HTTP Dynamic Streaming Specification Errata: May 2014

HTTP Dynamic Streaming Specification Errata: May 2014

HTTP Dynamic Streaming Specification Errata: May 2014

Published May 2014.

Abstract

This document corrects errors and omissions in the "HTTP Dynamic Streaming Specification Version 3.0" document as published in August 2013.

1. Add "Additional Bootstrap Requirements" section

Add the following section after Section 8.4 of the specification:

> ### *8.4.1 Additional Bootstrap Requirements*
>
> *This section describes additional requirements regarding HDS bootstraps. The requirements described in this section supercede any specified in [F4MSPEC] and apply to all HDS versions.*

Array indices in this section are specified as 0-based indices.

8.4.1.1 Manifest Bootstraps

A "manifest bootstrap" is bootstrap information that is referenced by or embedded within a <bootstrapInfo> element of the manifest.

Each rendition is associated with a manifest bootstrap; The @bootstrapInfoId attribute SHALL be present on all <media> elements.

8.4.1.2 Fragment Bootstraps

A "fragment bootstrap" is the bootstrap information that is embedded within a fragment.

A fragment MAY contain a bootstrap, but the meaning of such a bootstrap is currently undefined. The client SHALL ignore all fragment bootstraps.

8.4.1.3 'abst' box

This section provides additional requirements on the 'abst' box of manifest bootstraps.

Profile, Live, MovieIdentifier, ServerEntryCount, QualityEntryCount, DrmData, and MetaData are deprecated and SHALL be 0. ServerEntryTable and QualityEntryTable are also deprecated and SHALL be empty. The client SHALL ignore these fields since equivalent information is available within the F4M manifest document.

CurrentMediaTime SHALL equal the presentation time of the most recent message in the last fragment of the bootstrap. See Section 8.4.7 for a precise definition of the last advertised fragment.

Update SHALL be 0.

The bootstrap shall contain exactly 1 fragment run table and exactly 1 segment run table; SegmentRunTableCount and FragmentRunTableCount SHALL both be 1.

8.4.1.4 'asrt' box

This section provides additional requirements on the 'asrt' box of manifest bootstraps.

Flags SHALL be 0.

QualityEntryCount is deprecated and SHALL be 0. Therefore, the 'abst' box SHALL contain exactly 1 'asrt' box and QualitySegmentUrlModifiers SHALL be empty. The quality related fields of the bootstrap are deprecated since the @bootstrapInfoId attribute of the manifest associates each rendition with a single bootstrap.

8.4.1.5 'afrt' box

This section provides additional requirements on the 'afrt' box of manifest bootstraps.

Flags SHALL be 0.

TimeScale SHALL equal the enclosing 'abst' box's TimeScale.

QualityEntryCount SHOULD be 0. Therefore, the 'abst' box SHALL contain exactly 1 'afrt' box and QualitySegmentUrlModifiers SHALL be empty. As described in Section 8.4.1.4, the quality related fields of the bootstrap are deprecated.

The first FRAGMENTRUNENTRY cannot be a discontinuity entry; The value of FragmentRunEntryTable[0].FragmentDuration SHALL NOT be 0.

The last FRAGMENTRUNENTRY can only be a normal entry or an end of presentation discontinuity; If the FragmentDuration of the last FRAGMENTRUNENTRY of the FragmentRunEntryTable is 0, its DiscontinuityIndicator SHALL be 0.

Unless it is the final end of presentation discontinuity, a Discontinuity FRAGMENTRUNENTRY can only be followed by a normal entry; If the FragmentDuration of a FRAGMENTRUNENTRY is 0 and it is not the last FRAGMENTRUNENTRY of the FragmentRunEntryTable , the FragmentDuration of the next FRAGMENTRUNENTRY SHALL be non-zero.

The end of presentation discontinuity can only appear at the end of the segment run table; If the FragmentDuration of a FRAGMENTRUNENTRY is 0 and its DiscontinuityIndicator is 0, the FRAGMENTRUNENTRY SHALL be the last

FRAGMENTRUNENTRY in the FragmentRunEntryTable.

FRAGMENTRUNENTRY elements shall be in increasing timestamp order; That is:

- *If a FRAGMENTRUNENTRY e1 has a non-zero FragmentDuration and is followed by a FRAGMENTRUNENTRY e2 with non-zero FragmentDuration, e2.FirstFragmentTimestamp SHALL be greater than e1.FirstFragmentTimestamp.*
- *If a FRAGMENTRUNENTRY e2 has a 0 FragmentDuration and a DiscontinutityIndicator of 2 or 3, the preceding FRAGMENTRUNENTRY e1 and following FRAGMENTRUNENTRY e3 SHALL have values such that e1.FirstFragmentTimestamp < e2.FirstFragmentTimestamp and e2.FirstFragmentTimestamp < e3.FirstFragmentTimestamp.*
- *If a FRAGMENTRUNENTRY e2 has a 0 FragmentDuration and a DiscontinutityIndicator of 0 or 1, the FirstFragmentTimestamp of the FRAGMENTRUNENTRY SHOULD be ignored.*

8.4.1.6 Mapping presentation times to fragment ids

This section describes how the contents of the bootstrap determine which fragments are advertised.

