Request-URI = "*" / absoluteURI / abs_path / authority
```

注3 — RFC2396 定义：

```
absoluteURI   = scheme ":" ( hier_part / opaque_part )
hier_part     = ( net_path / abs_path ) [ "?" query ]
abs_path      = "/"  path_segments
```

F.2.2 GET请求

JPIP请求可作为一个HTTP请求提供给服务器。对于一个“GET”请求，该HTTP请求在以下几方面有限制：

— "Method"应为"GET"。

— "query"应为零或者用'&'分隔开的多个jpip-request-field请求域。

下面是在HTTP的“GET”请求中封装的JPIP请求的一个例子：

```
GET /images/kids.jp2?rsiz=640,480&roff=320,240&fsiz=1280,1024 HTTP/1.1
Host: get.jpeg.org
CRLF
```

另一个使用absolute而不是abs_path的等价的例子是：

```
GET http://get.jpeg.org/images/kids.jp2?rsiz=640,480&roff=320,240
&fsiz=1280,1024 HTTP/1.1
CRLF
```

注 — 本建议书|国际标准对绝对URI的命名空间的标识符（scheme）分量没有加限制。

F.2.3 POST请求

JPIP请求可经过HTTP的“POST”请求封装后提供给服务器。对于一个“POST”请求，该HTTP请求在以下几方面有限制：

- "Method"应为"POST"。
- "entity-body"应为零或者用'&'分割开的多个jpip-request-field请求域。
- "Content-type:"的头行应作为"entity-header"被包括，并且它要包含"application/x-www-form-urlencoded"的值。

下面是在HTTP的“POST”请求中封装的JPIP请求的一个例子：

```
POST /cgi-bin/j2k_server.cgi HTTP/1.1
Host: post.jpeg.org
Content-type: application/x-www-form-urlencoded
Content-length: 62
CRLF
target=/images/kids.jp2&rsiz=640,480&roff=320,240&fsiz=1280,1024
```

F.2.4 上传请求

上传请求是一个的合法的HTTP请求，限制如下：

- "Method"应为"POST"。
- URL应该包括上传的查询域。
- Content-type应是主体的图像类型：image/jpt-stream、image/jpp-stream或者一个完整的图像媒体类型。

JPIP的上传请求的一个例子如下：

```
POST /images/kids.jp2?rsiz=640,480&roff=320,240&fsiz=1280,1024 HTTP/1.1
Host: post.jpeg.org
Content-type: image/jpt-stream
CRLF
```

F.3 会话的建立

一个基于会话的HTTP会话是通过在新建通道域中用“http”的值建立的，即"cnew=http"为请求的一部分。这个请求通常由HTTP传送。该请求中可能包含一个视窗请求，成为新的通道中的第一个请求。对该请求的响应在和请求相同的连接上被返回。

客户端可能会打开一个HTTP连接并发送一个包含HTTP头为"Connection: keep-alive."的请求。这对有效的会话来说是有用的，但它并不是建立一个会话的必要和充分条件。单个HTTP连接可能用于不同对象、不同通道的流量，甚至是非JPIP流量，如，HTML文件的请求。作为会话的一部分，JPIP请求可能在不是用于请求和发送新通道的HTTP连接上到达，尽管并不鼓励这样。

F.4 响应

F.4.1 引言

附件D中描述的响应的每个分量都会被封装成合法HTTP响应的一部分。

注 — RFC 2616 第6章中定义的HTTP响应如下：

