ISO/IEC JTC 1/SC 29/WG 1 N2656

Date: 2002 July 15

ISO/IEC JTC 1/SC 29/WG 1
(ITU-T SG16)

Coding of Still Pictures

JBIG
Joint Bi-level Image Experts Group

JPEG
Joint Photographic Experts Group

TITLE: JPIP use cases from Genoa meeting

SOURCE: JPIP editors

PROJECT: JPEG-2000

STATUS: For information

REQUESTED ACTION: For review and action as appropriate

DISTRIBUTION: WG1 Mailing List

Contact:
ISO/IEC JTC 1/SC 29/WG 1
Convener - Dr. Daniel T. Lee
Yahoo!, Rm 2802, Sunning Plaza, 10 Hysan Avenue, Causeway Bay, Hong Kong
Tel: +1 408 992 7051/+852 2882 3898, Fax: +1 253 830 0372, E-mail: dlee@yahoo-inc.com
Webmaster - Richard Clark
Elysium Ltd, Milton House, Whitehill Road, Crowborough, East Sussex TN^ 1LB, United Kingdom
Tel: +44 1892 667411, Fax: +44 1892 667433, E-mail: richard@elysium.ltd.uk

Date: 2002 July

TITLE: **Use cases from Genoa meeting**

SOURCE: Richard Clark, UK - Editor

Introduction

The following use cases are those that were initially presented at the London February 2002 ad hoc meeting of the JPIP group, which were then subsequently refined and discussed, and some new cases added at the Genoa WG1 meeting in March 2002.

These were collected and edited by Eric Majani of Canon Inc, whose efforts are acknowledged and appreciated.

I apologise if numbering is incorrect, and for the late delivery, as the action of cutting and pasting these outlines caused significant problems in my installation of Word 2000.

Use Case #1

Source: Adobe

1. **Use-case scenario**

1.1 Title: **Embedded Web application**

2. **Context:** It is critical that any protocol developed be useful for other than standalone image applications. For example, the protocol needs to be applicable to plug-ins for various web browsers

2.1 **Market application** (include info about why is JPIP needed?) A major client-server environment for image data is web access. The protocol must be suitable for enhancing clients so that they achieve a JPEG 2000 enhanced experience when accessing images on dumb server/web sites.

2.2 **Client platform characteristics** (PDA, desktop, mobile phone, …) All web access points should be supported: PDA, desktop, mobile phone, etc.

2.3 **Server platform characteristics** (PDA, desktop, large server, number of clients to serve, …) While dumb (existing) servers are critical, all future servers are useful as well

2.4 **Network** (bandwidth, delay, errors, architecture, QoS …) All

2.5 **JPEG2000 Image Characteristics**: all conformance profiles

2.6 **Other characteristics**: none

3. **Description of use-case**

3.1 **Introduction:** all normal JPEG 2000 scalablity uses should be supported by the below feature

3.2 **Description in steps** n/a

4. **Features required**

4.1 **Title** (for each feature) Embedded client image software

4.2 **Definition**: A feature of the protocol is that it needs to be implementable in such a manner that it may be added to client software in a plug-in manner.

4.3 Detail of feature required

4.4 **Client requirement:** plug-in type implementation possible

4.5 **Server requirement**: all servers, specifically including JPEG 2000 unaware servers

4.6 **Signalling requirement**: any

4.7 **Other**: the key aspect of this scenario is that the implementer does not have the liberty to re-write the entire application where the client software resides.

Note: Feature requirement submissions are encouraged, even if they apply to a wide number of use-case scenarios (e.g. Application developer requirements)

Use Case #2

Source: Australian Defence

1. **Smart Dissemination Service (for very large imagery) – Reliable medium to high bandwidth network environment** (Case 1)
2. **Context Information**
   - 2.1. **Market application** – Strategic military environment, remote sensing imagery services, mapping agencies, raster map servers
   - 2.2. **Client platform characteristics** - Desktop
   - 2.3. **Server platform characteristics** - Workstations
   - 2.4. **Network characteristics** – TCP/IP networks - Mbps down to approximately 56 kbps for example domestic raster map servers.
   - 2.5. **JPEG2000 image characteristics** – Very large images (32kx32k)
3. **Description of Use Case**
   - 3.1. **Introduction** - Strategic military applications require access to intelligence, reconnaissance and surveillance imagery. Often this imagery is very large. Clients may be located at remote sites. Downloading the imagery is not possible due to network bandwidth limitations, however, the user may want to interact with the imagery (pan, zoom, cut, save region) as if it was on their desktop. JPIP would provide smart dissemination for clients to access the multiresolution data structure in JPEG2000 imagery, without having to download the entire file.
   - 3.2. **Description** of the steps of operation –
     - 3.2.1. Log onto the image server (assumes that the image to be viewed has already been identified)
     - 3.2.2. Download a thumbnail or overview image from the server to the client – e.g. this may be the LL subband from the JPEG2000 image data
     - 3.2.3. Client paints the thumbnail or overview image on the screen for the user
     - 3.2.4. User selects a region on the image and clicks the zoom button to zoom into that region
     - 3.2.5. Client immediately paints this region using the data already available on the client – e.g. maybe the LL subband data
     - 3.2.6. Client sends JPIP request for the new image data in the current view-port - resolution, etc.
     - 3.2.7. Server returns with JPEG2000 image data that will provide more resolution for the current image view-port
     - 3.2.8. Client progressively updates the displayed image using the new data as it becomes available
     - 3.2.9. Client receives more data from the server and updates the image for the user
     - 3.2.10. User pans to the left
     - 3.2.11. Client paints an image for this new view-port using the data it already holds
     - 3.2.12. Client sends JPIP requests for the new view-port
     - 3.2.13. The server only send to the client new data required for this new view-port (client or server knows what data the client holds in its JPIP cache)
     - 3.2.14. Client receives the data and paints the new updated image
     - 3.2.15. User zooms out of image
     - 3.2.16. Client paints view-port with data it holds
     - 3.2.17. Client makes JPIP request for new view-port
     - 3.2.18. Server only sends new data required for this new view-port

3.2.19. Client receives data and paints an update

3.2.20. User selects an image region and presses the “save region” button (this will save or writes this selected region into a legal JPEG2000 file)

3.2.21. JPIP request for that region is sent to the server

3.2.22. Server sends new data – (For the implicit JPIP request paradigm, the server would then send a “DONE” signal to signal that the client should now have all of the data required to write a JPEG2000 file of the selected image region at the selected resolution, quality etc. – If the server did not send the DONE signal, then perhaps the user could elect to write out the data in its cache, even though this would not meet the intended resolution, quality, etc parameters)

3.2.23. User logs off

3.2.24. Some indeterminable time later, the user logs onto the image server and selects the same image

3.2.25. Client or server knows by some means what the client holds in its JPIP cache w.r.t. the selected image

3.2.26. Client paints image with data it holds in its cache.

3.2.27. Client sends JPIP request based on current view-port – data exchange continues as above (See the Smart Dissemination Service – Case 2 for details of cache purging)

3.2.28. Response satisfies user requirements

4. **Feature Requirements** (for each feature / requirement specification)

4.1. **Title** – change of viewport?

4.1.1. **Description** – This is the message that a client sends to the server so that it can “know” what data to return - It is related to the user’s current “view-port” into the image – View-port specifies x1, y1, x2, y2, resolution, quality (maybe coding passes or quality layers), components etc.

