电 信 终 端 产 业 协 会 标 准

TAF-WG2-AS0020-V1.0.0:2018

移动智能终端应用软件网络资源使用效率测评方法

Technical Requirement and Test Specification for Network Resource Usage Efficiency of Smart Mobile Application

2018－09－04 发布　　2018－09－04 实施

电 信 终 端 产 业 协 会　　发 布

目　　次

前　言

本标准按照 GB/T 1.1-2009给出的规则编写。
本标准由电信终端产业协会提出并归口。
本标准起草单位：中国信息通信研究院、中国移动通信有限公司研究院
本标准主要起草人：曾晨曦、高立发、陈燕燕

移动智能终端应用软件网络资源使用效率测评方法

1 范围

本标准规定了移动智能终端应用软件网络资源使用效率的评测方法，包括技术要求、测试方法、以及评价方法。

本标准适用于所有移动智能终端应用软件。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

Hypertext Transfer Protocol -- HTTP/1.1

3 术语、定义和缩略语

3.1 术语和定义

3.1.1 缓存控制信息 Cache Control Header

http 传输报文中，用于标记缓存机制相关信息的头部文件。

3.1.2 传输能耗 Transmission energy consumption

终端因网络连接而消耗的电量资源，单位为 J。

3.2 缩略语

App	应用软件	Application
CSS	层叠式样式表	Cascading Style Sheets
JS	Java Script	Java Script

4 技术要求

4.1 传输内容

4.1.1 文本压缩

对文件进行压缩可缩减其大小，加快传送速度。大部分情况下，在终端上对文件进行解压消耗的头文件比 PC 端相对要少。因此，我们推荐对于文本类资源，应在服务器端先进行压缩处理，后执行下载动作。

App 在网络传输过程中发送超过的未压缩文本类文件应不超过所有文本文件的 5%。

4.1.2 重复内容

当客户端向服务器请求下载曾经使用过的重复内容时，会占用网络带宽、消耗终端的流量、增加应用响应时间，产生不必要的资源浪费。因此建议在应用中建立缓存机制，为属性为可缓存的文件，建立可临时存储的缓存区域。

因此，在 App 在网络传输过程中，不应出现未过期的重复下载内容。在时长不低于 5min 的遍历过程中，检测到的重复下载的内容应不超过 3 项。

4.1.3 缓存使用

为保证缓存机制的正常使用，服务器端应为下载资源设立缓存控制信息，以对其有效性进行标志。具体缓存控制信息的设立机制可参考 HTTP/1.1Protocol Specification (RFC2616)协议,第 13 章 (Section 13 : titled Caching in HTTP)

因此，App 在报文交互过程中，应建立且使用缓存控制信息。

4.1.4 过期缓存

信息有效性的标定，可采用缓存头文件记录方式。根据 Http1.1 协议，目前使用较多的为核对“有效性”和标记“过期时间”两种方式。

当客户需要下载某文件时，应用应首先在缓存中搜索，如发现匹配内容，则进一步检查其有效性，都符合要求即可直接使用，避免重复下载。若该内容已过期，则需向服务器发送请求，核查资源是否有变，若无变化服务器直接返回 304（无修改响应代码），应用接收便可直接从缓存中读取所需内容，节省了重新下载全部内容所消耗的资源。

app 应按照上述逻辑，正确处理过期缓存信息。对于服务器返回 304 代码，却依然请求完整资源重新下载的情况，应给出警告。

4.1.5 合并请求

当同一时间段内，出现多个对同类型资源的请求时，不但会减缓网络的下载速度，其头文件也会造成不必要的流量消耗。建议在服务器端，对同时段的同类型资源的请求进行合并。例如，一个应用有多个需下载的外部 CSS 和 JS 文件，会导致其 RTT(Round Trip Transfers)增多，造成资源浪费；而将多个 CSS 和 JS 文件分别进行合并，资源的使用效率将大大提高。

另外，使用 CSS Sprites 技术将多个小图片组合后再下载，也可达到同样的效果。

在 App 在网络传输过程中，2s 内应检测到不高于 2 个单独的 CSS 或 JS 请求。

4.1.6 图片处理

终端种类繁多，屏幕大小也从手机到平板，差异巨大。因此，使用同样大小的图片适配所有终端也会造成不必要的资源浪费。建议在对图片资源进行下载前，应根据其在屏幕上的显示尺寸适当调整其大小。

目前广泛使用的方法主要有以下三种：人工适配，CSS Media Query，内容管理系统(Content Management Systems)。三种方法各有其优缺点，在此就不深入讨论，开发者可根据应用自身业务特点，选取适合的解决办法。