```
Response = Status-Line                    ; HTTP Section 6.1
           0*(( general-header            ; HTTP Section 4.5
             / response-             ; HTTP Section 6.2
             / entity-header ) CRLF)      ; HTTP Section 7.1
            CRLF
            [ message-body ]              ; HTTP Section 7.2
```

在HTTP上传送的JPIP响应是合法的HTTP响应，因此对这些HTTP响应的某些部分的进一步的限制，在下面的小节中描述。

F.4.2 状态码和原因分析

D.1.3中列出的所有的状态码可直接作为HTTP的状态码使用。此外在HTTP上提供JPIP的服务器可能使用任何被认为是有用的HTTP状态码，如402。

D.1.3中列出的所有原因分析可直接作为HTTP的原因分析使用。原因分析应适于状态码。提供在HTTP上提供JPIP的服务器可以使用任何被认为是有用的HTTP的原因分析，例如：需要付款。

F.4.3 头信息

F.4.3.1 JPIP头

D.2中的头行应不加修改地作为“实体头”（"entity-header"）包含在HTTP响应中。

F.4.3.2 HTTP Accept头的使用

在HTTP上提供JPIP的服务器可能用请求中发现的HTTP "Accept:"头行确定JPIP响应的类型。如果请求中包含"type="的查询参数，则应答的类型应是类型参数中列出的类型之一；如果请求中既包括"type="的查询参数也包括"Accept:"头，服务器用"Accept:"头中指定的优先权选择"type="询问参数中指定的类型中选择应答的类型；如果在请求中没有"type="的询问参数，服务器将在"Accept:"的类型列表中选择一个被可能的JPIP服务器支持的返回类型。

F.4.3.3 缓存控制头的使用

注意HTTP代理中的缓存不同于JPIP中的缓存和缓存模型。

任何带有新建通道请求域的JPIP请求都是会话的一部分，这种响应一般不能由HTTP代理服务器缓存。同样，任何包含新建通道响应头的响应也是会话的一部分。这两种情况下，服务器响应应包含一个值为"no-cache"的HTTP"Cache-Control:"头。

F.4.3.4 内容类型头的使用

在HTTP提供上JPIP的服务器应该包含一个"Content-type:"头，指示主体中数据的类型，通常该类型为image/jpp-stream或image/jpt-stream。

F.4.3.5 重定向头的使用

HTTP的重定向头可能用于告知客户端资源已经被移动或者应该从另一个不同的主机访问该资源。

注意JPIP响应还定义了重定向的方法。应优先选择会话中的JPIP响应。

F.4.4 主体

附件D中的消息应包含在HTTP的响应主体中。注意HTTP响应应有一个确定响应长度的机制。如果服务器不想中断响应，那么它可能用HTTP头行"Content-Length"来提供这个信息。提供长度信息的首选方法是用HTTP头行"Transfer-Encoding: chunked"，然后以块的形式提供主体，块的大小由服务器确定并在每块前详细说明。不鼓励通过关闭HTTP连接作为响应结束指示的方法。

F.5 其它的HTTP特征

F.5.1 HTTP HEAD方法的使用

JPIP的客户端和服务器不要求使用或支持HTTP的"HEAD"方法。一个选择实现"HEAD"方法的服务器应如RFC 2616 9.4节中指定的那样。特别地，“除了服务器在响应中不应返回消息主体外，HEAD方法和GET是等同的”。

客户端可能发现发送HTTP"HEAD"请求作为一种确定服务器是否将修改附件D中规定的任何请求参数的有效途径。客户端不能发送带有缓存模型询问域的HTTP"HEAD"请求，因为这可能会导致服务器更新它的缓存模型。

注意如果客户端希望在不接收响应的情况下更新服务器缓存模型，它可能要使用最大响应长度请求域。

服务器可能拒绝任何或全部的"HEAD"请求。不像通常的HTTP"HEAD"请求那样服务器相对需要耗费很少的精力就能完成响应，一些JPIP服务器实现可能不得不从逻辑对象的多个位置获得数据，计算响应的性质，然后丢弃响应的主体对"HEAD"请求作出响应。

F.5.2 HTTP OPTIONS方法的使用

JPIP的客户端和服务器不要求使用或支持HTTP的"OPTIONS"方法。

F.5.3 Etag的使用

注意HTTP定义了实体标签（Etag）机制，它在表示资源的改变方面和JPIP的对象ID请求域是相似的。如果一个实体标签和一个对象ID都和同一个资源关联，建议当对象ID改变时，HTTP定义的实体标签也相应改变。

F.5.4 分块传送编码的使用

因为包含压缩数据的响应会非常大从而需要很长的传送时间，所以能够在数据传输中停止是很重要的。除非已经指定"Transfer-Encoding: chunked"，否则HTTP请求应在"Content-Length:"头中指定主体的完整长度或者通过关闭连接来表示数据的结束。这些在交互式协议中都是不可取的，因为停止当前的响应并在同样的连接上为一个新的响应发送更多的数据是必要的。

注1 － RFC 2616的19.4.6节中提供了消除分块传送编码的算法。

注2 － 用除HTTP以外的其它协议传送JPIP时，分块传送编码是很有用的。

F.6 HTTP和长度请求域（资料性）

有了HTTP返回通道，服务器不从客户端收到连续的反馈并且很容易将大量数据输入到管道里，这些数据在处理新窗口中的任何数据之前被全部接收。为了维持响应，客户端应该使用最大响应长度请求域来调节通信流量从而保持响应。客户端通常需要运行它们自己的流量控制算法来根据网络环境的变化调节响应长度。

附 件 G

用HTTP请求和TCP返回使用JPIP

（本附件是本建议书|国际标准的组成部分）

G.1 引言

除了由新建通道 ("cnew")请求域（见C.3.3）和新建通道 ("JPIP-cnew")响应头（see D.2.3）描述的通道请求需要对特定传送的细节进行通信外，在客户端请求和服务器响应的潜在传送机制方面JPIP协议本身是中立的。本建议书|国际标准定义了两种特定的传送方式，它们在新建通道请求值中分别由字符串"http"和"http-tcp"标识。本附件提供了第二种传送方式的细节，该传送方式在本文中用HTTP-TCP标识。第一种传送方式在附件F中进行了描述，在本文中用HTTP标识。

HTTP-TCP传输使用和HTTP传输完全相同的机制将客户端请求发送到服务器上并接收服务器响应头和状态码。不过，服务器的响应数据（不是响应头）是通过一个辅助的TCP连接传送的。在这个辅助TCP连接上传送的信息和在纯粹的HTTP响应上作为实体主体上传送的信息是一样的，唯一不同是这些信息被构造成块，每块上面标有块序列号。

客户端通过辅助TCP连接的返回通道将所到达的块的序列号回送给服务器，以此明确地确认该块已经到达。HTTP-TCP传送的原则性优点之一就是通过客户端的确认机制，服务器收到增加的关于它的响应数据块已被收到的通知。这种方式使服务器就能够像保持响应和网络效率一样控制数据流动。

所有从HTTP上传送的请求都应按照的HTTP标准进行编码。

G.2 客户端请求

请求将完全和HTTP请求一样在原来的通道上传输。它们和附件F种描述的HTTP传送方式通道中发送的请求有完全相同的格式。特别地，HTTP的"GET"和"POST"请求都可以使用。

G.3 会话的建立

G.3.1 通道的建立

通过向JPIP服务器发送一个包含新建通道请求域（见C.3.3）的请求，建立一个新的通道。举例来说，这个请求可以使用HTTP发送，尽管虽然这种请求还用任何合适的传输机制发送给特定JPIP服务器。如果服务器响应（通过D.2.3中的新建通道响应头）表明已经建立了一个用HTTP-TCP传输的新通道，那么客户端应使用新建通道响应头中返回的辅助端口号建立一个辅助的TCP连接。此外，包含新建通道请求域的请求应看成就其在新创建的HTTP-TCP传送通道中传送一样，也就是说一旦辅助的TCP连接建立，这个请求产生的响应数据应通过这个辅助的连接返回。

为了建立辅助的TCP连接，客户端要向由新建通道响应头指定的服务器主机发送一个TCP连接请求，在新建通道响应头所指定的端口上。然后客户端马上发送一个单行的ASCII文本，它包含新的通道id字符串，后面是两个连续的CR-LF对。这是在辅助TCP连接上传送的唯一的基于文本的通信。

然后客户端等待接收从辅助TCP连接上返回的服务器响应。这个响应数据不能为空，因为在HTTP-TCP传送通道上发送的每个请求都应该有一个至少包括EOR信息的响应数据流（见D.3）。G.4中有关于它的更多信息。

G.3.2 服务器对响应数据的组帧

所有由服务器通过辅助TCP连接发送的响应数据都应该组帧成块。每块包括一个8字节的块头，接着是块的主体它保存了服务器响应数据，如图G.1所示。块头的前两个字节保存了一个无符号按big-endian字节顺序存储整数，用来表示包括长度字在内的整个块的全部长度。块头中剩余的6个字节在本建议书|国际标准中没有定义。它们可用于附加的服务器特定的信令。客户端在该块的确认消息中要返回本块头的全部的8个字节数据。

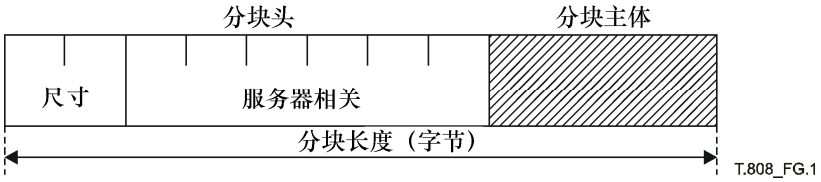

图 G.1－http-tcp 连接上响应数据的结构

G.3.3 客户端对服务器响应块的确认

客户端在辅助的TCP连接上收到服务器的响应数据块后，应该通过TCP连接的返回通道，将8个字节的块头以未经组帧的数据流形式发送回服务器。每个接收到的数据块这样依次被确认。

G.4 服务器响应

为了响应每个客户端请求，服务器在原来的通道上回复给客户端一个HTTP应答段。这个应答段包含状态码，原因分析，所有相关的JPIP响应头以及任何适当的HTTP响应头。不过，没有响应数据通过原来的通道返回。所以在HTTP-TCP响应中不应有HTTP实体主体。HTTP的"Content-length:"和"Transfer-encoding:"响应头都不能使用。

响应数据本身以G.3.2中描述的方式组帧成块并通过辅助的TCP通道传输。因为HTTP-TCP传输只用于会话，所以图像返回类型只有JPP-stream和JPT-stream，返回数据总是由JPP-stream或JPT-stream的消息序列组成。

每个请求产生的响应数据都应由许多完整的块组成，也就是说没有块可能包含两个不同请求产生的响应数据。

每个请求的响应都应该以EOR消息结束（见D.3），否则响应数据为空的情况。EOR消息被认为是响应数据的一部分并且和实际的JPP-stream 和JPT-stream消息一起组帧。

这意味着在一个以HTTP-TCP为传输的JPIP的通道上发送的每个请求都会使服务器产生至少一个非空的响应块，每个响应产生的最后一个块都以EOR消息结束。

注意实际上并不要求用HTTP-TCP传送的响应块在消息边界上对齐。

G.5 TCP和长度请求域（资料性）

有很少或没有理由在一个TCP返回通道上使用最大响应长度请求域，在该通道上服务器能够仔细地调节它对客户端的响应数据流以保持响应性。

附 件 H

使用其他传输使用JPIP

（本附件是本建议书|国际标准的组成部分）

H.1 引言

除了附件F中描述的"http"传输和附件G中描述的"http-tcp"传输外，本建议书|国际标准没有定义任何其它特定的传输协议。本附件的目的是为在不可靠传输中部署JPIP的配置的提供指导原则，并提供一个可用于各种传输的通用的方法。

为了详细说明这个通用的方法，有必要将通信分为两个逻辑传输连接，分别称为“请求连接”和“数据连接”。我们认为每个逻辑连接都提供了一个前向传送的通信路径和一个反向传送的通信路径。这些路径的作用如下：

— 前向请求连接路径用于从客户端传送JPIP请求给服务器。

— 反向请求连接路径是服务器用于确认收到请求并向客户端回送响应头。

— 前向数据连接路径用于从服务器传送JPIP流消息给客户端。

— 反向数据连接路径是客户端用于确认收到服务器发来的JPIP流消息。

读者可能发现这些作用和附件G中描述的"http-tcp"传输使用的两个TCP通道的前向和反向通信路径的作用是一致的。的确，本附件的内容可以被看作是"http-tcp"传输在不可靠传输上的扩展。然而，注意虽然本附件描述了两个不同的逻辑连接，但是没有理由说明为什么通信不能在单一传输连接上进行。

最后，假定每个逻辑连接提供下面两种服务之一：

a） 可靠的面向流的服务，如TCP所提供的服务。

b） 不可靠的面向数据包的服务，如UDP提供的服务。这种情况下，数据包可能乱序到达或者根本没有次序，应明确实现确认握手从而确定某个数据包是否成功地到达。

本附件中假想了两个场景。在第一个场景中，请求连接路径假设能够提供可靠的面向流的服务而数据连接路径是不可靠的。在第二个场景中，请求连接路径和数据连接路径都是不可靠的。我们依次考虑这两个场景。

H.2 可靠的请求和不可靠的数据

本节中，请求连接是可靠的，意味着请求无损的依次到达服务器，客户端同样无损地依次接收到服务器的响应。在这种情况下，和"http-tcp"协议中一样，请求域和响应头被准确地传送，实际上推荐HTTP传输请求和响应头。例如这种形式的传输协议我们可以命名为"http-udp"，但这已超出本附件的范围。

包括EOR消息（见D.3）的JPIP流消息应该划分成数据包并通过不可靠的数据连接（例如UDP）传送。客户端应通过将数据包的头发送回服务器以确认该数据包已收到。这使得服务器能够估算网络状况并确定是否应该重发数据包。当客户端视窗改变时，服务器可能决定不重发没有收到确认的数据包。

构建这种类型的传输协议时我们应该遵守下面的通用的指导原则：

a） 每个请求应包括一个请求ID请求域（见C.3.5）。

b） 对每一个请求都应有一个相应的EOR消息，即使是对该请求不发送JPIP流消息进行应答的情况下也是如此。这个要求在"http-tcp"传输中同样适用。

c） 由服务器建立的每个数据连接应该包括全部的JPIP流消息和/或EOR消息。此外，每个数据包的第一个流消息应该包含一个完整的头，它不依赖于码流标识符的副本或先前消息的类别码分量。

d） 在一个数据连接包中发现的所有JPIP流消息（不一定是EOR消息）都应属于对同一请求的响应，并且相应的请求ID应该在该数据包的头中进行编码。

e） EOR消息可能位于和请求具有相同请求ID值的数据包的末尾，作为该请求的响应即将结束，或者位于一个或多个连续的EOR消息块内，这个块是在带有该请求ID值的最后的一个数据包后的第一个数据包的起始位置。这种策略使对应于一个或多个连续的空响应（如由于前面的请求被先占了）的EOR消息能够打包在随后的非空响应中的第一个数据包内。

f） 除了请求ID值外，每个数据包的头还应该包括一个包序列号。和任意特定的请求ID值相关联的第一个数据包从0开始顺序计数。具有相同请求ID值的后续的数据包具有连续的序列号。该策略使客户端能够识别任何因包丢失而没收到任何EOR消息。重要的是客户端能够将请求和响应数据联系起来，从而的缓存模型操作声明对服务器的影响和它们本身的缓存保持同步。

g） 客户端收到每个数据包后应该通过响应数据连接路径向服务器发送确认消息。每个确认消息应该包含所收到数据包的头的副本，也可以包含其它附加信息。客户端在创建确认数据包时可以自行地将确认消息聚合成几个数据包。但是过多的聚和可能会影响可靠性，而服务器依此来估算网络的统计性能。

h） 服务器不强制对任何没有确认的数据包都进行重发，而且客户端也不应该期望丢失的包被重发。例如，智能的服务器可能依据它们和当前视窗的相关性来选择是否重发没有被确认的数据包。

H.3 不可靠的请求和不可靠的数据

本节主要考虑请求和数据连接都不可靠的传输。对于数据连接的指导方针和H.2中描述的完全相同，H.2中数据的传输也是不可靠的。在不可靠的请求连接的情况下，很可能丢失一个或多个请求或者请求无序地到达服务器。JPIP很适合于处理这种情况，因为当新的请求到达时服务器可自由的先占前面的请求。

除了H.2中列出的不可靠数据连接的指导原则外，处理不可靠请求还要遵循以下通用的指导原则：

a） 每个请求包应包含一个头来标识请求ID的值。

b） 每个请求包都应包含一个序列号，携带足够的信息来确定和同一个请求相关联的所有数据包是否全部被收到。

c） 很多情况下当新的请求到来时，服务器能够简单地忽略丢失的请求包。为了做到这一点，服务器只需在数据连接上发送EOR消息，指示丢失的请求被直接先占了。没有必要发送确认消息作为对请求包的响应。因为部分或全部请求包已经丢失，所以也没必要发送任何响应头来作为对被直接先占的请求的回复。

d） 对于每个完整到达服务器的请求，服务器应该发送一个或多个响应包，标识请求ID并包括所有的响应头。即使在后续请求的响应都已发出后这个请求才到达（例如：原因是这个请求的一些包被过度延迟）的情况下，也要发送响应包。这为客户端提供了确定服务器是否收到某个重要的请求的机制。

e） 某些类型的请求应被服务器处理以免和客户端失去同步。其中最重要的请求是那些包括减的缓存模型操作域的请求。为了使服务器在不用全部串行化请求流的情况下，就能检测出这种类型的请求，请求包的头应该包括下面的两个域：

1） 一个标记用来指示这个数据包是否属于在处理后续请求之前要预先处理的请求。

2） e1中所指的标记已置位的最新请求的请求ID。

如果服务器没有收到一个或多个e1标记置位的请求包（也就是，具备e2条件的请求到达，而e1中标记的请求丢失），服务器将处于空闲状态直到客户端重发数据包。

H.4 请求和响应语法

为JPIP协议设计新的传输时，应该遵循附件C和D中描述的请求和响应的语法。然而，允许对各种请求域和响应头开发等价的二进制表示法。

H.5 会话的建立

除了本建议书|国际标准描述的标准的"http" 和 "http-tcp"传输协议外，新建通道请求域（见D.2.3）及相应的响应头也可用于建立其它传输协议的通道。出于这个目的，新传输协议的名字应该在附件J中说明的注册委员会进行注册。为新传输创建通道的流程应该遵循"http-tcp"中概述相同的通用规约。特别地，创建新通道请求的响应头应该通过用来创建该通道的传输协议回送，而响应数据应使用新通道传输进行传送。

附 件 I

在JPIP中编制JPEG 2000文件的索引

（本附件是本建议书|国际标准的组成部分）

I.1 引言（资料性）

ITU-T T.800建议书| ISO/IEC 15444-1:2004和其它一些标准定义了一系列JPEG 2000文件格式。该系列使用相同的语法，这个语法的基本要素是称为框的容器。本附件定义了包含索引信息的新的文件格式框，在JPEG 2000系列文件中包含索引信息可能有助于在JPIP系统中那些文件的部署，通过使文件阅读器能够在文件中渐进的定位那些被请求用来重建图像的元素。

