Digital Media Division

WMT Specification Document

Fast cache streaming

Version 1.1

Stability: Medium

Authors: Troy Batterberry

Core Team

Program Management (WMS)	Troy Batterberry
Development	Anders Klemets, Jim Stewart, Dawson Dean.
Test	Adam Trevor, Lan Ye
Marketing	Gary Schare
User Education	TBD

EXHIBIT 8
Althea L. Miller
CSR No. 3353
Date: 4/30/03
Witness: Batterberry

1 [illegible]

This document is MICROSOFT CONFIDENTIAL– appropriate care should be taken when distributing copies. The intended audience is members of the DMD development group. Others may read this document at their own risk.

2 [illegible]

This document details the requirements and implementation of the "Fast cache streaming" mode of operation.

[illegible]

CBR (Constant Bitrate): CBR is an acronym used to describe an encoding method for media content. The bitrate of a CBR file is generally fixed over time. Most existing streaming media servers (including v4.1) assume content to be encoded at a CBR.

Fast cache streaming: Next-generation streaming metaphor where the content is not necessarily transmitted by the server at a rate equal to the encoded bitrate of the file. In cases where excess bandwidth is available, the client "buffers ahead" while rendering. In cases where insufficient bandwidth exists, the client "buffers ahead" sufficiently before starting to render. Zeus-based clients utilize the disk as the buffering mechanism for Fast cache streaming. Future implementations may actually employ in-memory buffering.

VBR (Variable Bitrate): VBR is an acronym used to describe an encoding method for media content. The bitrate of a VBR file varies over time.

[illegible]

Conventional streaming media implementations (including the v4.1 server architecture) assume a reasonably consistent bandwidth connection between the client and the server. This metaphor generally streams content to the client at a continuous and constant bitrate. Minimal content is buffered on the client in order to minimize startup and time-shift latencies.

The conventional streaming model does not work well in popular broadband network topologies such as cable modems used to connect to the Internet. These topologies often do not provide consistent and reliable throughput. Instead, the throughput varies dramatically over time. For example, cable modems often encounter "brownouts" where little or no content are received for periods of time. As a result, glitches and rebuffering are common. Improvements are needed to handle the "variable" throughput nature of these topologies and the Internet in general.

The peak network throughput in broadband connections is often much greater than the encoded bitrate. This additional bandwidth should be employed to allow the client to buffer further ahead whenever possible in order to help prevent temporary reductions in the network throughput from affecting the end user experience. This behavior is similar to the progressive download scenario currently performed with standard web servers. The progressive download scenario is becoming increasingly popular with customers.

A hybrid server solution that provides the combined functionality of reliable progressive download capabilities along with the ability to meter delivery and parse relevant MBR streams would greatly

enhance the end-user experience of broadband customers for on-demand content. It would also provide a much more seamless solution for CDNs, ICPs, and other customers to use for delivering Windows Media content.

5.1 VBR Content Delivery

Under certain situations, VBR content offers distinct advantages over CBR content in terms of quality and disk space. In VBR content, additional bandwidth is temporarily used to handle scenes that are difficult to encode, and thus the overall quality of the scene is improved. Conversely, highly compressible scenes can result in a relatively low amount of bandwidth consumed. The overall quality achieved by a VBR encoding often exceeds the quality of a CBR encoding of the same content using the same number of total bytes.

VBR content poses significant problems to the conventional streaming metaphors employed by v4.1 and Hercules. The small buffers used by the client and the CBR delivery mechanism used by the server directly conflict with the requirements of VBR delivery.

Sony and others wish to use VBR content in "download and play" scenarios in order to provide adequate quality for movie rentals and purchases over the Internet. Using IIS for delivery is problematic because load between each individual client is not metered based on the detected bandwidth. Support for VBR content in WMS would provide a seamless solution to deliver both CBR and VBR content from one full-featured service.

5.2 "Glitch-free" playback on broadband network connections to the Internet

Jake has a cable modem that offers bandwidth performance that can vary dramatically over a period of time. Jake often experiences rebuffering on his client when accessing conventional streaming media servers. However, when Jake accesses media on web servers through the progressive download technique, he experiences little or no rebuffering. Furthermore, Jake likes the ability to have the file available for viewing after he downloads it. Jake wonders why streaming media servers cannot provide a better user experience.