4.1.2. **Details of feature requirem**ent

4.1.2.1. Client implications – Client sends the message therefore the client must be able to formulate the request based on the user’s view-port parameters

4.1.2.2. Server implications – Must be able to interpret the message and generate the appropriate response

4.1.2.3. Signalling implications – No assumptions are or should be made about the ordering of messages or server data

4.1.2.4. Other implications –

4.2. Title – Data transmission efficiency & Client-side JPIP cache

4.2.1. Description – The data transmission efficiency is the ratio of the JPEG2000 image data sent to a client that is required to complete a JPIP request (view-port, resolution, etc) to the total data sent (required JPEG2000 image data, excessive JPEG2000 image data, redundant JPEG2000 image data [i.e. data that has already been received by the client], plus signalling overheads) - The client is able to cache JPEG2000 image data sent by the server

4.2.2. Details of feature requirement

4.2.2.1. Client implications – Client must be able to cache the data, and keep track of the cache contents – This function could be an aspect of the JPEG2000 decompressor component on the client

4.2.2.2. Server implications – For maximum efficiency, the server should be able to “know” what JPEG2000 image data is redundant – i.e. already held in the client’s cache – and it should not resend this data

4.2.2.3. Signalling implications – Signalling overheads should be as small as possible

4.2.2.4. Other implications –

4.3. Title – Client response to user’s requirements

4.3.1. **Description –** The user wants to view a view-port into the image on the display

4.3.2. **Details of feature requi**rement

4.3.2.1. Client implications – Client responds immediately by painting an image onto the display using the data held in its JPIP cache – The client sends an update to the server with a new JPIP request based on the new view-port requirements

4.3.2.2. Server implications – Server responds promptly with JPEG2000 image data for this new view-port

4.3.2.3. Signalling implications – Defined JPIP client requests and defined JPIP replies comprising of JPEG2000 image data

4.3.2.4. Other implications –

4.4. Title – Session persistence

4.4.1. Description – The client can utilise data in its JPIP cache in subsequent sessions

4.4.2. Details of feature requirement

4.4.2.1. Client implications – Either the client signals to the server what is the state of its cache, or there is some cumulative session record at the server (with a UUID), or through some other means, the client is able to inform the server what data would be redundant for a JPIP request; alternatively, the client might make explicit the data it is requesting, thus the server will not have to hold any state about the client's cache

4.4.2.2. Server implications – for non-explicit JPIP requests (ones where the JPIP request explicitly defines the JPEG2000 data to be served) the server will have to hold and update a model of the client's cache

4.4.2.3. Signalling implications – cache management signals may be required

4.4.2.4. Other implications

4.5. Title – Save client data

4.5.1. Description – Save the client data into a JPEG2000 image – Either save all of the cache, or save a selected region of the image

4.5.2. Details of feature requirement

4.5.2.1. Client implications – Client can write a JPEG2000 file – Perhaps if it is a selected image region to be saved, the client sends a special JPIP request for all of the required data first before the write function

4.5.2.2. Server implications – Server may have to send relevant data in response to the JPIP request

4.5.2.3. Signalling implications – Perhaps a special JPIP query

4.5.2.4. Other implications

Use Case

1. **Smart Dissemination Service (for very large imagery) – Low bandwidth error prone network environment (Case 2)**
2. **Context Information**
   - 2.1. Market application – Tactical military environment, mobile devices
   - 2.2. Client platform characteristics – PDA or Laptop
   - 2.3. Server platform characteristics – Laptop or Desktop
   - 2.4. Network characteristics – 3 to 20 kbps pseudo-UDP network with very poor BER
   - 2.5. JPEG2000 image characteristics – From small (1kx1k) to very large (32kx32k)
3. **Description of Use Case**
   - 3.1. **Introduction** – Tactical military applications require access to image libraries. (Note that it may involve the tactical user being both a client (viewing remote images) and a server (sharing images with others). For brevity and clarity, only the tactical client application will be presented below.) The user is located at remote sites with very low capacity (down to 3 kbps) and poor error performance (down to 10^{-3}BER). Downloading the imagery is not possible due to network restrictions, however, the user want to interact with the imagery (pan, zoom, cut) as if it was on their PDA. JPIP would provide smart dissemination for clients, without having to download the entire file.
   - 3.2. **Description of the steps of operation** –
     - 3.2.1. Log onto the image server (assumes that one has already discovered what image one wants to view)
     - 3.2.2. Download a thumbnail of the image from the server to the client – e.g. this may be the LL subband from the JPEG2000 image data – where the bit-budget for the transaction can be set by the user
     - 3.2.3. Client paints the thumbnail on the screen for the user
     - 3.2.4. User selects a region on the image and clicks the zoom button to zoom into that region
     - 3.2.5. Client immediately paints this region using the data it already holds – e.g. maybe the LL subband data
     - 3.2.6. Client sends JPIP request for that image view-port, resolution, etc. – Including a bit-budget and error-resilience mode
     - 3.2.7. Server returns with JPEG2000 image data that will provide more resolution for the current image view-port – The data transmitted should optimise the painted image result w.r.t. bit budget – Certain data (such as header information and very significant quality layers) can be sent with greater error-resilience while other data may have lower error-resilience – This may be within the JPEG2000 error-resilience technology or it may employ other means – i.e. the data may be sent using different logical channels
     - 3.2.8. Client paints an update of the image using the new data
     - 3.2.9. Client receives more data from the server and paints an update image for the user
     - 3.2.10. User pans to the left
     - 3.2.11. Client paints an image for this new view-port using the data it already holds
     - 3.2.12. Client sends JPIP requests for the new view-port
     - 3.2.13. The server only send to the client new data required for this new view-port (client or server knows what the client holds in its JPIP cache – presently there has been no cache purge at the client)
     - 3.2.14. Client receives the data and paints the new updated image
     - 3.2.15. Client decides that its cache is getting full – so it purges some data (For example, this might be the part of the LL subband that is the greatest distance away from the part of the image now under view, but note that this example assumes a certain type

of implementation for JPIP – use it only as an example to describe the purge function captured in this Use Case)

3.2.16. Client and/or server manage this cache purge appropriately

3.2.17. User zooms out of image

3.2.18. Client paints view-port with data it holds – but some data is required that was just purged

3.2.19. Client makes JPIP request for new view-port

3.2.20. Server sends data

3.2.21. Client receives data and paints an update

3.2.22. Client has a data overrun and cannot cache all the new data it receives

3.2.23. Client purges data out of its cache and client and/or server manages purge – includes that data the client has just purged plus data lost from the overrun

3.2.24. Server responds to this purge appropriately – Perhaps client resends JPIP request for current view-port

3.2.25. Server resends that data relevant to the view-port i.e. that relevant portion which was just "purged", or more exactly that which was lost data from the overrun

3.2.26. Client paints update

3.2.27. User is always happy with the response

4. **Feature Requirements** (for each feature / requirement specification)

4.1. Title – JPIP client request messages

4.1.1. **Description** – As in Part I above, but with extra features for managing the bit-budget and the error-resilience modes for the transactions

4.1.2. Details of feature requirement

4.1.2.1. Client implications –

4.1.2.2. Server implications –

4.1.2.3. Signalling implications –

4.1.2.4. Other implications – The "intelligence" for performing the error-resilience management may be in either the client or the server – For instance, the client may acknowledge receipt of data "packets" (i.e. transport packets and not JPEG2000 image data packets), and thus the server can estimate what actions to take (change mode, resend packets, etc)

4.2. Title – Client cache management

4.2.1. **Description** – To optimise data transmission efficiency, the client holds a JPIP cache – the server should not send redundant data, i.e. data already in the client cache

4.2.2. Details of feature requirement

4.2.2.1. Client implications – Client must hold and manage its cache – including purging and book-keeping

4.2.2.2. Server implications – Either the server "knows" what is in the client cache (implicit request mode), or (for explicit requests) the client manages the optimisation of data requests

4.2.2.3. Signalling implications – May need cache management signalling

4.2.2.4. Other implications

4.3. **Title** – Multi-channel data for graded error-resilience

4.3.1. **Description** – Different data can be sent on different channels that may have different error protection – For example, headers and important image data could have high error protection, while less important image data could have lower error protection

4.3.2. **Details of feature requirem**ent

4.3.2.1. Client implications – Must negotiate the channels

4.3.2.2. Server implications – Must negotiate & establish the channels

4.3.2.3. Signalling implications - Negotiation

4.3.2.4. Other implications – Perhaps the channels could be TCP and UDP; or perhaps they are UDP packets with error correction (turbo-codes)