特别地，这些框可能对以下方面有用：

— 服务器侧JPIP协议的实现；

— 客户端可以使用一个更简单的协议远程存取图像，该协议允许存取文件中指定字节范围的数据。

本附件定义对应于文件层信息和码流信息的索引框。这些框可分类如下：

— 码流索引（cidx）超级框编制对应于JPP-stream和JPT-stream中的主头、分片头、分片和分区数据块类型的码流信息的索引。它包含一个指向被索引码流的码流发现（cptr）框，一个概括其余内容的清单（manf）框和索引表框，索引表框包括头索引表（mhix）框，分片部分索引表（tpix）超级框，分片头索引（thix）超级框，分区数据包索引表（ppix）超级框和数据包头索引表（phix）超级框。索引表框对应于不同类型的码流数据，这些码流数据由附件A中定义的JPP-stream和JPT-stream中的数据块类型表示。这些索引表框是超级框，包含列出实际的码流元素的片断分组索引（faix）框或头索引表。每个头索引表、分区数据包和数据包头索引表超级框也包含一个清单框。

— 文件索引（fidx）超级框编制对应于JPP-stream和JPT-stream中元数据块的文件层信息的索引。除非它索引文件的最高层，在这种情况下，称其为根文件索引框；否则，它包含一个指向被索引的超级框的文件发现（fptr）框。它可能包含表示被索引的文件或超级框内容的代理（prxy）框。

— 索引发现（iptr）框指向一个根文件索引，使得能发现它的位置。

图I.1给出一个包含JPIP索引框的JPEG 2000例子文件：

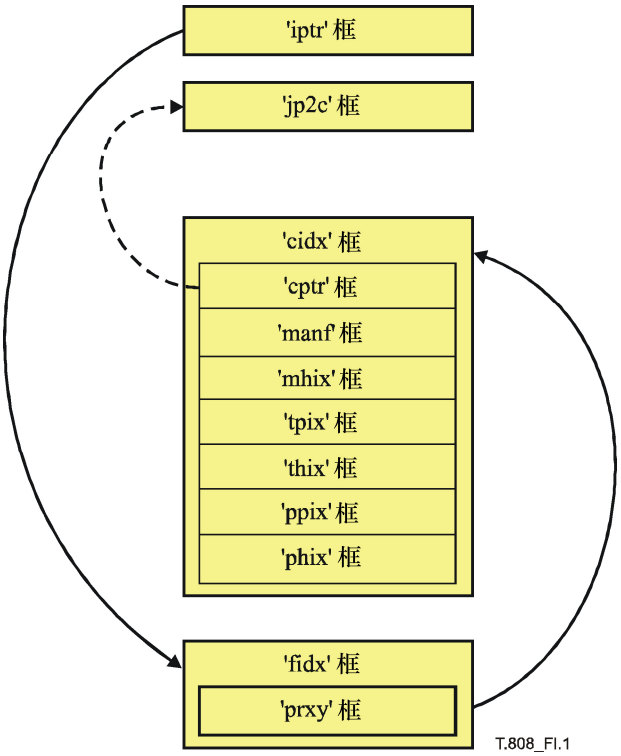

图 I.1－包含 JPIP 索引框的 JPEG 2000 例子文件的一部分

I.2 在JPEG2000文件格式兼容表中标识JPIP索引框的使用

包含一个或多个本建议书|国际标准中定义的索引框的文件在文件类型框中（ITU-T T.800建议书|ISO/IEC 15444-1的附件I中定义）可能包含值为'jpip' (0x6a70 6970)的CL[i]域。

I.3 已定义的框

I.3.1 概述

表I.1列出了本建议书|国际标准中定义的所有框。对每个框的位置和限制的信息，可参见定义该框的相关节。

表I.1是资料性的。每个框的标准定义包含在该表格提及的各节中。

表 I.1－已定义的框（资料性）

框名	类型	超级框	注释
码流索引框 (I.3.2)	'cidx' (0x6369 6478)	是	该框包含 JPEG 2000 码流的索引信息。
码流发现框 (I.3.2.2)	'cptr' (0x6370 7472)	否	该框指向一个 JPEG 2000 码流。
头索引表框 (I.3.2.4.3)	'mhix' (0x6D68 6978)	否	该框表示码流中主头的标记段的索引或一个分片的分片部分头。
分片部分索引表框 (I.3.2.4.4)	'tpix' (0x7470 6978)	是	该框表示码流中的每个分片部分的位置和长度。
分片头索引框 (I.3.2.4.5)	'thix' (0x7468 6978)	是	该框表示码流中每个部分的位置和长度，这些部分对于正确解码分区包数据以构造每个分片的分片头是必须的。
分区数据包索引表框 (I.3.2.4.6)	'ppix' (0x7070 6978)	是	该框表示码流中数据包的位置和长度。
数据包头索引表框 (I.3.2.4.7)	'phix' (0x7068 6978)	是	该框表示码流中的数据包头的位置和长度。
清单框 (I.3.2.3)	'manf' (0x6D61 6E66)	否	该框概括了紧跟在它后面的那些框，它包括的框或文件和清单框位于同一个等级。
片断分组索引框 (I.3.2.4.2)	'faix' (0x6661 6978)	否	该框表示码流中元素的位置和长度。
文件索引框 (I.3.3)	'fidx' (0x6669 6478)	是	该框可被用来寻找文件中其它的索引及任意数据。
文件发现框 (I.3.3.2)	'fptr' (0x6670 7472)	否	该框指向一个被索引的框。
代理框 (I.3.3.3)	'prxy' (0x7072 7879)	否	该框以文件索引框的形式表示一个框在文件中不同的位置。
索引发现框 (I.3.4)	'iptr' (0x6970 7472)	否	该框指向文件的根文件索引。

I.3.2 码流索引框（超级框）

I.3.2.1 概述

码流索引框包含关于JPEG 2000码流的索引信息。码流索引框的类型是'cidx' (0x6369 6478)。码流索引框的内容如下所示（图I.2）：

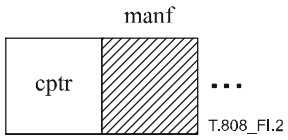

图 I.2－码流索引框内容的组织方式

cptr： 码流发现框。这个框指向由码流索引框索引的码流。它的结构在I.3.2.2中指定。

manf： 清单框。这个框概括了在它之后的索引表，这些索引表位于码流索引框中。它的结构在I.3.2.3中指定。

I.3.2.2 码流发现框

码流发现框指向一个JPEG 2000码流。码流发现框的类型是'cptr' (0x6370 7472)。码流发现的内容如下所示（图I.3）：

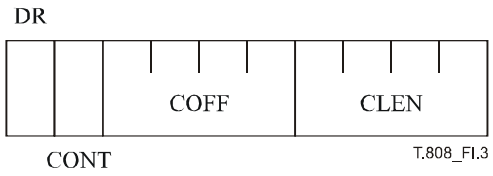

图 I.3－码流发现框内容的组织方式

DR： 数据引用。这个域表示码流的位置，或者代表码流的片断表框的位置。如果值为0，当前文件中存在该码流或代表它的片断表框。否则，该数值标识一个当前文件Data Reference框的入口。在这种情况下，由DR标识的数据引用入口指出了包含码流或片断表框的资源。这个域以2字节、按big endian字节顺序存储的无符号整数。

CONT： 容器类型。这个域以2字节、按big endian字节顺序存储的无符号整数。表I.2描述了在本建议书|国际标准中定义的值。

COFF： 码流偏移量。这个域表示码流或片断列表框相对于文件或由DR标识的资源的开始的的合适位置。这个域以8字节、按big endian字节顺序存储的无符号整数。

CLEN： 码流长度。这个域表示码流或片断列表框的长度。这个域以8字节按big endian字节顺序存储的无符号整数。

表 I.2－容器类型的值

CONT	意义
0	整个码流以连续范围的字节形式出现在它的文件或资源中。在这种情况下，这里给出的偏移量和长度数值指的是码流本身。注意码流可能在一个连续码流框中，但是偏移量和长度的值指的是码流本身，从 SOC 标记开始，在 EOC 标记后马上结束。
1	码流是分段的，位置和长度的值指的是片断列表框（包括它的框头），这个片断列表框描述码流的每个片段的位置和长度。注意所有后来的位置和长度都是相对于码流的开始的，就像在重建所有由片断列表框标识的片段之后出现的情况。
所有其它值	保留供 ISO 使用。

I.3.2.3 清单框

清单框概括了立即或者连续跟在它后面的那些框，它包括的框或文件和清单框位于同一个等级。

注 — 清单框可被用来帮助随机地获取下面的这些框，如在码流索引框中在它之后的一些索引框。

清单框的类型是'manf' (0x6D61 6E66)。清单框的内容如下所示（图I.4）：

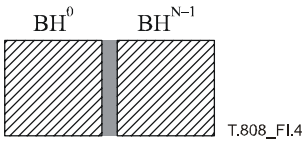

图 I.4－清单框内容的组织方式

BH^i： 框头。这个域包含紧接清单框之后的第*i*个框的完整的框头。如果包含在那个框头中的LBox域的值是1，则这个域的长度是16个字节，否则是8个字节。

框的数量*N*由清单框的长度决定，这些框的头包含在清单框内。当在分区数据包索引表框或数据包头索引表框中使用时，*N*是码流分量的数量。

在码流索引框，分片头索引表框，分区数据包索引表框或数据包头索引表框中，清单框应该包含所有在它之后、直到这个容器框结束的框。

I.3.2.4 索引表

I.3.2.4.1 概述

码流索引框可包含下述每个类型的码流数据的索引表：主头，分片部分，分片头，（分区）数据包和数据包头。每个索引表是一个不同类型的框。在码流索引框中每种类型的表应只出现一次。

分片部分索引表，分区数据包索引表和数据包头索引表框都是包含片断分组索引框的超级框。分片头索引表框是包含头索引表框的超级框。下面我们首先定义片断分组索引框，然后定义索引表框。

I.3.2.4.2 片断分组索引框

片断分组索引框列出了码流中各个元素的位置和长度。它应用于分片部分索引表，分区数据包索引表和数据包头索引表超级框。

片断分组索引框的类型是'faix' (0x6661 6978)。片断分组索引框的内容如下所示（图I.5）：

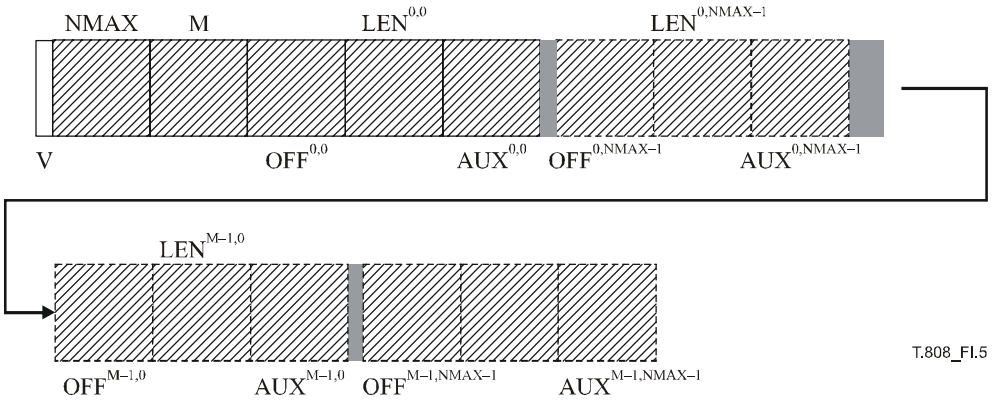

图 I.5－片断分组索引框内容的组织方式

V： 版本。这个域被编码为1字节无符号整数。表I.3描述了在本建议书|国际标准中定义的值。

NMAX：在矩阵的任意行中有效元素的最大数目。当在一个码流索引表内使用时，**NMAX**是指定给任意分片的元素的最大数目。

M： 矩阵行的数目。当在一个码流索引表内使用时，M是分片的数目。

OFF^{i,j}： 偏移量。这个域标识矩阵的第i行的第j个元素以字节计算的偏移量（相对于码流的开始）。

LEN^{i,j}： 长度。这个域表示矩阵的第i行的第j个元素以字节计算的长度。

AUX^{i,j}： 辅助的。这个域表示关于矩阵的第i行第j个元素的辅助信息。除非包含这个框的超级框允许，否则这个域的值为0。这个域的所有非0值被保留。

虽然片断分组索引框中的矩阵的所有行应该存储NMAX个元素，但是由那个矩阵描述的对象可有更小数目的元素需要规定。在这种情况下，任意的i行包含*J*个有效元素，这里J小余NMAX，OFF^{i,J}到OFF$^{i,NMAX–1}$和LEN^{i,J}到LEN$^{i,NMAX–1}$的值应为0。

表 I.3－版本值

CONT	意义
0	NMAX，M 和所有 $OFF^{i,j}$ 和 $LEN^{i,j}$ 域是 4 字节、按 big endian 字节顺序存储的无符号整数并且 $AUX^{i,j}$ 域不出现。
1	NMAX，M 和所有 $OFF^{i,j}$ 和 $LEN^{i,j}$ 域是 8 字节、按 big endian 字节顺序存储的无符号整数并且 $AUX^{i,j}$ 域不出现。
2	除了 V 的所有域被编码是 4 字节、按 big endian 字节顺序存储的无符号整数。
3	NMAX，M 和所有 $OFF^{i,j}$ 和 $LEN^{i,j}$ 域是 8 字节、按 big endian 字节顺序存储的无符号整数并且 $AUX^{i,j}$ 域是 8 字节、按 big endian 字节顺序存储的无符号整数。
所有其它值	保留供 ISO 使用。

I.3.2.4.3 头索引表框

头索引表框索引码流的主头或分片的分片部分头，指出整个主头长度或第一个分片部分长度和头中标记段的位置和长度。应该包括所有标记段都，除了SOT标记段以外，SOT标记段在仅包含SOT和SOD的分片部分头中可被忽略。标记段不须按照它们在码流中出现的顺序进行排列。头索引表框可能仅出现在一个码流索引框中。在顶层，它一次且仅一次索引码流。在分片头索引表框内，它索引分片部分头。

注 － 这样做的目的是提供一种有效方法跳过头中的指针信息，这些指针信息对于有效的浏览文件并不需要但会增大头。在头索引表框中，使用相同的标记码连续地列出多个标记段，可以使读者跳过它们不感兴趣的成组的标记段。

头索引表框的类型是'mhix' (0x6D68 6978)。头索引表框的内容如下所示（图I.6）：

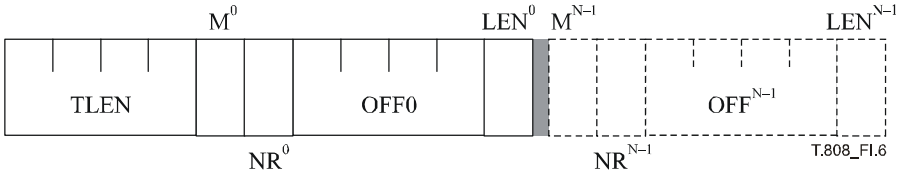

图 I.6－头索引表框内容的组织方式

TLEN：长度。当头索引表框索引一个主头时，这个域表示这个主头的总长度。当头索引表框索引分片部分头时，这个域表示第一个分片部分头的总长度。这个域的值是8字节、按big endian字节顺序存储的无符号整数。

M^i：标记码。这个域表示在这个框中列出的第i个标记段的标记码。这个域的值是2字节、按big endian字节顺序存储的无符号整数。

NR^i：剩余数量。这个域指出在这个列表中，（至少）NR^i个有相同标记码M^i的标记段被在第i个标记段之后立即或者连续的列出。这个域的值是2字节、按big endian字节顺序存储的无符号整数。

OFF^i：偏移量。这个域以字节为单位表示这个列表中第i个标记段的标记段参数（包括长度参数但不包括这个标记本身）相对于码流的开始的偏移量。这个域的值是8字节按big endian字节顺序存储的无符号整数。

LEN^i：长度。这个域以字节为单位表示这个列表中第i个标记段的标记段参数（包括2字节的长度参数但不包括这两个字节标记本身）的长度。

头索引表框中标记段的数量*N*由头索引表框的长度决定。

I.3.2.4.4 分片部分索引表框（超级框）

分片部分索引表框索引码流中的每个头部分的位置和长度，其中每个分片部分以它的SOT标记开始，以最后一个数据包结束。

分片部分索引表框的类型是'tpix' (0x7470 6978)。分片部分索引表框的内容如下所示（图I.7）：

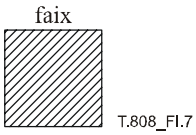

图 I.7－分片部分索引表内容的组织方式

faix： 片断分组索引框。这个框列出了码流中所有分片部分的位置和长度。它的结构在I.3.2.4.2中指定。这个表的第m行对应于码流的第m个分片。这个行的入口按码流顺序保留了相应分片的所有分片部分的位置和长度。如果片断分组索引框的Version等于2或3，辅助域表示每个分片部分的最小n，这个n使得对于所有分量$(N_L - n)$非负，当这个分片部分与同一分片中的所有前面的分片部分结合时，分辨率级别$(N_L - n)$和所有低分辨率级别已经完成，其中N_L是分解层次的数目，它可随分量而改变。如果任意分量的分辨率级别都没有完成，辅助域的值等于1加上所有分量的N_L的最大值。当所有分量的所有分辨率都已完成时，取值为0。因为分辨率没有必要按顺序出现在分片中，一些大于由辅助域指定的数值的分辨率级别可能已经完成。

I.3.2.4.5 分片头索引表框（超级框）

分片头索引表框索引每个分片的分片头，用于正确解码分区数据包数据。

分片头索引表框的类型是'thix' (0x7468 6978)。分片头索引表框的内容如下所示（图I.8）：

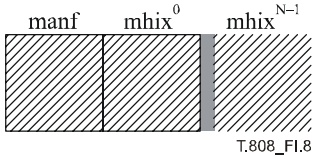

图 I.8－分片头索引表框内容的组织方式

头索引表的数量，N，是分片的数量。

manf： 清单框。这个框概述由分片头索引表框内的$mhix^i$指定的框。它的结构在I.3.2.3中指定。

$mhix^i$： 头索引表框。这个框索引第i个分片的分片部分头。它的结构在I.3.2.4.3中指定。

I.3.2.4.6 分区数据包索引表框（超级框）

分区数据包索引表框索引码流内的数据包。分区数据包索引表框的类型是'ppix' (0x7070 6978)。分区数据包索引表框的内容如下所示（图I.9）：

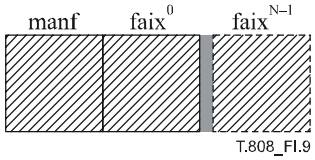

图 I.9－分区数据包索引表框内容的组织方式

片断分组索引的数量，N，应该不大于码流分量的数量。

manf： 清单框。这个框概括了由分区数据包索引表框内的$faix^i$指定的框。它的结构在I.3.2.3中指定。

faix^{i}： 第i个片断分组索引框对应于码流中第i个图像分量。这个表的第m行对应于码流的第m个分片。这个行的入口保存相应分片分量中所有数据包的位置和长度。数据包在它们各自的区域中以图层的顺序递增、连续出现；区域出现的顺序和在A.3.2.1中定义的序号s有关。然而，数据包的固定顺序没必要和码流中在任意COD/POC中指定的顺序相同。

如果数据包头被封装在PPM或PPT标记段中，在片段矩阵中对应的入口仅指数据包体的位置和长度，因为它出现在自己分片部分的主体内。不存在的数据包（或者因为在相同的矩阵中，相关的分片分量比另一个分片分量含有更少的数据包；或者因为在数据包应该存在的点以前，码流已经被截短了）的入口应该将它们的位置域设为0。指向一个主体是空的并且头由一个字节的数据0x80组成的数据包的入口可用长度值为0来标识。这种数据包在JPEG 2000码流中经常出现；应用程序可以避免明确地取得这种内容可被预测的数据包。如果相应的COD标记段指出EPH标记在某些分片的每个数据包头后会出现，在那个分片中，这个特别长度值0应该被解释为这个数据包由0x80字节和接下来的EPH标记组成。

I.3.2.4.7 数据包头索引表框（超级框）

数据包头索引表框是码流内数据包的头的索引。它的类型是'phix' (0x7068 6978)，它的内容如下：

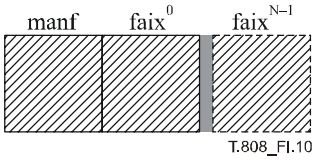

图 I.10－数据包头索引表框内容的组织方式

片断分组索引框的数量N应该不大于码流元素的数量。

manf： 清单框。这个框概括了在数据包头索引表框内由faix^{i}指定的框。它的结构在I.3.2.3中规定。

faix^{i}： 第i个片断分组索引框对应于码流中的第i个图像分量。这个表中的第m行对应于码流中的第m个分片。该行的入口保留了相应分片所有数据包头的位置和长度信息。数据包头在它们各自的分区内看起来是连续的并按照图层的顺序增加，而且分区的次序和A.3.2.1中定义的序列号的次序是相关的。但是，数据包头的次序不一定要和码流中任何COD/POC标记段的次序相同。片断分组索引框结构在I.3.2.4.2中规定。

不存在的数据包（或者因为在相同的数组中，相关的分片分量比另一个分片分量含有更少的数据包；或者因为在数据包头应该存在的点以前，码流已经被截短了）的入口应该将它们的位置域设为0。主体是空的并且头由一个字节的数据0x80组成的数据包的入口可用长度值为0来标识。这种数据包在JPEG 2000码流中经常出现；应用程序可以避免明确地取得这种内容可被预测的数据包。如果相应的COD标记段指出EPH标记在某些分片的每个数据包头后会出现，在那个分片中，这个特别长度值0应该被解释为这个数据包由0x80字节和接下来的EPH标记组成。

I.3.3 文件索引框（超级框）

I.3.3.1 概述

文件索引框可用于查找其它的索引（特别的，对应于某个码流的码流索引）以及文件中的任意数据。

根文件索引框编制文件最高层级的索引。任何其它的文件索引框编制该文件内超级框的索引。在特定文件的特定范围（最高级或特殊的超级框）内至多有一个文件索引框。

文件索引框的类型为'fidx' (0x6669 6478)，其内容如下（图I.11）：

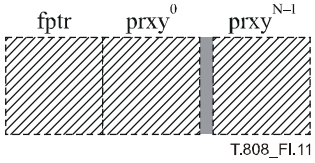

图 I.11－文件索引框的内容组织方式

fptr： 文件发现框。根文件索引框不包括这种类型的框。任何其它的文件索引框都应该包括该类型的框，它指向由文件索引框所索引的超级框。文件发现框的结构在I.3.3.2中规定。