5.3 Combination of Download Media and Streaming Media Support on a Single Server

A small ICP wants to sell movies based on a download and play scenario. However, the ICP offers preview clips using an MBR file that is streamed to the clients. The MBR file is used to allow the client to stream at the appropriate bitrate for the client network connection. A single server that can provide both download services and streaming services is required. A mechanism to handle load issues to ensure quality of service is also needed.

5.4 Ability to download individual streams in an MBR file

An ICP wants to offer both streaming and download services for a variety of movies on a variety of servers. The ICP only wants to encode the content once. The streaming clients will be connecting to the server over a variety of network topologies and speeds. Therefore, MBR files must be used to provide streaming support for the various connection speeds. The ICP does not want clients to download all the streams in the MBR file. Instead, the ICP wants to be able to explicitly specify the individual streams to be downloaded.

5.5 Flexible content delivery solution

A large CDN has an elaborate network of load balancing servers to deliver content for various customers. Since the customer requirements vary, the CDN must be able to support a variety of scenarios such as stream-based delivery and also progressive download delivery of WMC files. Currently, the CDN must explicitly assign some servers for providing download services and other servers for providing streaming services. The ability to dynamically change the aggregate ratio of download vs. streaming capacity is difficult.

There is considerable benefit to the CDN if the same computer used to stream media can also be used to provide conventional download support of the same WMC files. This could potentially reduce the number of servers (and disk space) needed by the CDN to host events. It could also be used to drive adoption of Windows in the CDN market, since Windows would be the only platform that could offer integrated support for both distribution scenarios. A unified administration metaphor is needed to make the support of both scenarios as seamless as possible.

6 REQUIREMENTS

6.1 Ability to "buffer ahead" in order to withstand brownouts/blackouts.

Fast cache streaming must be able to withstand interruptions (10+ seconds) in network connectivity with little or no impact to the client.

6.2 Content delivery at variable rates.

The Internet often provides non-deterministic and variable throughput performance over a period of time. Fast cache streaming must adapt to handle transient network conditions without impacting the user whenever possible.

6.3 Seek/Stride Support

Fast cache streaming shall offer seamless seek and stride support similar to conventional streaming.

6.4 Server-side control of Fast cache streaming.

Server administrators shall be able to turn Fast cache streaming support off for individual publishing points. They shall also be able to configure limits on the publishing point that control the amount of resources individual clients consume.

6.5 Client-side control of Fast cache streaming.

Users shall be able to disable Fast cache streaming support in the client UI.

6.6 Upper-limit of bandwidth for Fast cache streaming.

A client requests Fast cache streaming by requesting the duration of the acceleration to be –1 (i.e. the entire length of the file). In that request, the client also requests the rate to send the file. Since the client request results in bandwidth reserved on the server throughout the session, it is detrimental to allow clients to request artificially high values. The administrator requires a method to set the maximum per-client bandwidth available for Fast cache streaming on a publishing point. Note this server-side limit does not affect accelerated buffering requests. Therefore, a client can actually perform accelerated buffering in addition to Fast cache streaming.

6.7 Upper-limit of rate factor for Fast cache streaming.

The administrator requires a method to set the maximum per-client rate available for Fast cache streaming on a publishing point. Note this server-side limit does not affect accelerated buffering requests. Therefore, a client can actually perform accelerated buffering in addition to Fast cache streaming.

6.8 Explicit request for Fast cache streaming by client.

An explicit way to force Fast cache streaming is required for both standalone players as well as embedded players. This is needed so content authors can create links to content on a WMS server and force Fast cache streaming to occur for all clients.

6.9 Explicit request for a particular rate of delivery.

Content authors require the ability to author an explicit URI to request a specific rate of delivery.

6.10 Fast cache streaming at rates less than real-time

Some user scenarios will utilize Fast cache streaming even when the network bandwidth available is less than the encoded bit-rate of the file. The server and networking code must support this delivery mechanism. In such scenarios, Fast cache streaming must be invoked through an explicit request.

6.11 Download mode needed for player.

A mechanism is needed to "schedule" an unattended download to occur without actually rendering any of the data. For instance, users want the ability to start and finish a download without actually rendering any data.

[redacted]

7.1 On-demand content.

Fast cache streaming shall only be employed with on-demand content.

7.2 Available disk space.

Sufficient disk space must be available to cache the entire content clip in order for the client to request Fast cache streaming.