4.4. Title

4.4.1. Definition

4.4.2. Details of feature requirement

4.4.2.1. Client implications

4.4.2.2. Server implications

4.4.2.3. Signalling implications

4.5. Other implications Title

4.5.1. Definition

4.5.2. Details of feature requirement

4.5.2.1. Client implications

4.5.2.2. Server implications

4.5.2.3. Signalling implications

4.5.2.4. Other implications

Use Case #3

Source: Australia Defence (2)

1. **Smart Dissemination Service (for very large imagery) – Reliable medium to high bandwidth network environment (Case 1)**
2. **Context Information**
   - 2.1. Market application – Strategic military environment, remote sensing imagery services, mapping agencies, raster map servers
   - 2.2. Client platform characteristics - Desktop
   - 2.3. Server platform characteristics - Workstations
   - 2.4. Network characteristics – TCP/IP networks - Mbps down to approximately 56 kbps for example domestic raster map servers.
   - 2.5. JPEG2000 image characteristics – Very large images (32kx32k)
3. **Description of Use Case**
   - 3.1. Introduction - Strategic military applications require access to intelligence, reconnaissance and surveillance imagery. Often this imagery is very large. Clients may be located at remote sites. Downloading the imagery is not possible due to network bandwidth limitations, however, the user may want to interact with the imagery (pan, zoom, cut, save region) as if it was on their desktop. JPIP would provide smart dissemination for clients to access the multiresolution data structure in JPEG2000 imagery, without having to download the entire file.
   - **3.2. Description of the steps of operation –**
     - 3.2.1. Log onto the image server (assumes that the image to be viewed has already been identified)
     - 3.2.2. Download a thumbnail or overview image from the server to the client – e.g. this may be the LL subband from the JPEG2000 image data
     - 3.2.3. Client paints the thumbnail or overview image on the screen for the user
     - 3.2.4. User selects a region on the image and clicks the zoom button to zoom into that region
     - 3.2.5. Client immediately paints this region using the data already available on the client – e.g. maybe the LL subband data
     - 3.2.6. Client sends JPIP request for the new image data in the current view-port - resolution, etc.
     - 3.2.7. Server returns with JPEG2000 image data that will provide more resolution for the current image view-port
     - 3.2.8. Client progressively updates the displayed image using the new data as it becomes available
     - 3.2.9. Client receives more data from the server and updates the image for the user
     - 3.2.10. User pans to the left
     - 3.2.11. Client paints an image for this new view-port using the data it already holds
     - 3.2.12. Client sends JPIP requests for the new view-port
     - 3.2.13. The server only send to the client new data required for this new view-port (client or server knows what data the client holds in its JPIP cache)
     - 3.2.14. Client receives the data and paints the new updated image
     - 3.2.15. User zooms out of image
     - 3.2.16. Client paints view-port with data it holds
     - 3.2.17. Client makes JPIP request for new view-port

3.2.18. Server only sends new data required for this new view-port

3.2.19. Client receives data and paints an update

3.2.20. User selects an image region and presses the “save region” button (this will save or writes this selected region into a legal JPEG2000 file)

3.2.21. JPIP request for that region is sent to the server

3.2.22. Server sends new data – (For the implicit JPIP request paradigm, the server would then send a “DONE” signal to signal that the client should now have all of the data required to write a JPEG2000 file of the selected image region at the selected resolution, quality etc. – If the server did not send the DONE signal, then perhaps the user could elect to write out the data in its cache, even though this would not meet the intended resolution, quality, etc parameters)

3.2.23. User logs off

3.2.24. Some indeterminable time later, the user logs onto the image server and selects the same image

3.2.25. Client or server knows by some means what the client holds in its JPIP cache w.r.t. the selected image

3.2.26. Client paints image with data it holds in its cache.

3.2.27. Client sends JPIP request based on current view-port – data exchange continues as above (See the Smart Dissemination Service – Case 2 for details of cache purging)

3.2.28. Response satisfies user requirements

4. **Feature Requirements** (for each feature / requirement specification)

4.1. Title – JPIP client request messages

4.1.1. Description – This is the message that a client sends to the server so that it can “know” what data to return - It is related to the user’s current “view-port” into the image – View-port specifies x1, y1, x2, y2, resolution, quality (maybe coding passes or quality layers), components etc.

4.1.2. Details of feature requirement

4.1.2.1. Client implications – Client sends the message therefore the client must be able to formulate the request based on the user’s view-port parameters

4.1.2.2. Server implications – Must be able to interpret the message and generate the appropriate response

4.1.2.3. Signalling implications – No assumptions are or should be made about the ordering of messages or server data

4.1.2.4. Other implications –

4.2. Title – Data transmission efficiency & Client-side JPIP cache

4.2.1. Description – The data transmission efficiency is the ratio of the JPEG2000 image data sent to a client that is required to complete a JPIP request (view-port, resolution, etc) to the total data sent (required JPEG2000 image data, excessive JPEG2000 image data, redundant JPEG2000 image data [i.e. data that has already been received by the client], plus signalling overheads) - The client is able to cache JPEG2000 image data sent by the server

4.2.2. Details of feature requirement

4.2.2.1. Client implications – Client must be able to cache the data, and keep track of the cache contents – This function could be an aspect of the JPEG2000 decompressor component on the client

4.2.2.2. Server implications – For maximum efficiency, the server should be able to “know” what JPEG2000 image data is redundant – i.e. already held in the client’s cache – and it should not resend this data

4.2.2.3. Signalling implications – Signalling overheads should be as small as possible

4.2.2.4. Other implications –

4.3. Title – Client response to user's requirements

4.3.1. Description – The user wants to view a view-port into the image on the display

4.3.2. Details of feature requirement

4.3.2.1. Client implications – Client responds immediately by painting an image onto the display using the data held in its JPIP cache – The client sends an update to the server with a new JPIP request based on the new view-port requirements

4.3.2.2. Server implications – Server responds promptly with JPEG2000 image data for this new view-port

4.3.2.3. Signalling implications – Defined JPIP client requests and defined JPIP replies comprising of JPEG2000 image data

4.3.2.4. Other implications –

4.4. Title – Session persistence

4.4.1. Description – The client can utilise data in its JPIP cache in subsequent sessions

4.4.2. Details of feature requirement

4.4.2.1. Client implications – Either the client signals to the server what is the state of its cache, or there is some cumulative session record at the server (with a UUID), or through some other means, the client is able to inform the server what data would be redundant for a JPIP request; alternatively, the client might make explicit the data it is requesting, thus the server will not have to hold any state about the client's cache

4.4.2.2. Server implications – for non-explicit JPIP requests (ones where the JPIP request explicitly defines the JPEG2000 data to be served) the server will have to hold and update a model of the client's cache

4.4.2.3. Signalling implications – cache management signals may be required

4.4.2.4. Other implications

4.5. Title – Save client data

4.5.1. Description – Save the client data into a JPEG2000 image – Either save all of the cache, or save a selected region of the image

4.5.2. Details of feature requirement

4.5.2.1. Client implications – Client can write a JPEG2000 file – Perhaps if it is a selected image region to be saved, the client sends a special JPIP request for all of the required data first before the write function