prxy^{i}： 代理框。这个框表示由文件索引框索引的部分文件内的框。根文件索引框应该只包括文件最高级的那些框的代理。任何其它文件索引框应该只包括由文件索引框索引的超级框中处于最高级的那些框的代理。这些代理应和框有相同的次序，但是并不是所有的框都需要代理。代理框的结构在I.3.3.3中规定。

注 — 因为在某些情况下文件中框存在、缺席或排序的情况是很重要的，所以如果在任何像这样有代理的框之前，在索引的范围内没有框被索引所忽略，那么这对于应用可能是有用的。

I.3.3.2 文件发现框

文件发现框指向一个框。文件发现框的类型为'fptr' (0x6670 7472)，它的内容如下（图 I.12）：

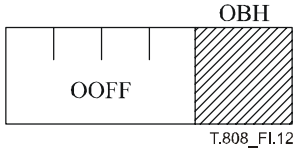

图 I.12－文件发现框的内容组织方式

OOFF：起始偏移量。这个域规定了该文件发现框所指向的框以字节为单位的偏移地址（相对于该文件的起始地址）。这个域的值是8字节、按big endian字节顺序存储的无符号整数。

OBH： 起始框头。这个域包含该文件发现框所指向的框的完整的框头。如果这个框头中LBox域的值为1，则这个域的长度为16个字节，否则为8个字节。

I.3.3.3 代理框

该框以文件索引框的形式表示一个框在文件中不同的位置，说明它的位置和长度，该框的任何索引的位置和长度，以及该框内容的前缀。

代理框的类型为'prxy' (0x7072 7879)，它的内容如下（图I.13）：

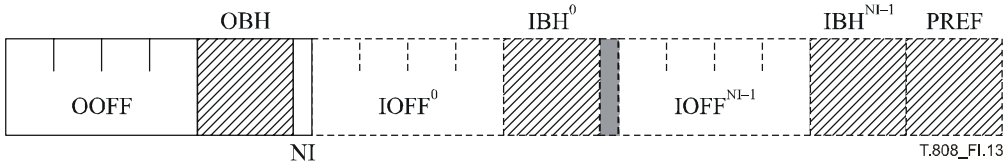

图 I.13－代理框的内容组织方式

OOFF：起始偏移量。这个域规定了由该代理框所代表的框的以字节为单位的偏移地址（相对于该文件的起始地址）。这个域的值为8字节、按big endian字节顺序存储的无符号整数。

OBH： 起始框头。这个域包含由该代理框所代表的框的完整框头。如果这个框头中LBox域的值为1，则这个域的长度为16个字节，否则为8个字节。

NI： 索引数量。这个域指出包括这个代理框在内的索引指针的数量。后面$IOFF^i$和IBH^i域的每个集合或者指向一个文件索引框或者指向一个码流索引框，这可以用来索引由该代理表示的框。所有其它的值均被保留。这个域的值为1个字节的无符号整数。

$IOFF^i$： 索引偏移量。这个域包含第i个索引框中以字节为单位的偏移量（相对于本文件的起始位置）。这个域的值为8字节、按big endian字节顺序存储的无符号整数。

IBH^i： 索引框头。这个域包含第i个索引框的完整的框头。如果这个框头中LBox域的值为1，则这个域的长度为16个字节，否则为8个字节。

PREF： 前缀。这个域包含由该代理框表示的框的一个任意前缀。它的长度可能从0到原始框内容的长度。

I.3.4 索引发现框

索引发现框指向一个文件的根文件索引框。只有在文件包含根文件索引框时它才会出现。索引发现框的类型为'iptr' (0x6970 7472)，它的内容如下(图I.14)：

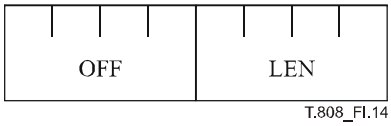

图 I.14－索引发现框的内容组织方式

OFF： 偏移量。这个域规定了根文件索引框相对于该文件起始位置的偏移量。这个域的值是8字节、按big endian字节顺序存储的无符号整数。

LEN： 长度。这个域规定了根文件索引框的大小。这个域的值是8字节、按big endian字节顺序存储的无符号整数。

I.4 码流索引和码流的关联

在JP2、JPX或JPM文件中，码流索引框应出现在文件顶层，并且第i个码流索引框对应于第i个码流，这也是在文件的顶层。码流索引框内的码流发现框也指示由码流索引框索引的那个码流。

I.5 放置的限制（资料性）

本附件中定义的框的位置几乎未加限制。根据需要，它们可以被放置在文件的末尾；当一个没有索引的文件在后来被编入索引时，这样似乎方便一些。但是，将索引发现框放在靠近文件开始的位置可能会好些，最好是直接在文章开始处一组连续的框组成的组后面（就像JP2文件的文件类型框后面或JPX文件的读者要求框后面），这个位置读者可以很容易发现。为了减少文件框的移动，如框的增加及在文件类型框的兼容性列表中选择性的添加一个'jpip'码，可在一个即将编入索引的文件中用一个空框(在ITU-T T.801建议书| ISO/IEC 15444-2的附件 M.11.20 中定义)作为占位符。

附 件 J

对本建议书|国际标准进行扩展的注册

（本附件是本建议书|国际标准的组成部分）

J.1 注册的简介

注册是指在本建议书|国际标准出版以后对它添加的扩展的过程。在本建议书|国际标准中的很多能力都可通过注册的方式进行扩展。本节确定了可能通过注册扩展的条款，注册这些能力的流程，以及注册管理处出版这些扩展的流程。只有本节中指定的这些条款才能通过注册进行扩展。

J.2 注册的要素

注册程序由下面要素组成：

— **注册管理处**：该组织实体负责所有有关注册事务的审查，保存，分配以及作为所有和注册想的实体之间的联系点。这个注册管理处还有待确定。

— **提交者**：提交者是请求某个条款被注册的组织或个人。

— **审查委员会**：审查委员会是批准某个被提议的条款注册通过的组织实体。它由审查委员会主席指定的ad hoc委员会组成。审查委员会应是ISO/IEC JTC 1/SC 29/WG 1 JPIP隶属组织。

— **审查委员会主席**：审查委员会主席的职责是确保每个候选的条款都被考虑到。他通过注册管理处和提交者进行沟通。审查委员会主席应是ISO/IEC JTC 1/SC 29/WG 1 JPIP隶属组织的主席。

— **测试**：审查委员会应该以此为基础决定是否应该注册该提议/条款。

— **提议/条款**：这是指要注册的提议。每个提议应包括要扩展的条款的名字、对该扩展提出的标签/身份以及该扩展的合理性/目的。

J.3 注册的评估标准

注册委员会应基于以下准则对所有的提议进行评估：

— 它能够满足该标准或其它扩展不能满足的需求吗？

— 这个扩展被充分说明了吗？

— 这个扩展满足的是通用的需求（例如普通的视频流应用）还是某个厂商特定的需求（例如特定厂商的视频流技术实现）？

J.4 通过注册可扩展的条款

J.4.1 占位符框内的扩展框

在占位符框（A.3.6.3）内ExtendedBoxList域中使用的新的框类型是可注册的。注册新框类型的提议应该包括这个框的完整定义（框类型和内容），说明什么时候服务器可以在占位符框中写这个框，并说明当客户端遇到包含该框的占位符框时它能做什么。

J.4.2 码流上下文

使用码流上下文域（C.4.7）请求特定码流的新的context-range值是可注册的。注册新的context-range的提议应该包括这个值的完整定义，说明服务器如何将该值映射到逻辑对象中可用的码流上，并说明服务器对码流上下文响应头如何作出响应。

J.4.3 通道传输

新的通道传输（附件H）是可注册的。注册新通道传输的提议应该包括该传输的完整定义，该定义应包括该传输的标识符。

J.4.4 偏好

新的客户端偏好是可注册的。这包括新的偏好组（如C.10.2.1中定义的新的`related-pref-set`值），或现有和已注册的偏好组的新的选项。注册新的偏好选项或偏好组的提议应包括语法、新选项的意义的完整定义，并说明服务器采样这种偏好时如何响应。

J.5 注册流程

下面是注册流程：

a） 提交者创建一个候选的注册条款。

b） 该候选的条款被提交到注册管理处。

c） 注册管理处将该候选条款递交给审查委员会主席。

d） 审查委员会主席将该候选条款分发给审查委员会并安排会议、电话等适当的方式商讨这个条款。

e） 审查委员会评估所有的提议。如果提议的文本不满足要求，将被送回提交者进行澄清。趋向于支持较通用的解决方案，且那些非常针对特定厂商的提议可能被返回给提交者，使其最大程度上变成更通用更适用于工业界。

f） 如果提议通过，主席将这个批准传达给注册管理处，注册管理处通报ISO和提交者，并使注册的或出版该条款。

g） 如果提议被拒绝，主席准备一个说明该条款被拒的原因的回复文档，并将该文档传达给注册管理处，注册管理处将此通报给提交者。

J.6 注册流程的时间进度

注册委员会将在提交日期7个月内回复所有的注册请求。在这段时间内，注册委员会将在ISO/IEC JTC 1/SC 29/WG 1的正式会议上会面，评估该提议，作出决定以及起草回复文件。

附 件 K

应用举例

（本附件不是本建议书|国际标准的组成部分）

K.1 引言

本附件介绍了一些关于JPIP实现方面的资料性例子。

K.2 在其它文件格式中使用JPIP码流

JPIP也可被用于访问存储在非JPEG 2000系列的文件格式中的JPEG 2000码流。例如，DICOM和PDF文件都能包含JPEG 2000码流。在客户端－服务器环境中，本建议书|国际标准中未规定的某些程序可被用来定位JPEG 2000码流。一旦定位了码流，即可在对象上使用JPIP请求和响应。子对象请求域就是用于这种情况的。另外，服务器还能提供通过不同的URL存取这些码流。

K.3 分片部分的实现技术

K.3.1 服务器为视窗请求确定相关的分片部分

对于通过分片部分进行的通信，将一个视窗映射到一组分片是简单的。想要的图像区域被转换为“参考网格单元”。用SIZ标记段的XTsiz和YTsiz部分确定哪些分片和视窗相交。

注 － 尽管所有的分片在参考网格中具有相同的维数，但在二次抽样的参考格子中，在子带分解后，并不一定所有的分片具有相同的维数。一个和视窗相交的分片，甚至一个完全包含在视窗中的分片，可能在最低分辨率级别下对视窗毫无贡献；然而，实现不需要利用这点在响应中将整个分片省略掉。

用分辨率级别和质量确定所需的分片部分。分片部分索引表框，若有，可用于获得分片部分在码流中的位置信息和（如果包括辅助域）分片部分内的分辨率级别的完成信息。SOT标记段也给出了分片和分片部分的索引及每个分片部分所含的字节数。码流中对应于需发送的分片部分的适当的字节被传送给客户端。在视窗改变并且对应的相关分片也改变的情况下，只需传送前面没有被传送的相关的分片部分以更新显示的图像。

K.3.2 从返回的JPT-stream消息中解码一个图像

JPIP规定了用于在客户端和服务器之间传送压缩的图像数据和元数据的机制。没有规定客户端如何显示返回的数据的机制，而且确实不同的应用之间有很大的差别。本节提供从返回的数据中获得分量样本的信息。

已经收到了所有主头数据（由对头数据块0的响应消息中已完成的头数据块指示）的客户端应用，可将那些数据块和分片数据块中完整的分片部分连接起来组成合法的JPEG 2000码流。这个码流可提供给一个兼容的JPEG 2000解码器并显示结果。当然，出于效率考虑，客户端可能希望随同码流一起向智能解码器提供视窗参数，这样只有当前视窗需要的部分才会被显示。

K.3.3 分片部分的辅助信令

表K.1和K.2显示了辅助域在扩展分片数据块消息和分片部分索引表框中的使用。

注 — 在这个例子中，r的定义和本建议书|国际标准其它地方使用的r不同，但和ITU-T T.800建议书| ISO/IEC 15444-1:2004 的附件B中的定义一致。

表K.1显示了一种简单的情况，在这种情况下，一个渐进分辨率分片的所有分片分量具有相同数量的分解层并且消息边界（在数据块的情况）或者分片部分的边界（在索引框的情况）仅出现在每个相继的分辨率级别之间。

表 K.1－辅助域在一个简单情况中应用的例子

数据块中的消息序列号或分片中的分片部分的编号	分辨率级别 r	$n = N_L - r$	辅助值
0	0	2	2
1	1	1	1
2	2	0	0

表K.2显示了一种更复杂的情况，在这种情况下，分解层的数目随分片分量而改变。每个新的辅助值第一次出现时，在表的最后一列给出了（相应的）注释。这种情况对应于一个分片，这个分片来自一个按RC…渐进顺序的三基色图像，例如一个单层对应的LRCP渐进顺序，或者分片中单个分区对应的RPCL渐进顺序。消息边界（在数据块的情况）或分片部分的边界（在索引框的情况）出现在每个分辨率级别的每个分量之间及分辨率级别之间。分量0和1有两个分解层次（$N_L = 2$），分量2有一个分解层次（$N_L = 1$）。

表 K.2－辅助域在一个更复杂情况中应用的例子

数据块中的消息序列号，或分片中的分片部分的编号	分量索引 c	分辨率级别 r	$n = N_L - r$	辅助值	注释
0	0	0	2	3	没有层已完成
1	1	0	2	2	$n = 2$ 现在已完成
2	2	0	1	2	
3	0	1	1	2	
4	1	1	1	1	$n = 1$ 现在也已完成
5	2	1	0	1	
6	0	2	0	1	
7	1	2	0	0	所有层现在都已完成

K.4 基于分区的实现技术

K.4.1 服务器为视窗请求确定相关的分区

当通信涉及JPP-stream媒体类型时，服务器将客户端请求的图像区域转换为一组和请求相应的分区。对于每个相关的码流，这个过程的第一部分包括将帧尺寸，区域尺寸和区域偏移量请求域提供的参数fx、fy、sx、sy、ox和oy转换为码流帧尺寸，区域尺寸和偏移量参数fx'、fy'、sx'、sy'、ox' 和oy'。这种转换对基于分区和基于分片的服务以相同的方式进行，且基于公式C-1和C-2，可能会根据公式C-3和C-4做一些修正。本节描述了在特定的码流中服务器应该如何确定和由参数fx'、fy'、sx'、sy'、ox'和oy'定义的区域相关的分区。

基于客户端的请求，令公式C-1中的r为非负整数，服务器使用该公式来查找fx'和fy'。正如上面提到的，r最容易被解释为舍去的最高分辨率DWT的级数，尽管r允许超过实际对于任意给定的分片分量都是可得到的DWT级数。将由sx'、sy'、ox'和oy'描述的区域首先映射到码流的高分辨率网格中是方便的。这个映射产生了一个区域，它的左上角由 $(E_1^{\text{reg}}, E_2^{\text{reg}})$ 给出，右下角由 $(F_1^{\text{reg}} - 1, F_2^{\text{reg}} - 1)$ 给出。其中：

$$E_1^{\text{reg}} = \text{XOsiz} + 2^r \cdot \text{ox}', \quad E_2^{\text{reg}} = \text{YOsiz} + 2^r \cdot \text{oy}', \quad F_1^{\text{reg}} = E_1^{\text{reg}} + 2^r \cdot \text{sx}', \quad \text{and} \quad F_2^{\text{reg}} = E_2^{\text{reg}} + 2^r \cdot \text{sy}'$$

服务器仅需考虑和码流的高分辨率网格中的区域相交的那些分片。对于每个分片，服务器仅需考虑客户端请求的那些图像分量，通常以分量和码流上下文请求域结合的方式来描述。对于每个由t 和 c表示的被考虑的分片分量，令$D_{t,c}$为用于压缩该分片分量的DWT级数。如果Dt,c≥r，服务器应丢弃所有属于分片分

量的*r*最高分辨率级别的分区弃；否则，服务器应丢弃所有属于分片分量的$D_{t,c}$最高分辨率级别的分区，仅留下表示分片分量的最低LL子带的那些分区。

对于丢弃上面提到的分片、分量和分辨率级别后剩下的每个区域，服务器应该识别属于该区域的码块对由码流的高分辨率格子中的$E_1^{\text{reg}}, E_2^{\text{reg}}$和$F_1^{\text{reg}}, F_2^{\text{reg}}$定义的区域的重建是否有贡献。一个码块对该区域有贡献，如果它的任意样本影响任意全分辨率图像分量样本的重建，这个全分辨率图像分量样本的坐标*(x,y)*满足：

$$E_1^{\text{reg}} \le \text{XRsiz}^c \cdot x < F_1^{\text{reg}} \quad \text{and} \quad E_2^{\text{reg}} \le \text{YRsiz}^c \cdot y < F_2^{\text{reg}}$$

其中XRsiz^c和YRsiz^c表示相应分量的水平和垂直二次采样因子，其中相应分量指的是码流的SIZ标记段的*c*。

重要的是要记住全分辨率图像分量的重建包括小波综合，这是一个固有的易扩张的过程。因此，一个由任意给定的分区贡献的区域通常与另一个由它的相邻分区贡献的区域重叠。在确定哪些区域和客户端请求时，服务器应准备好考虑这种小波变换的扩张效果。

"JPEG2000: image compression fundamentals, standards and practice" [11]这本书的10.6.4节描述了一种计算任意给定的子带的样本的方法，该子带对码流的高分辨率网格上给定的区域有贡献。从子带的区域中，推断出有贡献的码块和分区是一件简单的事。

K.4.2 从返回的JPP-stream消息中解码一个图像

JPIP规定了用于在客户端和服务器之间传送压缩的图像数据和元数据的机制。没有规定客户端如何显示返回的数据的机制，而且确实不同的应用之间有很大的差别。

K.5 JPIP 协议脚本

K.5.1 引言

在下面例子的脚本中，行首符号"<<"后的文本是从客户端发送到服务器的，行首符号">>"后的文本是从服务器发送到客户端的，符号"--"后的文本是注释，实际中不传送。这些注释可能指出没有显示一些传输了的数据。

K.5.2 使用HTTP

下面的脚本显示了5个从客户端发给服务器的请求及服务器的响应。

第一个请求要求一个名为phoenix.jp2的JP2文件，请求这个文件中的第一个码流，最大长度被放到响应中，请求了对象ID，要求数据以JPP-stream的形式返回，并要求建立一个HTTP上的会话。由于没有窗口，因此没有请求图像数据。

服务器的回复中提供一个图像的对象ID和新建立的通道的ID。以"JPIP-cnew"开头的头行表示一个可以访问图像文件的新的路径。路径"jpip"的值可能是用于处理所有JPIP交互命令的服务器的一个CGI程序的路径。这个文件中的一些数据在主体中返回；这些是文件格式框，也许是第一个码流的主头。

客户端的第二个请求使用一个名为"jpip.cgi"的新路径和一个通道ID标识想要的图像（不需要图像名或对象ID）。该请求也规定了一个感兴趣的特定窗口。

第二个请求的响应指出视窗已被改变且正返回请求的视窗中心的一个缩小的窗口。服务器开始返回该窗口的数据。

在尚未接收完对第二个请求的所有响应之前，客户端发送了第三个请求。客户端已经按照服务器规定的尺寸调整了它的视窗尺寸。

服务器暂时继续响应第二个请求，然后开始响应第三个请求。在这个响应的过程中，客户端发送稍有不同的区域的第四个请求。服务器暂时继续对响应第三个请求然后开始响应第四个请求。

客户端等待，直到第四个响应完成，然后发送一个请求以结束会话和HTTP连接。本例中没有显示连接关闭时的响应的数据。