7.3 Additional network bandwidth detected.

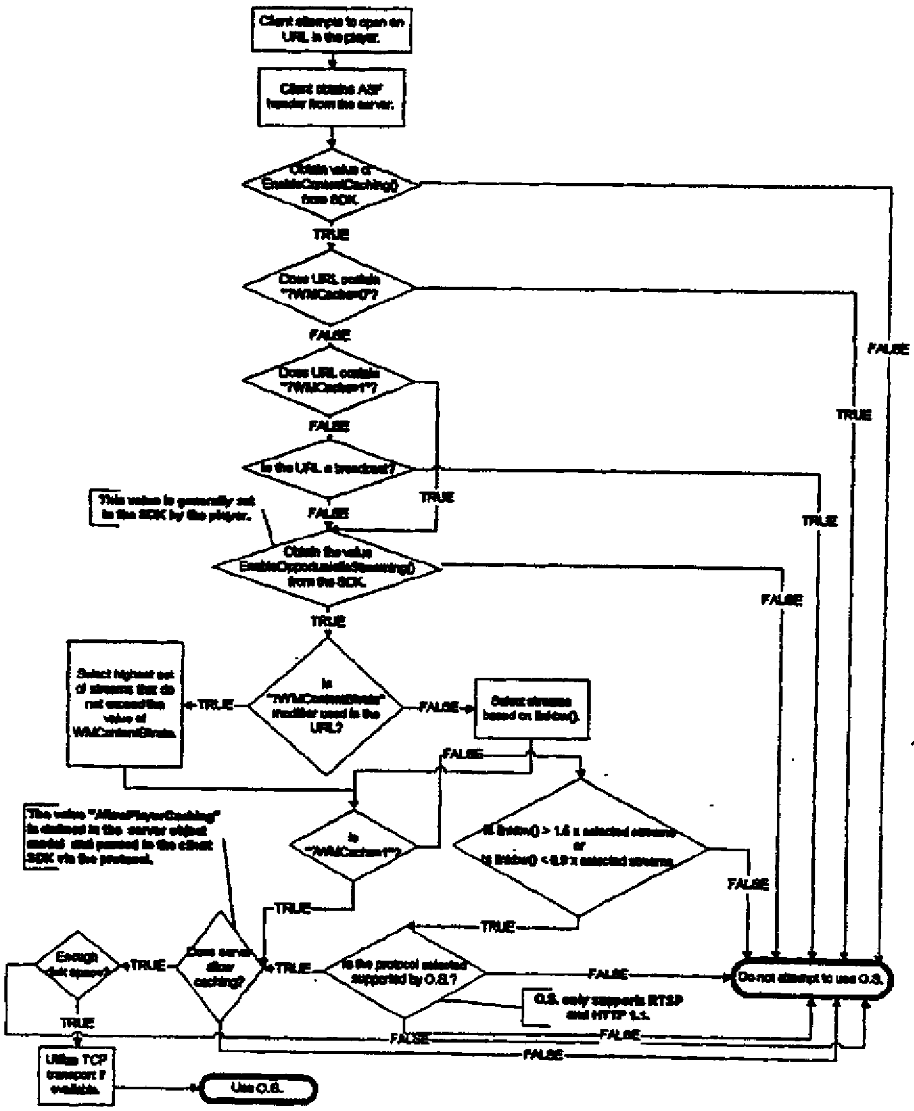
Fast cache streaming shall only be employed when the bandwidth detected exceeds the sum of the bitrate of the stream(s) selected by a factor of 1.5. (This factor may be refined pending the results of empirical testing).

7.4 AllowPlayerCaching property is TRUE.

If the AllowPlayerCaching property is set to FALSE, the server shall inform clients not employ Fast cache streaming. By default, AllowPlayerCaching shall be set to TRUE.

7.5 Flowchart describing negotiation of Fast cache streaming.

Negotiation of Opportunisitic Streaming in v9 SDK.



8 [illegible]

8.1 Playlists

8.1.1 Client-side playlists

Client-side playlists shall be processed and placed in cache in a rate constrained by the various client and server limits. If the client is able to secure sufficient client and server resources, it is possible that the client will cache many clips in advance of actually rendering them. This will help ensure truly seamless stream switching.