4.5.2.2. Server implications – Server may have to send relevant data in response to the JPIP request

4.5.2.3. Signalling implications – Perhaps a special JPIP query

4.5.2.4. Other implications

Use Case

1. **Smart Dissemination Service (for very large imagery) – Low bandwidth error prone network environment (Case 2)**
2. **Context Information**
   - 2.1. Market application – Tactical military environment, mobile devices
   - 2.2. Client platform characteristics – PDA or Laptop
   - 2.3. Server platform characteristics – Laptop or Desktop
   - 2.4. Network characteristics – 3 to 20 kbps pseudo-UDP network with very poor BER
   - 2.5. JPEG2000 image characteristics – From small (1kx1k) to very large (32kx32k)
3. **Description of Use Case**
   - 3.1. **Introduction** – Tactical military applications require access to image libraries. (Note that it may involve the tactical user being both a client (viewing remote images) and a server (sharing images with others). For brevity and clarity, only the tactical client application will be presented below.) The user is located at remote sites with very low capacity (down to 3 kbps) and poor error performance (down to 10^{-3}BER). Downloading the imagery is not possible due to network restrictions, however, the user want to interact with the imagery (pan, zoom, cut) as if it was on their PDA. JPIP would provide smart dissemination for clients, without having to download the entire file.
   - **3.2. Description of the steps of operation –**
     - 3.2.1. Log onto the image server (assumes that one has already discovered what image one wants to view)
     - 3.2.2. Download a thumbnail of the image from the server to the client – e.g. this may be the LL subband from the JPEG2000 image data – where the bit-budget for the transaction can be set by the user
     - 3.2.3. Client paints the thumbnail on the screen for the user
     - 3.2.4. User selects a region on the image and clicks the zoom button to zoom into that region
     - 3.2.5. Client immediately paints this region using the data it already holds – e.g. maybe the LL subband data
     - 3.2.6. Client sends JPIP request for that image view-port, resolution, etc. – Including a bit-budget and error-resilience mode
     - 3.2.7. Server returns with JPEG2000 image data that will provide more resolution for the current image view-port – The data transmitted should optimise the painted image result w.r.t. bit budget – Certain data (such as header information and very significant quality layers) can be sent with greater error-resilience while other data may have lower error-resilience – This may be within the JPEG2000 error-resilience technology or it may employ other means – i.e. the data may be sent using different logical channels
     - 3.2.8. Client paints an update of the image using the new data
     - 3.2.9. Client receives more data from the server and paints an update image for the user
     - 3.2.10. User pans to the left
     - 3.2.11. Client paints an image for this new view-port using the data it already holds
     - 3.2.12. Client sends JPIP requests for the new view-port
     - 3.2.13. The server only send to the client new data required for this new view-port (client or server knows what the client holds in its JPIP cache – presently there has been no cache purge at the client)
     - 3.2.14. Client receives the data and paints the new updated image
     - 3.2.15. Client decides that its cache is getting full – so it purges some data (For example, this might be the part of the LL subband that is the greatest distance away from the part of the image now under view, but note that this example assumes a certain type

of implementation for JPIP – use it only as an example to describe the purge function captured in this Use Case)

3.2.16. Client and/or server manage this cache purge appropriately

3.2.17. User zooms out of image

3.2.18. Client paints view-port with data it holds – but some data is required that was just purged

3.2.19. Client makes JPIP request for new view-port

3.2.20. Server sends data

3.2.21. Client receives data and paints an update

3.2.22. Client has a data overrun and cannot cache all the new data it receives

3.2.23. Client purges data out of its cache and client and/or server manages purge – includes that data the client has just purged plus data lost from the overrun

3.2.24. Server responds to this purge appropriately – Perhaps client resends JPIP request for current view-port

3.2.25. Server resends that data relevant to the view-port i.e. that relevant portion which was just "purged", or more exactly that which was lost data from the overrun

3.2.26. Client paints update

3.2.27. User is always happy with the response

4. **Feature Requirements** (for each feature / requirement specification)

4.1. Title – JPIP client request messages

4.1.1. Description – As in Part I above, but with extra features for managing the bit-budget and the error-resilience modes for the transactions

4.1.2. Details of feature requirement

4.1.2.1. Client implications –

4.1.2.2. Server implications –

4.1.2.3. Signalling implications –

4.1.2.4. Other implications – The "intelligence" for performing the error-resilience management may be in either the client or the server – For instance, the client may acknowledge receipt of data "packets" (i.e. transport packets and not JPEG2000 image data packets), and thus the server can estimate what actions to take (change mode, resend packets, etc)

4.2. Title – Client cache management

4.2.1. Description – To optimise data transmission efficiency, the client holds a JPIP cache – the server should not send redundant data, i.e. data already in the client cache

4.2.2. Details of feature requirement

4.2.2.1. Client implications – Client must hold and manage its cache – including purging and book-keeping

4.2.2.2. Server implications – Either the server "knows" what is in the client cache (implicit request mode), or (for explicit requests) the client manages the optimisation of data requests

4.2.2.3. Signalling implications – May need cache management signalling

4.2.2.4. Other implications

4.3. Title – Multi-channel data for graded error-resilience

4.3.1. Description – Different data can be sent on different channels that may have different error protection – For example, headers and important image data could have high error protection, while less important image data could have lower error protection

4.3.2. Details of feature requirement

4.3.2.1. Client implications – Must negotiate the channels

4.3.2.2. Server implications – Must negotiate & establish the channels

4.3.2.3. Signalling implications - Negotiation

4.3.2.4. Other implications – Perhaps the channels could be TCP and UDP; or perhaps they are UDP packets with error correction (turbo-codes)

Use Case #4

Source: Canon

1 Use-case scenario

1.1 Title: PC-based browsing of digital photography albums

2 Context

2.1 **Market application** (include info about why is JPIP needed?): digital photography, among others

2.2 **Client platform characteristics** (PDA, desktop, mobile phone, …): PC

2.3 **Server platform characteristics** (PDA, desktop, large server, number of clients to serve, …): any platform containing digital photography albums

2.4 **Network** (bandwidth, delay, errors, architecture, QoS …): 56kbps

2.5 **JPEG2000 Image Characteristics**: any (images could have come from anywhere)

2.6 Other characteristics

3 Description of use-case

3.1 Introduction: the user makes zoom in/out and pan operations on a JPEG2000 image located at the server

3.2 Description in steps

3.2.1 User makes zoom and/or pan request

3.2.2 A request is made to the server by the client application

3.2.3 A response a received from the server, and processed by the client application

3.2.4 The user quickly views the result of the request

3.3 Feature 1

3.3.1 Title: transmission of minimum codestream portions needed from server to client

3.3.2 Definition: the client has obtained information about the contents of the jp2 file on the server, and is capable of determining exactly which portion of the codestream he wishes to obtain to satisfy a user request of the type zoom in/out, pan

3.3.3 Detail of feature required

3.3.4 Client requirement: capability for the client to request specific portions of jp2 codestream

3.3.5 Server requirement: capability for the server to deliver specific portions of jp2 codestream

3.3.6 Signalling requirement: requested codeblocks, group of codeblocks, precincts, …

Other

Use Case #5

Source: IBM

1.1 Title
Security Video Tracking with ROI Hierarchy

1.2 **Context**
Monitoring on-line video

1.2.1 **Market Application** (include info about why JPIP needed)
JPIP required to provide communication about ROI position/size/movement, so minimising network traffic while providing best quality subject to bandwidth constraints

1.2.2 User or Application developer
This benefits the end-user

1.2.3 Client platform characteristics (PDA, Desktop,mobile phone,…)
Desktop, or even PDA-like device mounted in police-car/helicopter/…

1.2.4 Server platform characteristics
Whatever is hosting/recording the video sourced at possibly many camera-sites.

1.2.5 Network (bandwidth, delay, errors, architecture, QoS,…)
May lose packets if monitoring station located in moving car,
so must be error-tolerant

1.2.6 JPEG2000 Image descriptions
TV-like resolution images, maybe 25 frames/sec from monitoring camera(s), but the client may only want to view a small portion of that image at “good” quality.

1.3 **Description of use-case**
Monitoring client may select region of image that is of prime interest, creating a ROI, eg a box around a “suspect”. Client app might track the motion of this automatically from frame-to-frame, and as this ROI moves, client sends updates to server. Server responds by sending the latest ROI updated image at “best” quality, while surrounding areas may updated less frequently, and at lower quality.

1.3.1 Introduction

1.3.2 Description in steps

2 Features required

2.1 **Title**
Hierarchical ROI

2.2 **Definition**
Ability to define 1 or more ROIs, and associate them with guaranteed-delivery channel,

and relegating low-importance parts of the image (anything NOT associated with an ROI) to a low-priority channel which will be the first to discard packets if bandwidth insufficient.