```
<< GET /phoenix.jp2?stream=0&len=2000&tid=0&type=jpp-stream&cnew=http HTTP/1.1
<< Host: dst-m
<<
    >> HTTP/1.1 200 OK
    >> JPIP-tid: 281B6E135135BBC0BC588452AC9B73C5
    >> JPIP-cnew: cid=JPH_033C38BE48115AC9,path=jpip.cgi,transport=http
    >> Cache-Control: no-cache
    >> Transfer-Encoding: chunked
    >> Content-Type: image/jpp-stream
    >>
    >> 102
       -- 258个字节的二进制数据
    >> 0
    >>
<< GET /jpip.cgi?fsiz=834,834&roff=0,0&rsiz=834,790&comps=0-
2&stream=0&len=2000&cid=JPH_033C38BE48115AC9 HTTP/1.1
<< Host: dst-m
<< Cache-Control: no-cache
<<
    >> HTTP/1.1 200 OK, with modifications
    >> JPIP-roff: 120,114
    >> JPIP-rsiz: 593,561
    >> Cache-Control: no-cache
    >> Transfer-Encoding: chunked
    >> Content-Type: image/jpp-stream
    >>
    >> 393
       -- 915个字节的二进制数据
<< GET /jpip.cgi?fsiz=834,834&roff=120,114&rsiz=593,561&comps=0-
2&stream=0&len=2000&cid=JPH_033C38BE48115AC9 HTTP/1.1
<< Host: dst-m
<< Cache-Control: no-cache
<<
    >> 3f9
       -- 1017 bytes of binary data
    >> 0
    >>
    >> HTTP/1.1 200 OK
    >> Cache-Control: no-cache
    >> Transfer-Encoding: chunked
    >> Content-Type: image/jpp-stream
    >>
    >> 359
       -- 857个字节的二进制数据
<< GET /jpip.cgi?fsiz=834,834&roff=309,297&rsiz=121,86&comps=0-
2&stream=0&len=3906&cid=JPH_033C38BE48115AC9 HTTP/1.1
<< Host: dst-m
<< Cache-Control: no-cache
<<
    >> 234
       -- 564个字节的二进制数据
    >> 3d0
       -- 976个字节的二进制数据
    >> 24f
       -- 591个字节的二进制数据
    >> 0
    >>
    >> HTTP/1.1 200 OK
    >> Cache-Control: no-cache
    >> Transfer-Encoding: chunked
```