The use of Fast cache streaming shall not affect the end-user experience of an ASX file. For example, the element attribute "CLIENTSKIP" field shall be enforced even when the content is obtained from cache.

8.1.2 Server-side Playlists

Server-side playlist caching on the client shall be similar to the mechanism used to cache playlists on the server. The playlist will be treated as an atomic unit. An ASXv3 file shall be dynamically built by the client networking code to organize the individual clips for cached playback. Use of the SAVE AS function shall move both the ASX file and various WMC files over to the Media Library.

While the playlist entries are in cache, attributes such as "seekable" and "stridable" shall be honored. For example, if the server-side playlist includes advertisement clips marked as "noSkip", this behavior shall be enforced when the content is coming from cache. The resulting ASX file moved over during a "SAVE AS" operation shall include the "CLIENTSKIP" field as appropriate.

8.2 Authentication

Administrators that want to tightly control authentication should set the content cache expiration to zero. Clients will be able to play content from cache until it expires regardless of whether their access on the server has been revoked. To fully protect content and control licenses even after the content has been cached, DRM protection should be used.

8.3 Re-indexing of streams originating from an MBR file.

An MBR file must be re-indexed if the player wishes to save the file or seek/stride from cache. The client networking code shall re-index the streams.

8.4 Client Commands Affected by Fast cache streaming

8.4.1 Pause Command.

The player shall continue to download the content (via Fast cache streaming) into cache when the player is paused.

8.4.2 Seek/Stride Commands.

The following sections denote the behavior of the client during seek/stride commands.

8.4.2.1 Client cache behavior following a seek/stride command within cached data.

The client cache manager shall satisfy this request while the networking code continues streaming content into the cache. Therefore, a contiguous file will continue to be created.

8.4.2.2 Client cache behavior following a seek/stride command beyond cached data.

A client that seeks or strides beyond the data currently cached locally in buffers shall render the cached data dirty and discard it. The client will be unable to save the complete file in this scenario. Future implementations of the client may include "sparse file" support that is capable of reassembling file fragments resulting from such user actions.

8.4.3 Stop Command.

Caching will continue even after a STOP method is called if the client is within 30 seconds or 5% of the end of the file. Closing the file shall stop any further caching.

8.5 Size of client buffer.

The client networking code shall buffer the entire content segment on disk assuming sufficient disk space is present and the no-caching bit is not set. If sufficient disk space is not present or the no-caching bit is set, Fast cache streaming shall not take place.

Future versions of the player/SDK may provide incremental streaming metaphors. For example, a hybrid metaphor of caching content using a large in-memory buffer may be used instead of a disk.

8.6 Stream thinning/Intelligent streaming.

A client operating in the "Fast cache streaming" mode shall not dynamically request stream thinning. Intelligent streaming shall effectively be disabled.

8.7 Negotiated Fast cache streaming protocols.

In order to simplify the implementation, only RTSP (TCP data delivery) and HTTP shall be negotiated as Fast cache streaming protocols.

8.8 Protocol Rollover Logic.

Fast cache streaming shall never change the "FirstProtocol" setting used by the protocol rollover functionality. Consult the protocol rollover specification for more details on protocol "stickiness".

8.9 Logging.

8.9.1 Log entry following a population of the client cache.

A log entry shall be submitted by the client cache code immediately after completion of a particular network request. For example, if the end of the file is reached, the client network code shall submit a log entry indicating the data received. Note this log entry shall lack information regarding rendering. A successful entry will have a status of 200.

Since the log entry shall be sent immediately after receipt of the data, the server shall always have corresponding server-side statistics. Therefore, aside from the lack of rendering data, the entry shall be complete.

8.9.2 Log entry following a rendering state change in the client.

The client shall send a log event each time the rendering state changes. This is consistent with the conventional logging metaphor used in the v4.1 server/client. The client shall maintain a connection to the server even when the content is played from cache. This connection shall keep a client context present on the server and allow for the submission of log entries. Note such log entries will primarily contain client information similar with "remote" log entries used in caching appliance scenarios. A successful log entry shall have a status code of 204, indicating that no content was actually delivered from the server. Consult the logging specification for details on fields that are valid for remote log events.

8.9.3 Logging when the client is offline.

To simplify the design, the client shall not attempt to save log information generated from playing cached content. Therefore, if the client is offline, the log data shall be lost.