2.3 Detail of feature required

2.3.1 Client requirement
Ability to negotiate/command server to allocate QoS resources to specific ROI channel(s)

2.3.2 Server requirement
Ability to prioritise image-data delivery according to ROI priority and manage the available bandwidth accordingly. Might discard colour information as a last resort in an attempt to maintain frame-rate, or use other heuristics.

2.3.3 Signalling requirement

2.3.4 Other

Use Case #6

Source: IBM

1.1 **Title**
Transcoding Usecase

1.2 **Context**
Precision image-display

1.2.1 **Market Application**(include info about why JPIP needed)
Anyone requiring precisely-sized image display

1.2.2 **User or Application developer**
Benefits the end-user

1.2.3 **Client platform characteristics** (PDA, Desktop,mobile phone,…)
Any

1.2.4 **Server platform characteristics**
Any

1.2.5 **Network** (bandwidth, delay, errors, architecture, QoS,…)
Any

1.2.6 JPEG2000 Image descriptions

1.3 **Description of use-case** (in steps)

1.3.1 **Introduction**
a Service-provider e.g. web-based home-shopping retailer may require that their images are precisely sized to fit the displayed web-page exactly. This will require re-scaling, either at client/server end.

1.3.2 **Description in steps**
Client device informs server what the image-display-area characteristics are. Client & Server might negotiate to decide who is to do the actual resizing.

2 **Features required**
Ability to negotiate transcoding issues between client & server. Tiny client like memory-constrained PDA may not have any capability to do this, whereas a fast desktop with fast Lan connection may be able to do thus faster than the heavily-loaded server supplying the data. Negotiations may even include estimating lan-delays as well as cpu-delays at each end, and are likely to be application-specific.

2.1 **Title**
Transcoding negotiability

2.2 Definition

2.3 **Detail of feature required**
See description above

2.3.1 Client requirement

2.3.2 Server requirement

2.3.3 Signalling requirement

2.3.4 Other

Use Case #7

Source: IBM

1.1 **Title**
UserExit Usecase

1.2 **Context**
Any application using JPIP that needs to communicate with server over JPIP connection

1.2.1 **Market Application**(include info about why JPIP needed)
unspecified

1.2.2 **User or Application developer**
This benefits both the developer, and makes application building easier.

1.2.3 **Client platform characteristics** (PDA, Desktop,mobile phone,…)
Any

1.2.4 **Server platform characteristics**
Any

1.2.5 **Network** (bandwidth, delay, errors, architecture, QoS,…)
Any

1.2.6 **JPEG2000 Image descriptions**
Any

1.3 Description of use-case (in steps)

1.3.1 **Introduction**
Server application is in process of supplying data to client, but extra authentication is needed before the client is allowed to view in greater detail, or maybe credit-card details & price needs negotiation in order to pay for the enhanced service requested. If UserExits (stub API calls) are provided in JPIP, these can be configured to provide additional functionality not yet considered in this standard.

1.3.2 **Description in steps**
Client or Server has the ability to invoke a function which passes a request/data to the Server/Client. The interpretation of this is application-dependent.

2 **Features required**
A general purpose function to enable configurable communication.

2.1 **Title**
UserExit Function

2.2 **Definition**
See description above

2.3 Detail of feature required

2.3.1 Client requirement

2.3.2 Server requirement

2.3.3 Signalling requirement

2.3.4 Other

Use Case #8

Source: JPM

1. Use Case for JPM Document Access from Client PC

2. Context Information

2.1. Market application - (include info about why JPIP needed)

Web access to document archives is needed which provide efficient navigation from page-to-page and between the layout objects (text columns, photographs, etc.) on a page

2.2. Client platform characteristics - (PDA, desktop, mobile phone) –

A desktop client is envisioned in this use case.

2.3. Server platform characteristics - (PDA, desktop, large server, number of clients to serve)

A typical web server, possibly with a large number of clients.

2.4. Network characteristics - (bandwidth, delay, errors, architecture, QoS)

Typical web connection: range of rates from 56K to DSL and beyond

2.5. JPEG2000 image characteristics –

JPM file format files with mix of compression types including any of JPEG 2000, JPEG, T.6 or JBIG2

3. Description of Use Case

3.1. Introduction

A user views a portion of a document through a web browser and navigates to other portions of the document, in some cases browsing or skimming and in other cases reading in depth.

JPM files consist of hierarchies of page collections, pages and layout objects. Layout objects are each composed of as many as 2 image files having one of four compression types.

3.2. Description of the steps of operation -

3.2.1. Step 1

Using a web browser, the user performs a text search on a search server (server A) searching on OCR metadata across a collection of JPM documents. This search server (which is also a JPIP server) produces a search results JPM document containing 3 snippets of image data corresponding to the OCR data from 3 layout objects in 3 different JPM documents on 3 different JPIP servers. Each of these snippets (search hits 1, 2 and 3) is completely defined by a jpip:// URL pointing to sub-rectangles of pages in their source JPM files. These documents live on servers B, C and D.

3.2.2. Step 2

The user clicks into one of the snippets, which has a URL to a specific layout object on a specific page in a specific page collection in a specific JPM document on a specific JPIP server (say snippet 1 which points to the 1.jpm file on server B).

3.2.3. Step 3

JPIP server B returns layout object 4 of page 7 of 1.JPM and the user views it, with the specified layout object (say a text column) of the page filling the screen. Additional quality data is continuously being sent, refining the rendering of the layout object.

3.2.4. Step 4

The user zooms back to view the entire page, then clicks onto a different layout object.

3.2.5. Step 5

The user continues reading by going to the next page of 1.jpm, retrieving that page's URL from the page collection of 1.jpm. The page header of that page is retrieved, then the page thumbnail. This is displayed to the user. Next, the lowest quality layers of the page's layout objects are retrieved and displayed, incrementally covering up the page thumbnail.

3.2.6. Step 6

The user clicks onto a layout object of the page. The browser fills the screen with that layout object, terminates pending requests for the refinement of the other layout objects of the page and requests even better refinement data for the new focus layout object.

4. Feature Requirements (for each feature / requirement specification)

4.1. Browse JPM file which references data on multiple JPIP servers

4.1.1. Description –

This means either that the client talks to multiple JPIP servers at once or that JPIP server providing the main JPM file must serve as a proxy to go get data from multiple JPIP servers on behalf of the user.

4.1.2. Details of feature requirement

4.1.2.1. Client implications -

4.1.2.2. Server implications -

4.1.2.3. Signalling implications -

4.1.2.4. Other implications -

4.2. Retrieve sub-objects from a JPM file, including page collections, pages, layout objects, image objects, thumbnails, metadata at various levels

4.2.1. Description –

This requires protocol elements which allow a nested set of prefixes to identify codestreams within a JPM file

4.2.2. Details of feature requirement

4.2.2.1. Client implications -

4.2.2.2. Server implications -

4.2.2.3. Signalling implications -

4.2.2.4. Other implications –

4.3. Support byte serving of codestreams which are not JPEG2000 codestreams, including JPEG2000, JPEG, T.6, and JBIG2

4.2.1. Description -

4.2.2. Details of feature requirement

4.2.2.1. Client implications -

4.2.2.2. Server implications -

4.2.2.3. Signalling implications -

4.2.2.4. Other implications -

Use Case #9

Source: Kodak

1. Title : server controlled image transfer based on client capabilities

2. Context Information

2.a. Market application : consumer makes lot of sense, but may apply to many segments

2.b. Client platform characteristics : all

2.c. Server platform characteristics : indifferent, but probably can handle lot of concurrent users

2.d. Network characteristics (bandwidth, delay, errors, architecture, QoS) : all

2.e. JPEG2000 image characteristics : the image has at least one resolution per family of devices (printer, PC, TV, PDA, Smartphone, phones)

3. Description of Use Case

3.a. Introduction

In this configuration, the system is made very efficient by the knowledge the server has on the client capabilities. The server sends only the image data that best fit the client characteristics (color, resolution, quality). The client then may has the opportunity to make complementary requests to get further data (not part of this use case), but one can assume that the interaction may be limited to the first set of data sent by the server in many cases.