```
        >> Content-Type: image/jpp-stream
        >>
        >> 3b2
           -- 946个字节的二进制数据
        >> 400
           -- 1024个字节的二进制数据
        >> 263
           -- 611个字节的二进制数据
        >> 356
           -- 854个字节的二进制数据
        >> 209
           -- 521个字节的二进制数据
        >> 0

   << GET /jpip.cgi?cclose=JPH_033C38BE48115AC9&len=0 HTTP/1.1
   << Host: dst-m
   << Connection: close
   << Cache-Control: no-cache
   <<
```

下面是使用基于会话的HTTP GET实现模型请求的例子。

```
   << GET /jpip.cgi?fsiz=1024,768&cid=JPH_5&model=Hm,H*,M*,P* HTTP/1.1
   << Host: jpip.jpeg.org
   << Cache-Control: no-cache

       >> HTTP/1.1 200 OK
       >> Cache-control: no-cache
       >> Transfer-Encoding: chunked
       >> 3
       -- 3个字节的二进制数据
       >> 0
```

下面使用无状态的HTTP GET实现模型请求的例子。

```
   << GET /images/kids.jp2?fsiz=1024,768&model=M0,Hm,H0:20,P0 HTTP/1.1
   << Host: jpip.jpeg.org
   << Cache-Control: no-cache

       >> HTTP/1.1 200 OK
       >> Cache-Control: no-cache
       >> Transfer-Encoding: chunked
       >> Content-Type: image/jpp-stream
       >> 400
       -- 1024个字节的二进制数据
       >> 3f8
       -- 1016个字节的二进制数据
       >> 0
```

K.5.3 用HTTP和TCP返回

```
   << GET /phoenix.jp2?stream=0&len=2000&tid=0&type=jpp-stream&cnew=http-
   tcp,http HTTP/1.1
   << Host: dst-m
   <<
       >> HTTP/1.1 200 OK
       >> JPIP-tid: 281B6E135135BBC0BC588452AC9B73C5
       >> JPIP-cnew: cid=JPHT033C38BE481154F9,path=jpip,transport=http-
          tcp,auxport=80
       >> Cache-Control: no-cache
       >>
           << JPHT033C38BE481154F9 – [Note: This is the TCP channel connection
message]
           <<
```

```
<< GET /jpip.cgi?fsiz=834,834&roff=0,0&rsiz=834,790&comps=0-
2&stream=0&cid=JPHT033C38BE481154F9 HTTP/1.1
<< Host: dst-m
<< Cache-Control: no-cache
<<
    >> HTTP/1.1 200 OK, with modifications
    >> JPIP-roff: 120,114
    >> JPIP-rsiz: 593,561
    >> Cache-Control: no-cache
    >>

<< GET /jpip.cgi?fsiz=834,834&roff=229,254&rsiz=155,113&comps=0-
2&stream=0&cid=JPHT033C38BE481154F9 HTTP/1.1
<< Host: dst-m
<< Cache-Control: no-cache
<<
    >> HTTP/1.1 200 OK
    >> Cache-Control: no-cache
    >>