8.9.4 Logging when playing data locally from the Media Library.

Logging shall not occur once the data is moved from the local disk cache. If the content is placed in the Media Library or on another area of the client disk, no logging shall occur.

8.10 "Slow skip" mode in Fast cache streaming

If disk space is exhausted while streaming a specific element using the Fast cache streaming metaphor, an error shall be returned. However, the client networking code shall re-evaluate the use of Fast cache streaming at the start of each playlist element. This could result in some entries being Progressiveally streamed and others conventionally streamed.

8.11 Incomplete cached content.

Fragments of media elements caused by a variety of situations including the user seeking beyond a cached region, network outages, closing the player prematurely, etc shall be discarded immediately in order to free up disk space.

8.12 Freeing server b/w resources

The server shall release the bandwidth reserved by a client once an End of File (EOF) is reached. If the client attempts to stream the content again, it must re-reserve the appropriate bandwidth.

8.13 WMP UI shall display the white bar indicating the current amount of the file present in cache in a "slower than realtime" scenario.

The Rosetta player shall display the white bar indicating buffering progress in a slower than real-time streaming scenario. This will provide visual feedback to the user that buffering progress is occurring. Note that in normal streaming modes as well as Fast cache streaming at rates equal or greater than the encoded bitrate of the file, a white bar shall not be displayed.

9 [redacted]

9.1 Player

9.1.1 Player UI

9.1.1.1 Enable/disable Fast cache streaming.

The player UI shall offer a checkbox entitled "Allow caching of streamed content to disk" that is capable of enabling/disabling Fast cache streaming/caching. This single checkbox shall be located in the "Performance" tab of the Options dialog in the Rosetta Media Player. RAID bug 42408 is tracking this issue.

9.2 WM Format SDK

9.2.1 Interface properties related to Fast cache streaming

9.2.1.1 Enable/disable Fast cache streaming.

The following two properties shall be supported in the SDK to control Fast cache streaming:

HRESULT IWMReaderNetworkConfig2.GetEnableProgressiveStreaming([out] BOOL *pfEnableOppStreaming)

HRESULT IWMReaderNetworkConfig2.SetEnableProgressiveStreaming([in] BOOL fEnableOppStreaming)

9.2.1.2 Enable/disable content caching.

HRESULT GetEnableContentCaching([out] BOOL *pfEnableContentCaching);
HRESULT SetEnableContentCaching([in] BOOL fEnableContentCaching);

9.2.1.3 SAVE Content methods/properties.

Attempting to save the content can lead to numerous end-case conditions and problems. For instance, the client may have insufficient disk space. The user could seek beyond the cached data and subsequently flush previously cached (but desired) data. The method could be called after several playlist elements have already been streamed and potentially removed from cache. It is the responsibility of the end-user application to manage the client state and warn the user of problems. It is also important for the application to prohibit actions that could result in undesirable results.

9.2.1.3.1 "Save File As" Method.

Save the current file/playlist. This operation is asynchronous. WMT_SAVEAS_STOP indicates that the save has completed. Closing the reader will abort a save operation that has not completed.

HRESULT SaveFileAs([in] const WCHAR *pwszFilename);

9.2.1.3.2 Save Progress Property

When saving a file or playlist, the operation may take awhile. Between WMT_SAVEAS_START and WMT_SAVEAS_STOP, this call will return progress values. pdwPercent returns the percentage of the save as that has completed.

HRESULT GetSaveAsProgress([out] DWORD *pdwPercent);

9.2.2 URI Query values used to control Fast cache streaming

The content author can use specific URI query strings to explicitly control Fast cache streaming. This section describes the various strings and their behavior. For general information on the URI syntax, see RFC 2396.

Note individual modifiers such as WMCache and WMBitrate can be combined in a single URI string by using the ampersand ("&") operator. For instance, the following URL is valid:

mms://server/file.asf? WMCache =1& WMBitrate=56000

9.2.2.1 Explicitly enable/disable Fast cache streaming

The modifier "WMCache=" can explicitly enable or disable Fast cache streaming in the client by acting as a Boolean value. For example, the following URI will explicitly use Fast cache streaming:

mms://server/file.asf?WMCache=1

9.2.2.2 Explicitly set the link b/w used for the URL

The modifier "WMBitrate=" is used to explicitly set the link b/w (in bits/s) available to stream the URL. This modifier was added to the v7 player and is not specific to Fast cache streaming:

mms://server/file.asf?WMBitrate =56000

9.2.2.3 Explicitly set the maximum total encoded bitrate streams to be selected.

The modifier "WMContentBitrate =" is used to set the maximum total encoded bitrate of the streams selected. For example, the request below shall cause the SDK to select the highest encoded bitrate stream(s) that do not exceed 56,000 bits/s.