3.b. Description of the steps of operation

Example 1 :

C→S : Client askes to open a particular file for intelligent download

S→C : file exists, server supports analysis of client capabilities (class i). Server also responds with a initial set of image data (at some base resolution and quality)

C→S : client sends capabilities. (Namely : compatibility, colorspace preference, resolution, RAM, max bandwidth, color depth, vendor defined capablities, standard and vendor features if not already provided in other fields.)

S→C : server returns any additional image data needed to bring the client display to an optimal state

Example 2:

C→S : Client asks for the image (possibly at a specific region (spatial region and/or component set) of the image), targeted to a specific set of capabilities (client transmits desired region and capability and preference specifcation in the same request)

S→C : Server returns optimal set of image data

4. Feature Requirements

4.a. Title : support for client capabilities by the server

i. Definition : the server must be able to specify upfront if it has support for client capabilities analysis. A system of « classes is used to precise the level of understanding the server has.

ii. Details of feature requirement

1. Client implications : client must support the system of « classes »

2. Server implications : server must support the system of classes (class 0 = no support for client capabilities)

3. Signalling implications : class supported is sent by the server

4. Other implications

4.b. Title : sending of the capabilities by the client

i. Definition : the clients sends under a textual or binary form its capabilities

ii. Details of feature requirement

1. Client implications : client must be able to get the necessary informations from the platform (screen resolution,…). Client must also be able to structure the data.

2. Server implications

3. Signalling implications : sending of the capabilities

4. Other implications

4.c. Title : analysis of the capabilities by the server

i. Definition : the server gets the capabilities, analyses them and returns the best suited image data

ii. Details of feature requirement

1. Client implications

2. Server implications : read client capabilities, analyze them, select best suited data (may have transcode before)

3. Signalling implications

4. Other implications

Use Case #10

Source: Kodak

1. Title : Query XML metadata

2. Context Information

2.a. Market application : all

2.b. Client platform characteristics : all

2.c. Server platform characteristics : indifferent, but probably can handle lot of concurent users

2.d. Network characteristics (bandwidth, delay, errors, architecture, QoS) : all

2.e. JPEG2000 image characteristics : contain XML metadata

3. Description of Use Case

3.a. Introduction

A client desires to get specific elements from a specific set of XML documents which may be embedded within the JP2 file. It desires to minimize the number of round trips required, so it would like to be able to specify the targetted XML boxes and the desired elements in a single request. The server then returns the requested elements, if they exist.

Example 1: If the image contains an SVG map overlay of military targets, the client desires to see all of the locations of tanks in the image. It requests that the server locate any map overlays and then extract the tank location elements.

Example 2: A photo printer wants to perform red-eye correction before printing the image. To do so, it needs to know if the flash was fired when the image was captured. This information is stored in the Image Creation metadata.

Alternately, the server builds a catalog of the metadata (including binary metadata) and returns that. The client then references specific documents specified in the catalog

3.b. Description of the steps of operation

C→S : client specifies, using XPath, the characteristics of the target XML documents, and then specifies how the desired elements are identified

S→C : server either replies with an empty reply (if the desired metadata is not in the file) or with the NodeList if the desired metadata is found.

4. Feature Requirements

4.a. Title : Support for complex metadata queries

ii. Definition : the client must be able to specify portions of XML documents embedded within the file

ii. Details of feature requirement

1. Client implications : has the option of using XPath to specify the desired metadata

2. Server implications : NA

3. Signalling implications : NA

4. Other implications

4.b. Title : support for the return of NodeLists or other XPath result types

iii. Definition : the server returns the requested elements

iv. Details of feature requirement

1. Client implications : NA

2. Server implications : the server must be able to parse the XPath request, generate a catalog of all of the metadata in the file, parse the XML documents and extract the requested elements

3. Signalling implications : Return the NodeList or other XPath result types

4. Other implications

4.c. Title : support for the return of a metadata catalog

v. Definition : the server returns of all of the metadata in the file (both XML and binary), along with any association information (how particular pieces of metadata are associated with image data elements or other metadata)

vi. Details of feature requirement

1. Client implications : NA

2. Server implications : The server must be able to generate the catalog

3. Signalling implications : Return the catalog data

4. Other implications

Use Case #11

Source: Kodak

1. Title : upload of an image by parts

2. Context Information

2.a. Market application : consumer

2.b. Client platform characteristics : camera mainly, also applies for other connected devices

2.c. Server platform characteristics : indifferent, but probably can handle lot of concurent users

2.d. Network characteristics (bandwidth, delay, errors, architecture, QoS) : all

2.e. JPEG2000 image characteristics :

3. **Description of Use Case**

3.a. Introduction

A user takes a picture and choses to upload it from his wirelessly connected camera at low resolution/quality to a server to immediately share it with friends. Once home, he uploads the « rest » of the picture to the server. Optimally, he just wants to send the complementary data to the server. Therefore, the server must be able to describe what data it already has received during the first upload.

3.b. Description of the steps of operation

C→S : client sends UUID of the file

S→C : server either replies “new file” or sends the description of the data it already has on that file

C→S : client sends the image data

4. **Feature Requirements**

4.a. Title : support for partial upload by the client

iii. Definition : the client must be able to select the part of the codestream data that fits with the current available bandwidth and usage

ii. Details of feature requirement

1. Client implications : must be able to select right resolution / quality
2. Server implications : NA
3. Signalling implications : NA
4. Other implications

4.b. Title : support for file description by the server

vii. Definition : the server sends an « index » of the data it already has

viii. Details of feature requirement

1. Client implications : NA
2. Server implications : the server must be able to set up a description of the data it already has received
3. Signalling implications : sending of the description
4. Other implications

4.c. Title : sending of the complementary image data by the client

iii. Definition : the client gets the description of what the server already has on the picture and only sends the complementary data

iv. Details of feature requirement

1. Client implications : only sends relevant data
2. Server implications : « append » complementary data correctly
3. Signalling implications : NA
4. Other implications

Use Case #12

Source: Priam

1 Title : upload of an image by parts

2 Context Information

2.1 Market application : consumer

2.2 Client platform characteristics : camera mainly, also applies for other connected devices

2.3 Server platform characteristics : indifferent, but probably can handle lot of concurent users

2.4 Network characteristics (bandwidth, delay, errors, architecture, QoS) : all

2.5 JPEG2000 image characteristics :

3 Description of Use Case

3.1 **Introduction**

A user takes a picture and choses to upload it from his wirelessly connected camera at low resolution/quality to a server to immediately share it with friends. Once home, he uploads the « rest » of the picture to the server. Optimally, he just wants to send the complementary data to the server. Therefore, the server must be able to describe what data it already has received during the first upload.

3.2 Description of the steps of operation

C→S : client sends UUID of the file

S→C : server either replies “new file” or sends the description of the data it already has on that file

C→S : client sends the image data

4 Feature Requirements

4.1 Title : support for partial upload by the client

Definition : the client must be able to select the part of the codestream data that fits with the current available bandwidth and usage

ii. Details of feature requirement

4.1.1 Client implications : must be able to select right resolution / quality

4.1.2 Server implications : NA

4.1.3 Signalling implications : NA

4.1.4 Other implications

4.2 Title : support for file description by the server

4.2.1 Definition : the server sends an « index » of the data it already has

4.2.2 Details of feature requirement

4.2.3 Client implications : NA

4.2.4 Server implications : the server must be able to set up a description of the data it already has received

4.2.5 Signalling implications : sending of the description

4.2.6 Other implications

4.3 Title : sending of the complementary image data by the client

4.3.1 Definition : the client gets the description of what the server already has on the picture and only sends the complementary data

4.3.2 Details of feature requirement

4.3.3 1. Client implications : only sends relevant data

4.3.4 2. Server implications : « append » complementary data correctly

4.3.5 3. Signalling implications : NA

4.3.6 4. Other implications

Use Case #13

Source: Priam

1 Use Case Title

1.1 Navigation in Remote sensing images

2 Context Information

2.1 Market application - (include info about why JPIP needed)

Client need to access through the internet to earth observation images

The access is made distantly from the server location using a general internet connection with a wide range of clients types. In some cases, the access must be open to new external users. This requires a standard protocol enabling this type of users to access without having any specific equipment.