<< GET /jpip.cgi?fsiz=1667,1667&roff=457,507&rsiz=310,226&comps=0-
2&stream=0&cid=JPHT033C38BE481154F9 HTTP/1.1
<< Host: dst-m
<< Cache-Control: no-cache
<<
    >> HTTP/1.1 200 OK
    >> Cache-Control: no-cache
    >>

<< GET /jpip.cgi?fsiz=3334,3334&roff=914,1014&rsiz=620,452&comps=0-
2&stream=0&cid=JPHT033C38BE481154F9 HTTP/1.1
<< Host: dst-m
<< Cache-Control: no-cache
<<
    >> HTTP/1.1 200 OK
    >> Cache-Control: no-cache
    >>

<< GET /jpip.cgi?cclose=JPHT033C38BE481154F9 HTTP/1.1
<< Host: dst-m
<< Cache-Control: no-cache
<<
```

K.6 和HTML一起使用JPIP

JPIP系统能以多种方式应用于HTML和XHTML页面。如果JPIP服务器包括将一个图像的多个部分转换为JPEG或其它完整的图像媒体类型的能力，那么HTML可被用来存取一个JPEG 2000图像的多个部分而无需对目前的浏览器作任何改变。

考虑包含下面的HTML片段的一个网页：

```
<img
src="http://jpip.jpeg.org/name.jp2?fsiz=128,128&rsiz=128,128&type=image/j
peg" width="128" height="128">
```

任何希望显示和图像一起显示该网页的浏览器将发送一个获取这个图像的请求。这个请求以下面的文字开始：

```
GET /name.jp2?fsiz=128,128&rsiz=128,128&type=image/jpeg
Host: jpip.jpeg.org
```

并且包括许多其它的HTTP头行，通常标识该浏览器和该浏览器接收的东西的类型。这个HTTP请求是一个合法的JPIP请求，收到这个请求的JPIP服务器应该返回一个错误消息或者确定JP2文件的相关部分以获取并将它转换为JPEG文件。返回的消息应该形似于：

```
HTTP/1.1 200 OK
Content-type: image/jpeg
Content-length: 20387
CRLF
JPEG-Compressed-Image-Data
```

这是一个合法的JPIP响应；也是一个合法的HTTP响应，所有的图像浏览器都知道如何显示。注意优先选择但并不要求服务器实现分块编码传送，因此该请求可以被中断。前面的例子不是分块编码传送的例子。

还可能当仅有JPEG可用时，用JPEG写网页，当JPEG 2000 可用时，使用JPEG2000写网页；当在客户端浏览器中可用时，使用JPT-stream或JPP-stream。考虑HTML片段：

```
<img src="http://jpip.jpeg.org/name.jp2?rsiz=128,128" width="128"
     height="128">
```

在这种情况下，没有请求明确的类型。因此一个使用HTTP的JPIP服务器应该检查由客户端发送的HTTP请求的"Accept:"行。根据image/jp2 或 image/jpt-stream 或 image/jpp-stream 或 image/jpeg是否出现，服务器可以决定返回兼容的格式。

附 件 L

JPIP的ABNF集合

（本附件不是本建议书|国际标准的组成部分）

L.1 JPIP请求的ABNF

```
;================================
; C.1.1 请求结构
;================================

jpip-request-field = target-field
                   / channel-field
                   / view-window-field
                   / metadata-field
                   / data-limit-field
                   / server-control-field
                   / cache-management-field
                   / upload-field
                   / client-cap-pref-field

target-field       = target                          ; C.2.2
                   / subtarget                       ; C.2.3
                   / tid                             ; C.2.4

channel-field      = cid                             ; C.3.2
                   / cnew                            ; C.3.3
                   / cclose                          ; C.3.4
                   / qid                             ; C.3.5

view-window-field  = fsiz                            ; C.4.2
                   / roff                            ; C.4.3
                   / rsiz                            ; C.4.4
                   / comps                           ; C.4.5
                   / stream                          ; C.4.6
                   / context                         ; C.4.7
                   / srate                           ; C.4.8
                   / roi                             ; C.4.9
                   / layers                          ; C.4.10

metadata-field     = metareq                         ; C.5.2

data-limit-field   = len                             ; C.6.1
                   / quality                         ; C.6.2

server-control-field = align                         ; C.7.1
                   / wait                            ; C.7.2
                   / type                            ; C.7.3
                   / drate                           ; C.7.4

cache-management-field = model                       ; C.8.1
                   / tpmodel                         ; C.8.3
                   / need                            ; C.8.4
                   / tpneed                          ; C.8.5
                   / mset                            ; C.8.6

upload-field       = upload                          ; C.9.1
```

```
client-cap-pref-field = cap                              ; C.10.1
                      / pref                             ; C.10.2
                      / csf                              ; C.10.3;
 ================================
; C.2.2 对象(target)
;================================
target = "target" "=" PATH
;================================
; C.2.3 子对象(subtarget)
;================================
subtarget = "subtarget" "=" byte-range
byte-range = UINT-RANGE
;================================
; C.2.4 对象ID (tid)
;================================
tid = "tid" "=" target-id
target-id = TOKEN
;================================
; C.3.1 通道ID (cid)
;================================
cid = "cid" "=" channel-id
channel-id = TOKEN
;================================
; C.3.2 新建通道(cnew)
;================================
cnew = "cnew" "=" 1#transport-name
transport-name = TOKEN
;================================
; C.3.3 关闭通道 (cclose)
;================================
cclose = "cclose" "=" ("*" / 1#channel-id)
;================================
; C.3.4 请求ID (qid)
;================================
qid = "qid" "=" UINT
;================================
; C.4.2 帧尺寸 (fsiz)
;================================
fsiz = "fsiz" "=" fx "," fy ["," round-direction]
fx = UINT
fy = UINT
round-direction = "round-up" / "round-down" / "closest"
;================================
; C.4.3 偏移量 (roff)
;================================
roff = "roff" "=" ox "," oy
ox = UINT
oy = UINT
```

```
;================================
; C.4.4 区域尺寸 (rsiz)
;================================
rsiz = "rsiz" "=" sx "," sy
sx = UINT
sy = UINT
;================================
; C.4.5 分量 (comps)
;================================
comps = "comps" "=" 1#UINT-RANGE
;================================
; C.4.6 码流 (stream)
;================================
stream = "stream" "=" 1#sampled-range
sampled-range = UINT-RANGE [":" sampling-factor]
sampling-factor = UINT
;================================
; C.4.7 码流上下文 (context)
;================================
context = "context" "=" 1#context-range
context-range = jpxl-context-range / mj2t-context / reserved-context
jpxl-context-range = "jpxl" "<" jpx-layers ">" [ "[" jpxl-geometry "]" ]
jpx-layers = sampled-range
jpxl-geometry = "s" jpx-iset "i" jpx-inum
jpx-iset = UINT
jpx-inum = UINT
mj2t-context = "mj2t" "<" mj2-track ">" [ "[" mj2t-geometry "]" ]
mj2-track = NONZERO ["+" "now" ]
mj2t-geometry = "track" / "movie"
reserved-context = 1*( TOKEN / "<" / ">" / "[" / "]" / "-" / ":" / "+" )
;================================
; C.4.8 抽样率 (srate)
;================================
srate = "srate" "=" streams-per-second
streams-per-second = UFLOAT
;================================
; C.4.9 ROI (roi)
;================================
roi = "roi" "=" region-name
region-name = 1*(DIGIT / ALPHA / "_")
            / "dynamic"
;================================
; C.4.10 图层(layers)
;================================
layers = "layers" "=" UINT
```

```
;================================
; C.5.2 元数据请求 (metareq)
;================================
metareq = "metareq" "=" 1#("[" 1$(req-box-prop) "]" [root-bin] [max-depth])
                        [metadata-only]
req-box-prop = box-type [limit] [metareq-qualifier] [priority]
limit = ":" (UINT / "r")
metareq-qualifier = "/" 1*("w" / "s" / "g" / "a")
priority = "!"
root-bin = "R" UINT
max-depth = "D" UINT
metadata-only = "!!"
;================================
; C.6.1 最大响应长度 (len)请求域
;================================
len = "len" "=" UINT
;================================
; C.6.2 质量 (quality)请求域
;================================
quality = "quality" "=" (1*2DIGIT / "100")        ; 0 to 100
;================================
; C.7.1 对齐 (align)请求域
;================================
align = "align" "=" ("yes" / "no")
;================================
; C.7.2 Wait (wait)请求域
;================================
wait = "wait" "=" ("yes" / "no")
;================================
; C.7.3 图像返回类型 (type)
;================================
type = "type" "=" 1#image-return-type
image-return-type = media-type / reserved-image-return-type
media-type = TOKEN "/" TOKEN *( ";" parameter )
reserved-image-return-type = TOKEN *( ";" parameter )
parameter = attribute "=" value
attribute = TOKEN
value = TOKEN
;================================
; C.7.4 传送速率 (drate)
;================================
drate = "drate" "=" rate-factor
rate-factor = UFLOAT
```

```
;================================
; C.8.1.1 Model (model)
;================================
model = "model" "=" 1#model-item
model-item = [codestream-qualifier ","] model-element
model-element = ["-"] bin-descriptor
bin-descriptor = explicit-bin-descriptor    ; C.8.1.2
               / implicit-bin-descriptor    ; C.8.1.3
codestream-qualifier = "[" 1$(codestream-range) "]"
codestream-range = first-codestream-id ["-" [last-codestream-id]]
first-codestream-id = UINT
last-codestream-id = UINT
;================================
; C.8.1.2 显式
;================================
explicit-bin-descriptor = explicit-bin
                       [":" (number-of-bytes / number-of-layers )]
explicit-bin = codestream-main-header-bin
             / meta-bin
             / tile-bin
             / tile-header-bin
             / precinct-bin
number-of-bytes = UINT
number-of-layers = %x4c UINT                    ; "L"
codestream-main-header-bin = %x48 %x6d    ; "Hm"
meta-bin = %x4d bin-uid                              ; "M"
tile-bin = %x54 bin-uid                              ; "T"
tile-header-bin = %x48 bin-uid                  ; "H"
precinct-bin = %x50 bin-uid                     ; "P"
bin-uid = UINT / "*"
;================================
; C.8.1.3 隐式
;================================
implicit-bin-descriptor = 1*implicit-bin [":" number-of-layers]
implicit-bin  = implicit-bin-prefix (data-uid / index-range-spec)
implicit-bin-prefix = %x74    ; t -- tile
                    / %x63    ; c -- component
                    / %x72    ; r -- resolution level
                    / %x70    ; p -- position
index-range-spec = first-index-pos "-" last-index-pos
first-index-pos = UINT
last-index-pos = UINT
data-uid = UINT / "*"
```

```
;================================
; C.8.3 Tile-part Model involving JPT-streams (tpmodel)
;================================
tpmodel =  "tpmodel" "=" 1#tpmodel-item
tpmodel-item = [codestream-qualifier "," ] tpmodel-element
tpmodel-element = ["-"] tp-descriptor
tp-descriptor = tp-range / tp-number
tp-range = tp-number "-" tp-number
tp-number = tile-number "." part-number
tile-number = UINT
part-number = UINT
;================================
; C.8.4 无状态请求的需要(need)
;================================
need = "need" "=" 1#need-item
need-item = [codestream-qualifier "," ] bin-descriptor
;================================
; C.8.5 无状态请求的分片部分需要 (tpneed)
;================================
tpneed = "tpneed" "=" 1#tpneed-item
tpneed-item = [codestream-qualifier "," ] tp-descriptor
;================================
; C.8.6 会话中请求的模型组 (mset)
;================================
mset =  "mset" "=" 1#sampled-range
;================================
; C.9.1 上传 (upload)
;================================
upload = "upload" "=" upload-type
upload-type = image-return-type                    ; C.7.3
;================================
; C.10.1 客户端能力(cap)
;================================
cap = "cap" "=" 1#capability-group
capability-group = processing-capability
                 / depth-capability
                 / config-capability
processing-capability = compatibility-capability
                      / vendor-capability
compatibility-capability = "cc." compatibility-code
vendor-capability = "vc." vendor-code [":" vendor-value]
vendor-code = 1*(LOWER / DIGIT / "." / "-")
vendor-value = TOKEN
depth-capability = "depth:" UINT
config-capability = "config:" UINT
```