mms://server/file.asf?WMContentBitrate =56000

9.2.3 Exported functions from wmvcore.dll

9.2.3.1 SDK API needed to check for the presence of a file in cache.

The player needs to efficiently determine whether a file is available in cache for offline playback, etc. The player will have to call this API frequently, and therefore requiring an instantiation of the reader is not practical. Instead, WMVCORE.DLL shall export a function to check for the presence of the file in cache. This exported function is called "WMIsAvailableOffline".

9.3 Server/Administration

9.3.1 Adjustable Settings in the Server Administration UI.

9.3.1.1 Enable/disable Fast cache streaming.

This setting shall be configurable on each publishing point. This setting shall control whether Fast cache streaming is available to clients.

9.3.1.2 Set the maximum per-client bandwidth used by Fast cache streaming.

This setting applies to all types of streaming clients. It is configurable for each publishing point and also for the server.

9.3.1.3 Set the maximum delivery rate (i.e. "speedup" factor) used by Fast cache streaming.

This setting shall be configurable on each publishing point.

9.3.2 Properties in the Server Object Model for Fast cache streaming.

9.3.2.1 Enable/disable Fast cache streaming

IWMSPublishingPoint.AllowPlayerSideDiskCaching

HRESULT AllowPlayerSideDiskCaching ([in] VARIANT_BOOL newVal);
HRESULT AllowPlayerSideDiskCaching ([out, retval] VARIANT_BOOL *pVal);

9.3.2.2 Setting the maximum per-client bandwidth

9.3.2.2.1 IWMSServerLimits.PerPlayerConnectionBandwidth

This property already exists in the Object Model. However, the meaning of the value is being extended to include Fast cache streaming clients.

9.3.2.2.2 IWMSPublishingPointLimits.PerPlayerConnectionBandwidth

This property already exists in the Object Model. However, the meaning of the value is being extended to include Fast cache streaming clients.

9.3.2.3 Setting the maximum delivery rate for Fast cache streaming clients

IWMSPublishingPointLimits.PlayerCacheDeliveryRate

This property shall allow the administrator to set the maximum rate an Fast cache streaming client may request a file. The default value shall be 5.0.

10 POSTPONED REQUIREMENTS / LIMITATIONS

10.1 Perfmon Counters

10.1.1 Existing Publishing Point Perfmon Counters Affected By Fast cache streaming

10.1.1.1 Current Connected Players

10.1.1.2 Current File Read Rate

10.1.1.3 Current Late Read Rate

10.1.1.4 Current Player Allocated Bandwidth

10.1.1.5 Current Stream Error Rate

10.1.1.6 Peak Connected Players

10.1.1.7 Peak Player Allocated Bandwidth

10.1.1.8 Total Connected Players

10.1.1.9 Total File Bytes Read

10.1.1.10 Total Late Reads

10.1.1.11 Total Player Bytes Sent

10.1.1.12 Total Stream Denials

10.1.1.13 Total Stream Errors

10.1.1.14 Total Stream Terminations

10.1.2 New Publishing Point Perfmon Counters Required for Fast cache streaming

10.1.2.1 Current Fast cache streaming HTTP Players

10.1.2.2 Current Fast cache streaming RTSP Players

10.1.2.3 Current Fast cache streaming Players

10.1.2.4 Peak Fast cache streaming Players

10.1.2.5 Total Fast cache streaming Players

10.2 Explicit request for a particular set of stream(s).

Postponed.

10.3 Assembly of file fragments resulting from seek or stride requests.

In order to reduce the complexity of the client code, seek or stride requests beyond the currently buffered content shall result in the existing buffers being flushed. Instead, new buffering will begin at the new seek position.

Care should be taken in the design to allow the client to build a "sparse file" cache management function without requiring new changes to the final version of the Hercules server.