2.2 Client platform characteristics - (PDA, desktop, mobile phone) –

Wide range types of Desktop

2.3 Server platform characteristics - (PDA, desktop, large server, number of clients to serve)

large server

2.4 Network characteristics - (bandwidth, delay, errors, architecture, QoS)

variable bandwidth : from 56k to ADSL, high quality (up to lossless) images stored on the server

low delay

error prone environment in the case of mobile terminal

2.5 JPEG2000 image characteristics

Typical image size range from 128 x 128 to 6000 x 6000 pixels

Gray scale and color images

Image depth goes from 8 bits/pixel to 16 bits/pixel for gray scale images, and from 8 bits/pixel to 24 bits/pixel color images

raw J2K embedded into other proprietary file format

3 Description of Use Case

3.1 Introduction –

After his initial view the client requests additional resolution or quality of the image, or requests a different spatial region of interest.

3.2 Description of the steps of operation -

Step 1

After connection, the client selects a spatial region of interest on a map. Given that selection and the client capabilities, the server selects the required j2k images necessary for the construction of a new mosaic image.

Step 2

After construction of the mosaic, the client navigates in that new image. The navigation is quite simple (zoom, pan, ask for more quality, …)

4 Features requirements

4.1 Title : Ability to modify or to build a codestream on a server

4.1.1 Description –

In summary, the image is not yet created at the time of the selection by the client.

There more, it is important to keep some information on the server describing the link between the images and its neighbour images.

Details of feature requirement

4.1.1 Client implications –

4.1.2 Server implications –

4.1.3 Signalling implications –

Regardless where the construction of the image is performed, it may be necessary to signal the link between the images necessary for the mosaic.

4.2 Other implications -

4.2.1 Title – Independence of the file format.

Description –

JPIP must offer the possibility to deal with pure Jpeg2000 codestream, i.e. without using any Jpeg2000 optional file formats. It is important in remote sensing imagery applications where the file format is already very standardized. In this domain, it may be necessary to include raw jpeg2000 bitstreams into an existing proprietary file format, different from the optional JPEG2000 file formats..

Details of feature requirement

4.2.1 Client implications -
4.2.2 Server implications -
4.2.3 Signalling implications -
4.2.4 Other implications -

4.3 Title – Efficient caching mechanisms

Description –

The caching and buffering mechanism must be able to deal with several images, (It means that the server needs to keep information about the already sent data for each image. Another consequence is that during the first phase of the client/server dialog, the server has to specify the number of images of an exam, and the cache requirements for that set of images. The client has to describe his cache status for the set of images when requesting a first view of an exam.

Details of feature requirement

4.3.1 Client implications -
4.3.2 Server implications -
4.3.3 Signalling implications -
4.3.4 Other implications -

4.4 Title – Client capabilities description

Description –

User hardware and software (including decoder) capabilities must be described to the server when establishing the transfer. It should allow the parser to make intelligent choices about the relevant jpeg2000 data to send to the client. For instance, regarding the client displayer capabilities and the requested number of images, the parser can define a maximum level of resolution level.

Details of feature requirement

4.4.1 Client implications –

When contacting the server, the client must describe his software and hardware capabilities and also network bandwith and transmission conditions.

4.4.1 Server implications –

In the case of an intelligent server, the client capabilities have to be taken into account for the selection of the data to send.

4.4.1 Signalling implications –

n the negotiation phase, there is a need to include the capabilities description.

Other implications -

4.5 Title : Predictive protocol

Description –

The protocol must be able to deal with different users behaviour and changing channel conditions. It has also to adapt on demand to those conditions. An initialisation phase has to deal with these aspects and the protocol should include an updating procedure.

Details of feature requirement

4.5.1 Client implications -
4.5.2 Server implications -
4.5.3 Signalling implications -
4.5.4 Other implications -

Use Case #14

Source: Rosenbaum

1. **Browsing and display of large images**
2. **Context Information**
   2.1. Market application – mobile communication
   2.2. Client platform characteristics – Mobile phone, PDA
   2.3. Server platform characteristics – Laptop or Desktop
   2.4. Network characteristics – > 9600 bps
   2.5. JPEG2000 image characteristics – small (<1kx1k)
3. **Description of Use Case**
   3.1. Introduction – The still increasing market of mobile devices leads to always new and enhanced services for the user. Modern services for mobiles are starting to introduce more complex media types like image or video. With this trend new techniques considering these media types are developed. The description below describes a scenario which could be used for a Local Based Service offering new and up-to-date still image information to a mobile or PDA user. In our example it is a map of the suburb the user currently stands (but it could be every image information, where regions of the image content changing from time to time). The actual position is tracked by the client device and the area belonging to the position is displayed on screen. Additionally, the user wants to interact with the imagery by changing the view parameters (zoom, pan, crop) and the image information itself (parts of the content are only temporally valid) . I.e., if the user is interested in traffic jams in the surrounding, this information should be displayed by marking jammed streets on the original map at server side and sending these updated region(s) to the client (depending the actual view parameters etc.). This procedure avoids a lot of logic needed at client side to receive such requests via a second channel and render this information in an additional process to the screen.

   In our example the server knows the properties (cache size and strategy, display size and bitdepth, etc.) of the client. Therefore, he can adapt the whole procedure to the client, which allows a more effective handling in terms of used bandwidth and computing power.

   3.2. Description of the steps of operation –

   3.2.1. Log onto the image server (using the actual position, the actual interests(may transmitted via metadata) and client properties.

   3.2.2. The server selects the valid area (depending the position and the view params) from a very large map already encoded with JPEG2000(or created now, depending the clients props) and creates a new j2k-datastream containing only the required spacial area(i.e. via transcoding). The current interests of the user are (or have already been) imported in a copy of the whole map's j2k-datastream. Now the server blends the new and the "content-enhanced"-stream, be substituting areas of the new stream with "content-enhanced"-areas of the other stream using the special random access property of a J2k-datastream and transcoding.

   3.2.3. The server send this new datastream to the client (in an appropriate progression order)

   3.2.4. The client decodes and renders the received image to the screen (may be cyclical for a progressive refinement of the image information).

3.2.5. The user pans or moves to the left and requests new data for the missing area from the server (Caching: the client may or need to purge parts of the stream not needed for the current view.)

3.2.6. The server blends the image content for the requested region, depending the (at the moment not changing) additional users interests, in the same manner as above.

3.2.7. The server sends this information

3.2.8. The client includes the incoming data to the right place in cache.

3.2.9. The client renders the new part to the screen (may be cyclical for a progressive refinement of the image information).

3.2.10. The client changes (cancels and/or adds) his actual interests and signals this to the server.

3.2.11. The server sends UPDATES of the affected areas to the client. These UPDATES are areas from other streams containing the image information according to the actual interests of the user.

3.2.12. The client receives an UPDATE, completely deletes the belonging region in its cache and includes the Update-region properly.

3.2.13. The client renders and displays the (updated) image (may be cyclical for a progressive refinement of the image information).

3.2.14. Without a notice from the client the server may send UPDATES for regions which have to be updated to maintain the up-to-date state of the client (i.e. a new traffic jam). This scenario should be handled in the same manner as described in Step 11-13.

3.2.15. The user zooms into image and client sends the new Region of his Interest to Server.

3.2.16. The Server reacts by looking at the data have already been sent to the client and (if necessary) sends more resolution levels.

3.2.17. The user zooms out of image and client sends the new Region of his Interest to Server.

3.2.18. The Server reacts by looking at the data have already been sent to the client and sends the missing area/s (on former purges of the clients cache: once again).