The id of the first fragment advertised by the bootstrap is determined by the following formula:

```
FirstFragmentId = afrt.FragmentRunEntryTable[0].FirstFragment
```

The id of the last fragment advertised by the bootstrap is determined by the following formula:

```
LastFragmentId = SRE_LastFragmentId(abst.SegmentRunEntryCount - 1)
```

See Section 8.4.7 for a definition of SRE_LastFragmentId. The current media time SHALL NOT be used to determine the last advertised fragment id.

Each non-discontinuity FRAGMENTRUNENTRY maps a contiguous range of presentation times to a contiguous range of fragment ids. The start of the time range for FRAGMENTRUNENTRY[i] (in TimeScale units) determined by the following formula:

```
FRE_StartTime(i) = afrt.FragmentRunEntryTable[i].FirstFragmentTimestamp
```

The end of the time range for FRAGMENTRUNENTRY[i] (in TimeScale units) equals the return value of FRE_EndTime(i) in following pseudocode:

```
    function FRE_EndTime(i)
        if i == afrt.FragmentRunEntryCount - 1
            return LastFragmentEndTime()
        else if FragmentDuration == 0
            return FRE_StartTime(i)
        else if afrt.FragmentRunEntryTable[i+1].FragmentDuration == 0
            let nextDiscontinuityIndicator =
afrt.FragmentRunEntryTable[i+1].DiscontinuityIndicator
            if nextDiscontinuityIndicator == 0
                  return LastFragmentEndTime()
            else if nextDiscontinuityIndicator == 1
                return afrt.FragmentRunEntryTable[i+2].FirstFragmentTimestamp
            else // nextDiscontinuityIndicator == 2 or nextDiscontinuityIndicator == 3
                return afrt.FragmentRunEntryTable[i+1].FirstFragmentTimestamp
            end if
        else
            return afrt.FragmentRunEntryTable[i+1].FirstFragmentTimestamp
        end if
    end function

    function LastFragmentEndTime()
        var frtEntry
        if afrt.FragmentRunEntryTable[afrt.FragmentRunEntryCount - 1].FragmentDuration != 0
            set frtEntry = afrt.FragmentRunEntryTable[afrt.FragmentRunEntryCount - 1]
        else
            set frtEntry = afrt.FragmentRunEntryTable[afrt.FragmentRunEntryCount - 2]
        end if
        return frtEntry.FirstFragmentTimestamp + (LastFragmentId - frtEntry.FirstFragment + 1)
* fre.FragmentDuration
    end function
```

The time range for a given FRAGMENTRUNENTRY includes FRE_StartTime(i) and all times up to, but not including, FRE_EndTime(i).

Therefore, the fragment id corresponding to a given presentation time (in TimeScale units) equals the return value of the following pseudocode:

```
    function fragmentId(t)
       let i = 0
       while i < afrt.FragmentRunEntryCount
           if FRE_StartTime(i) <= t and t < FRE_EndTime(i)
               let frtEntry = afrt.FragmentRunEntryTable[i]
               return frtEntry.FirstFragment + floor((t - FRE_StartTime(i)) /
frtEntry.FragmentDuration)
           end if
           set i = i + 1
       end while
       return undefined // time does not map to any fragment
    end function
```

8.4.1.7 Mapping fragment ids to segment ids

Each SEGMENTRUNENTRY maps a contiguous range of fragment ids to a contiguous range of segment ids. The first fragment id mapped by a SEGMENTRUNENTRY at index i of asrt box's SegmentRunEntryTable is equal to the return value of the following pseudocode:

```
function SRE_FirstFragmentId(i)
    let afrt = abst.FragmentRunTableEntries[0]
    let asrt = abst.SegmentRunTablesEntries[0]
    if i == 0
        return afrt.FragmentRunEntryTable[0].FirstFragment
    else
        return (SRE_FirstFragmentId(i-1) +
                asrt.SegmentRunEntryTable[i-1].FragmentsPerSegment *
                (asrt.SegmentRunEntryTable[i] - asrt.SegmentRunEntryTable[i-1]))
    end if
end function
```

Notice that the first fragment id mapped by the segment run entry table is the first fragment id that appears in the bootstrap.

The last fragment id mapped by a SEGMENTRUNENTRY at index i of asrt box's SegmentRunEntryTable is equal to the return value of the following pseudocode:

```
function SRE_LastFragmentId(i)
    if i == asrt.SegmentRunEntryCount - 1
        return (SRE_FirstFragmentId(i) +
                abst.SegmentRunTableEntries[i].FragmentsPerSegment - 1)
    else
        return SRE_FirstFragmentId(i+1) - SRE_FirstFragmentId(i)
end function
```

Notice that the last SEGMENTRUNENTRY only provides the segment ids for a single fragment.

The first and last fragment ids advertised by the segment run table SHALL NOT refer to discontinuities. This implies that the fragment id specified by an end of presentation discontinuity's FRAGMENTRUNENTRY SHALL NOT be in the range of any SEGMENTRUNENTRY.

Therefore, the segment id for a given fragment id equals the return value of the following pseudocode:

```
function segmentId(f)
    var i = 0
    while i < asrt.SegmentRunEntryCount - 1
        if f >= SRE_FirstFragmentId(i) and f <= SRE_LastFragmentId(i)
            let srtEntry = abst.SegmentRunTableEntries[i]
            return SRE_LastFragmentId(srtEntry.FirstSegment + floor((f -
SRE_FirstFragmentId(i)) / srtEntry.FragmentsPerSegment)
        end if
        i = i + 1
    end while
    return undefined // fragment is not advertised
end function
```