10.4 Dynamic load adaptation between Fast cache streaming clients.

In order to simplify the implementation in the server, an Fast cache streaming client will reserve a consistent level of server resources throughout a specific streaming session. For instance, if a client requests to Progressiveally stream a file at 1 mbit/s but is only able to stream the file at 300 kbits/s, the client will still consume 1 mbit/s worth of server resources. Dynamic load adaptation will not occur in this release.

10.5 Protection of cached data

Ideally, the data written to the IE cache should be protected with some type of encryption such as DRM. This platform release will not provide this capability. Instead, it is up to the content providers to encrypt the content prior to delivery.

It is also possible to simply disable client-side disk caching using the publishing point property AllowPlayerSideDiskCaching. However, this provides no protection against rogue clients attempting to illegally copy data.

10.6 Dynamic transition from Fast cache streaming.

In order to simplify the design for Eclipse, clients will not automatically stop using Fast cache streaming and transition to conventional streaming while within a media element. It is assumed that the b/w measurement logic will be accurate in a majority of client scenarios. In the cases where the b/w measurement is incorrect, the client will have to manually disable Fast cache streaming in the client UI or manually set their available link b/w.

10.7 "Slow skip" mode in Fast cache streaming

The following feature was postponed. If disk space is exhausted while streaming a specific element using the Fast cache streaming metaphor, an error shall be returned. However, the client networking code shall re-evaluate the use of Fast cache streaming at the start of each playlist element. This could result in some entries being Progressiveally streamed and others conventionally streamed.

If the client exhausts all available disk caching space during an Fast cache streaming session, the client code needs to gracefully recover. First, the client cache will be flushed to create additional free disk space. If this is still insufficient, the client shall transition to the conventional streaming metaphor. If the no-cache bit is set or Fast cache streaming is disabled on the publishing point, the explicit request shall fail with an appropriate error.

10.8 Download mode needed for player.

A download manager will not be available in the Rosetta player. However, ISVs can essentially write a download manager by attempting to stream the file and simply pause it.

Original requirement: A mechanism is needed to programmatically "schedule" an unattended download to occur without actually rendering the data. In other words, a "pre-stuff" API is needed to populate the client cache.

10.9 Live content.

End-to-end live stream support for Fast cache streaming is beyond the scope of the Eclipse release. However, the SDK shall not preclude users from developing a live caching solution.

10.10 Secure logging.

The concept of true secure logging will not be supported in Eclipse. Instead, the server will only accept log entries from clients with a matching client ID on the server. This means Fast cache streaming clients will have to maintain a connection to the server even when rendering from cache in order to successfully log.

Original requirement: When the client is streaming solely from cache (and no further download is taking place), it does not have a TCP connection to the server. Furthermore, the server does not have any remaining knowledge of the client. In such cases, the server must be able to ensure the log being submitted by the client is valid. A TBD secure logging mechanism shall be used to ensure log data security.

10.11 Add field to client UI indicating the use of Fast cache streaming.

Postponed due to scheduling issues with Rosetta.

[redacted]

11.1 Impact to overall server scalability due to Fast cache streaming.

Progressive on-demand streaming implies the use of the TCP protocol for a data transport. In previous versions of the server, TCP has provided considerably less performance on large-scale systems. In some cases, reductions of nearly 50% have been observed when switching from the UDP data transport to the TCP data transport with the v4.1 server. The reduction in performance using the Hercules architecture needs to be better understood.

[redacted]

12.1 Freshness checking

Freshness checking in server-side playlists requires obtaining the header from individual clips and comparing them with cached content. The client networking code shall obtain headers for seekable elements and skip forward to the next element when possible. If the content is not seekable, the client will be forced to stream the content in order to determine freshness of the next element.

12.2 Minimum buffer size calculations for VBR files.

The client code needs to approximate the minimum buffer size needed before rendering can begin when a VBR file is Progressiveally streamed. This issue is being tracked as NMPU_WMT bug 42860.

12.3 Optimization of client-side ASX processing

Prerolling subsequent entries in a client-side ASX needs to be optimized to fully utilize the benefits of Fast cache streaming. The player currently attempts to preroll 5 s worth of data following a WMT_END_OF_STREAMING notification. This code should be changed to allow the client to preroll the entire clip via Fast cache streaming if the mode is available.

12.4 Add no-skip field to the content description.

Fast cache streaming currently does know if content is explicitly flagged as "no-skip" for the purpose of ensuring an ad is viewed.

[redacted]

http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html