```
;================================
; C.10.2.1 概述
;================================
pref = "pref" "=" 1#(related-pref-set ["/r"])
related-pref-set = view-window-pref            ; C.10.2.2
                 / colour-meth-pref            ; C.10.2.3
                 / max-bandwidth               ; C.10.2.4
                 / bandwidth-slice             ; C.10.2.5
                 / placeholder-pref            ; C.10.2.6
                 / codestream-seq-pref         ; C.10.2.7
                 / other
other = TOKEN
;================================
; C.10.2.2 视窗处理的偏好
;================================
view-window-pref = "fullwindow" / "progressive"
;================================
; C.10.2.3 彩色空间方法的偏好
;================================
color-meth-pref = 1$(color-meth [":" meth-limit])
color-meth = "color-enum" / "color-ricc" / "color-icc" / "color-vend"
meth-limit = UINT
;================================
; C.10.2.4 最大带宽
;================================
max-bandwidth = "mbw:" mbw
mbw = UINT ["K" / "M" / "G" / "T"]
;================================
; C.10.2.5 带宽切片
;================================
bandwidth-slice = "slice:" slice
slice = NONZERO
;================================
; C.10.2.6 占位符偏好
;================================
placeholder-pref = "meta:" placeholder-branch
placeholder-branch = "incr" / "equiv" / "orig"
;================================
; C.10.2.7 码流排序
;================================
codestream-seq-pref = "codeseq:" codestream-seq-option
codestream-seq-option = "sequential" / "reverse-sequential"
                      / "interleaved"
;================================
; C.10.3 对比灵敏度 (csf)
;================================
csf = "csf" "=" 1#csf-sample-line
csf-sample-line = csf-density [";" csf-angle] ";" 1$sensitivity
csf-density = "density" ":" UFLOAT
```

```
csf-angle = "angle" ":" UFLOAT
sensitivity = UFLOAT
```

L.2 JPIP响应的BNF

```
;================================
; D.1.1 响应结构
;================================
Status-Code = 3DIGIT
Reason-Phrase = *<TEXT, excluding CR and LF>
jpip-response-header =
                      / JPIP-tid                    ; D.2.2
                      / JPIP-cnew                   ; D.2.3
                      / JPIP-qid                    ; D.2.4
                      / JPIP-fsiz                   ; D.2.5
                      / JPIP-rsiz                   ; D.2.6
                      / JPIP-roff                   ; D.2.7
                      / JPIP-comps                  ; D.2.8
                      / JPIP-stream                 ; D.2.9
                      / JPIP-context                ; D.2.10
                      / JPIP-roi                    ; D.2.11
                      / JPIP-layers                 ; D.2.12
                      / JPIP-srate                  ; D.2.13
                      / JPIP-metareq                ; D.2.14
                      / JPIP-len                    ; D.2.15
                      / JPIP-quality                ; D.2.16
                      / JPIP-type                   ; D.2.17
                      / JPIP-mset                   ; D.2.18
                      / JPIP-cap                    ; D.2.19
                      / JPIP-pref                   ; D.2.20
;================================
; D.2.2 对象ID(JPIP-tid)
;================================
JPIP-tid = "JPIP-tid" ":" LWSP target-id
;================================
; D.2.3 新建通道 (JPIP-cnew)
;================================
JPIP-cnew = "JPIP-cnew" ":" LWSP "cid" "=" channel-id
            ["," 1#(transport-param "=" TOKEN)]
transport-param = TOKEN
;================================
; D.2.4 请求ID (JPIP-qid)
;================================
JPIP-qid = "JPIP-qid" ":" LWSP UINT
;================================
; D.2.5 帧尺寸 (JPIP-fsiz)
;================================
JPIP-fsiz = "JPIP-fsiz" ":" LWSP fx "," fy
;================================
; D.2.6 区域尺寸 (JPIP-rsiz)
;================================
JPIP-rsiz = "JPIP-rsiz" ":" LWSP sx "," sy
```

```
    ;================================
    ; D.2.7 偏移量 (JPIP-roff)
    ;================================
    JPIP-roff = "JPIP-roff" ":" LWSP ox "," oy
    ;================================
    ; D.2.8 分量 (JPIP-comps)
    ;================================
    JPIP-comps = "JPIP-comps" ":" LWSP 1#UINT-RANGE
    ;================================
    ; D.2.9 码流 (JPIP-stream)
    ;================================
    JPIP-stream = "JPIP-stream" ":" LWSP 1#(prefixed-range / sampled-range)
    prefixed-range = "<" ctxt-id ":" ctxt-elt ">" sampled-range
    ctxt-id = UINT
    ctxt-elt = UINT
    ;================================
    ; D.2.10 码流上下文 (JPIP-context)
    ;================================
    JPIP-context = "JPIP-context" ":" LWSP 1$(context-range "=" 1#sampled-
range)
    ;================================
    ; D.2.11 ROI (JPIP-roi)
    ;================================
    JPIP-roi = "JPIP-roi" ":" LWSP
               "roi" "=" region-name ";"
               "fsiz" "=" UINT "," UINT ";"
               "rsiz" "=" UINT "," UINT ";"
               "roff" "=" UINT "," UINT ";"
    region-name = 1*(DIGIT / ALPHA / "_")
    ;================================
    ; D.2.12 图层(JPIP-layers)
    ;================================
    JPIP-layers = "JPIP-layers" ":" LWSP UINT
    ;================================
    ; D.2.13 抽样率 (JPIP-srate)
    ;================================
    JPIP-srate = "JPIP-srate" ":" LWSP UFLOAT
    ;================================
    ; D.2.14 元数据请求 (JPIP-metareq)
    ;================================
    JPIP-metareq = "JPIP-metareq" ":" LWSP
                   1#( "[" 1$(req-box-prop) "]" [root-bin] [max-depth] )
                   [metadata-only]
    req-box-prop = box-type [limit] [metareq-qualifier] [priority]
    ;================================
    ; D.2.15 最大响应长度 (JPIP-length)
    ;================================
    JPIP-len = "JPIP-len" ":" LWSP UINT
    ;================================
    ; D.2.16 质量 (JPIP-quality)
    ;================================
```

```
JPIP-quality = "JPIP-quality" ":" LWSP (1*2DIGIT / "100" / "-1")
;================================
; D.2.17 图像返回类型 (JPIP-type)
;================================
JPIP-type = "JPIP-type" ":" LWSP image-return-type
;================================
; D.2.18 模型组 (JPIP-mset)
;================================
JPIP-mset = "JPIP-mset" ":" LWSP 1#sampled-range
;================================
; D.2.19 需要的能力 (JPIP-cap)
;================================
JPIP-cap = "JPIP-cap" ":" LWSP 1#capability-code
;================================
; D.2.20 偏好不可得 (JPIP-pref)
;================================
JPIP-pref = "JPIP-pref" ":" LWSP 1#related-pref-set
```

附 件 M

专利声明

（本附件不是本建议书|国际标准的组成部分）

国际标准化组织（ISO）和国际电工委员会（IEC）提请注意一个事实，即据声称，使用ISO/IEC 15444的这部分可能涉及对专利的使用。

ISO和IEC对有关这些专利权的证据、有效性和范围不表示意见。

专利权人已经向ITU、ISO和IEC保证他们愿意在合理且非歧视的条件与期限下和全世界的申请人进行许可谈判。在这方面，已向ITU、ISO和IEC注册了这些专利权人的声明。可从下面列出的公司中获取相关信息。

ITU、ISO和IEC没有对有关这个专利权的证据、有效性和范围采取措施。

提请注意，ISO/IEC 15444的这部分中某些部分可能是未本附件中提到的专利权的题目。ISO和IEC不负责对任何或者全部这种专利权的鉴别。

	公司
1	Canon Inc.
2	Ricoh Company, Limited.

附 件 N

参考资料

（本附件不是本建议书|国际标准的组成部分）

[1] TAUBMAN (D.): Remote Browsing of JPEG 2000 Images, *Proc. Int. Conf. on Image Processing*, Vol. 1, pp. 229-232, Sept. 2002.

[2] LI (J.), SUN (H.), LI (H.), ZHANG (Q.), LIN (X.): Vfile – A Virtual File Media Access Mechanism and its Application in JPEG2000 Images for Browsing over Internet, *ISO/IEC JTC 1/SC 29/WG 1 Document Register: N1473*, Nov. 1999.

[3] BOLIEK (M.), WU (G.K.), GORMISH (M.J.): JPEG 2000 for Efficient Imaging in a Client/Server Environment, *Proc. SPIE Conf. on Applications of Digital Image Processing*, Vol. 4472, pp. 212-223, Dec. 2001.

[4] DESHPANDE (S.), ZENG (W.): Scalable Streaming of JPEG2000 Images Using Hypertext Transfer Protocol, *Proc. ACM Conf. on Multimedia*, pp. 372-381, Oct. 2001.

[5] WRIGHT (A.), CLARK (R.), COLYER (G.): An Implementation of JPIP Based on HTTP, *ISO/IEC JTC 1/SC 29/WG 1 Document Register: N2426*, Feb. 2002.

[6] GORMISH (M.), BANERJEE (S.): Tile-Based Transport of JPEG 2000, N. Garcia, J.M. Martinez, L. Salgado (Eds.), VLVB03, LNCS 2849, pp. 217-224, 2003.

[7] TAUBMAN (D.), ROSENBAUM (R.): Rate-Distortion Optimized Interactive Browsing of JPEG2000 Images, *Proc. Int. Conf. on Image Processing*, Sept. 2003.

[8] TAUBMAN (D.), PRANDOLINI (R.): Architecture, Philosophy and Performance of JPIP: Internet Protocol Standard for JPEG2000, presented at *Visual Communications and Image Processing*, Lugano, Switzerland, 2003.

[9] GORMISH (M.J.): TRUEW: Transport of Reversible and Unreversible EmbeddedWavelets (A JPIP Proposal), *ISO/IEC JTC 1/SC 29/WG 1 Document Register: N2602*, July 2002.

[10] CANON: Proposal for JPIP Tier 2 protocol, *ISO/IEC JTC 1/SC 29/WG 1 Document Register: N2608*, June 2002.

[11] TAUBMAN (D.), MARCELLIN (M.): JPEG2000: image compression fundamentals, standards and practice, *Kluwer Academic Publishers*, Boston, 2001.

ITU-T 系列建议书

A系列	ITU-T工作的组织
D系列	一般资费原则
E系列	综合网络运行、电话业务、业务运行和人为因素
F系列	非话电信业务
G系列	传输系统和媒质、数字系统和网络
H系列	视听和多媒体系统
I系列	综合业务数字网
J系列	有线网和电视、声音节目及其他多媒体信号的传输
K系列	干扰的防护
L系列	线缆的构成、安装和保护及外部设备的其他组件
M系列	电信管理，包括TMN和网络维护
N系列	维护：国际声音节目和电视传输电路
O系列	测量设备技术规程
P系列	电话传输质量、电话装置、本地线路网络
Q系列	交换和信令
R系列	电报传输
S系列	电报业务终端设备
T系列	**远程信息处理业务的终端设备**
U系列	电报交换
V系列	电话网上的数据通信
X系列	数据网和开放系统通信及安全
Y系列	全球信息基础设施、互联网的协议问题和下一代网络
Z系列	用于电信系统的语言和一般软件问题

瑞士印刷
2005年，日内瓦