3.2.19. The user crops the actual displayed image, to save screen space

3.2.20. Client react as if the user zooms out

3.2.21. Server recognizes that the requested region is smaller than the display size and reacts appropriately

4. **Feature Requirements** (for each feature / requirement specification)

(This section is based on the given requirements for Smart Dissemination Service (for very large

imagery) Part I/II, to ease finding commonnesses and differences)

4.1. Title – JPIP Client/Server logon

4.1.1. Description – This is the procedure between client and server client where the client tells the Server its “properties”, like display and cache size (may be the cache status, if re-logon at the same or nearly same position of the user). Additionally, the name of the J2k-File to browse (in our example this is implied by the application).

4.1.1.1. Client implications - Client initiates the connection – So the client must be able to formulate an appropriate request

4.1.1.2. Server implications - Must be able to interpret the message and formulate a response. Based on this response the client can derive if the server is able to serve the client or not. By means of the transmitted client-properties, the server should be so smart to derive as much as possible information for its further interaction with the client. This saves a lot of signalling overhead and reduces the transmission of redundant information.

4.1.1.3. Signalling implications – After the server accepts and responds the clients logon, it has to know all information about the client necessary for a non-redundant and effective handling. The transmitted client properties should be chosen in a manner the server knows "all" about the client that could affect further service processing.

4.2. Title – JPIP client request messages

4.2.1. Description – This is the message that a client sends to the server so that it can "know" what data to return - It is related to the user's current "view-port" into the image – View-port specifies x1, y1, x2, y2. All other data like resolution, quality, components etc. should be chosen by the server. During the logon process the server gains all information necessary to handle these values suitably.

4.2.2. Details of feature requirement

4.2.2.1. Client implications – Client sends the message – So the client must be able to formulate the request based on the user's view-port parameters

4.2.2.2. Server implications – Must be able to interpret the message and formulate the response

4.2.2.3. Signalling implications –

4.2.2.4. Other implications – an error resilience strategy should be used, to ensure the server has received the message. Otherwise, the client needs a lot of logic to oversee the cache and to repeat the message if necessary.

4.3. Title – Data transmission efficiency & Client-side JPIP cache

4.3.1. Description – The data transmission efficiency is the ratio of the JPEG2000 image data sent to a client that is required to complete a JPIP request to the total data sent (required JPEG2000 image data, excessive JPEG2000 image data, redundant JPEG2000 image data [i.e. data that has already been received by the client], plus signalling overheads) - The client is able to cache JPEG2000 image data sent by the server

4.3.2. Details of feature requirement

4.3.2.1. Client implications – Client must be able to cache the data. The client needs an intelligent cache which is closely connected to the chosen transmission protocol and the decompressor component to avoid signalling overhead as much as possible. The cache must act as input for the decoder which means the cache contains data already partially decoded (inverse-tier2, if the underlying transmission protocol supports the transmission of single code blocks or coding passes).

4.3.2.2. Server implications – For maximum efficiency, the server should be able to "know" what JPEG2000 image data is redundant – i.e. already held in the client's cache – and it should not resend this data

4.3.2.3. Signalling implications – Signalling overheads should be as small as possible

4.3.2.4. Other implications – The "intelligence" for performing the error-resilience management may be in either the client or the server – For instance, the client may acknowledge receipt of data "packets" (i.e. transport packets and not JPEG2000 image data packets), and thus the server can estimate what actions

to take (change mode, resend packets, etc) . Not only the amount of packets have been sent should be checked, but also the integrity of the packet itself(may optional, if executed by a used low level transport protocol).

4.4. Client cache management

4.4.1. Description – To optimise data transmission efficiency, the client holds a JPIP cache – the server should not send redundant data, i.e. data already in the client cache

4.4.2. Details of feature requirement

4.4.2.1. Client implications – Client must hold and manage its cache – including purging and book-keeping

4.4.2.2. Server implications – the server "knows" what is in the client cache. The client has to keep as simple as possible and take only care about the user input, the cache, the decoding and the display of the image

4.4.2.3. Signalling implications – May need cache management with regards to a reconnection and re-usage of data already in cache

4.4.2.4. Other implications

4.5. Title – Client response to user's requirements

4.5.1. Description – The user wants to view a view-port into the image on the display

4.5.2. Details of feature requirement

4.5.2.1. Client implications – Client does an cyclic or smart update by painting an image onto the display using the data held in its JPIP cache – The client sends an update to the server with a new JPIP request based on the new view-port

4.5.2.2. Server implications – Server responds promptly with JPEG2000 image data for this new view-port

4.5.2.3. Signalling implications – Defined JPIP client requests and defined JPIP replies comprising of JPEG2000 image data

4.5.2.4. Other implications –

4.6. Title – Session persistence

4.6.1. Description – The client can utilise data in its JPIP cache in subsequent sessions

4.6.2. Details of feature requirement

4.6.2.1. Client implications –The client has to inform the server exactly about its cache cache-contents (in a fully server steered handling the server needs and holds all information about the client) or if this to expensive he signals an empty cache. One possibility to inform the server about the cache content would be to signal the successful execution of a client request message which belongs to the cache content. Thus, the server can reconstruct the cache status of the client easily

4.6.2.2. Server implications –the server have to hold and update a model of the client's cache

4.6.2.3. Signalling implications – cache management signals are required

4.6.2.4. Other implications

Use Case #15

Source: Taubmann

1. **Title: Collaborative Image Browsing**
2. **Context Information**
   2.1. Market application – medical diagnosis, dissemination of educational materials, …
   2.2. Client platform characteristics – desktop or mobile PDA
   2.3. Server platform characteristics – nothing specific
   2.4. Network characteristics – > Lan, internet or RF link
   2.5. JPEG2000 image characteristics – Medium to large spatial dimensions, possibly with many components (volumetric imagery).
3. **Description of Use Case**

   Multiple clients are connected to a single server, interacting with the same image source. Image access or other requests made by one client have side effects for the other clients, in that they affect the prioritization of data delivery to those clients, as well as the interpretation of regions of interest to those clients. For example, a pathologists points at some portion of an image as relevant to diagnosis of a patient's condition. This causes the pathogist's image quality to improve progressively with emphasis in that region, while simultaneously causing the quality of the image to improve in the same region on other clients' displays. Each client's display will improve in a manner which is sensitive to that client's connection and possibly other interests of that client which may or may not be shared interests.

   In a related application, the dynamically changing prioritization of image data resulting from a pathologist's interaction with the image may have an impact on the order in which data is delivered to another client, interacting with the image at any later point in time.

   3.1. Description of the steps of operation –

   3.1.1. Clients connect asynchronously to the server, as the need arises.

   3.1.2. The client identifies its region of interest within the image, which it may update from time to time.

   3.1.3. The server customizes its response to the client based upon prioritization information, established by other clients, interacting with the same image either concurrently or previously.

   3.1.4. Clients may post information to the server identifying that areas of interest should be assigned higher priority in the delivery of data to both the present client and other clients collaborating with the image.

   3.1.5. Clients may disconnect asynchronously from the server.

4. **Feature Requirements** (for each feature / requirement specification)

   4.1. Title – Dynamic Data Delivery

   4.1.1. Description: The way in which a client's regions of interest are served may vary in a manner which depends upon the way in which other clients are interacting with the image or have previously interacted with the image.

   4.2. Title – Classification of Requests

   4.2.1. Description: Some client requests affect only delivery to that client; others affect the interpretation of the importance of different regions of the image, for all clients.

   4.3. Title – Anonymous Collaboration

   4.3.1. Description: Although clients collaborate, the server should manage this collaboration. Clients need not know about each other, or might not be allowed to communicate directly with each other for security reasons.

4.4. Title – Data transmission efficiency and responsiveness

4.4.1. Description – Clients in the collaboration who have low bandwidth connections need to have their image displays updated as rapidly and efficiently as possible to maintain responsive interaction with other clients in the collaboration.

General comment: I have tried very hard to make feature requirements independent of any possible way of addressing this use case. For this reason, it is not possible to give many details for feature requirements, and I suspect this will be a common problem for many use cases.