在 App 在网络传输过程中，检查所有图片规格，所有下载图片的规格都应在填充区域大小的 150%以内。

4.1.7 信息精简

为了方便开发者编写和阅读，代码中常夹杂如空格、换行符等格式化信息；实际上，这类信息对于代码的执行并无意义。信息精简即在应用代码中去除所有非必要的字符，如空格、评论、分隔符等。

在 App 在网络传输过程中，检查非用户阅读类的文本信息，可精简的信息应不超过总量的 5%。

4.1.8 图片组合

建立快速网络连接的最佳方式为减少 http 请求数量，因此将多个小图文件打包为一张大图，可减少网络请求数量，甚至节省一定流量。

因此，App 网络交互过程中不应有大量，且可组合的图片流频繁出现。具体出现频率可根据业务场景，以及开发者对 App 的优化期望度定制化规定，在此不进行量化要求。

4.2 传输过程

4.2.1 建立连接

一般来说，应用在建立网络连接时会触发一系列初始化信息交互，随之而来的是一系列资源消耗，而这部分交互并不携带有用数据。

为避免诸如此类的低效建立连接，建议：

- 在连接建立初始尽快下载较多资源；
- 尽可能使所有 TCP 包排列更紧凑；
- 对部分可预测的用户需求，可在建立连接时下载。

在 App 在网络传输过程中，测试过程中数据请求的发起动作相对集中。 集中程度可根据业务场景，以及开发者对 App 的优化期望度定制化规定，在此不进行量化要求。

4.2.2 周期连接

心跳连接和数据更新。这两点对开发者保持应用与服务器正常连接，以及获取用户行为相关的分析数据，从而改进应用都有重要意义；

但是，每次周期性连接都会产生大量控制信息，如果管理不当，随着时间的积累，耗费在周期性连接上的资源将会超过交换用户真正需求的消耗。

在业务场景允许的情况下，App 在网络传输过程中，应无明显的周期性网络连接模式。对于特殊场景和业务要求，开发者可自行权衡。

4.2.3 并发连接

如果应用在同一时间段建立了多个并发 TCP 连接，由于网络总容量有限，每个连接的吞吐量必然随连接数量增多而减少，造成有效信息的传输大小受限，建立连接所消耗的头文件反而增多，降低资源使用效率。

建议应用避免在同一时段建立多个并发性连接，即尽可能将多个需求组合成后发送。开发者可根据应用的自身特点，选择在客户端与服务器间建立长连接，同时结合 HTTP pipelining 来解决这一问题。

在业务场景允许的情况下，App 在网络传输过程中，不应出现同时并发的 TCP 连接。对于特殊场景和业务要求，开发者可自行权衡。

4.2.4 屏幕旋转

设备屏幕旋转会造成页面重新加载和数据采集信息的传送，如果过于频繁的话会造成能量浪费和网络拥挤。

App 在网络传输过程中，不宜出现因屏幕旋转而造成的网络连接。

4.2.5 关闭连接

很多开发者在设计应用时常常不注意网络连接的关闭问题，以至于很多不再进行交互的连接依然占用着网络资源无法释放。这类未被释放的连接通常要等到网络连接超时被触发后，才能自动关闭。然而，在触发超时后，终端将自动进入高能量状态，以便关闭连接，消耗的一系列能量和资源仅仅为了关闭一个不需要的连接，这种使用方法是十分低效的。

建议开发者在建立每个网络连接时，在有效信息传输完毕后尽量将其立即关闭（特殊需长连接的情况除外）。具体方法可采用将 FIN 位置设为 1，与最后一个有效信息包捆绑发送。

App 在网络传输过程中，用于超时后的关闭连接请求的能量占比应少于 5%的传输耗能。

4.3 加载性能

4.3.1 响应错误

HTTP 响应状态字的第一位取值为 1-5，分别代表了该相应的返回类，如 3xx 类代表了资源（URI）的重定向；4xx 类代表客户端请求异常；5xx 类代表服务器端的异常。其中，404(Not

Found)应最为人们所熟知。客户端应具备识别状态字并进行相应处理的能力。

理论上，高效使用网络的应用在使用过程中不应返回任何错误；因此，当连接过程中出现代表错误的状态字时，开发者应根据状态字提示，溯源其网络连接过程，定位错误并对其进行改正。

App 在页面加载过程中，应未被返回任何 301、302、400、500HTTP 响应状态码。因负载均衡原因引起的 3xx 重定向代码除外。

4.3.2 第三方脚本使用

如 4.3.1 所述，高效使用网络的应用在使用过程中不应返回任何错误。第三方脚本由于不可控性大，出现错误返回的几率高，应尽量避免使用。

App 在页面加载过程中，任何页面中都不应存在多于两个第三方脚本的调用。

4.3.3 JS 异步加载

当 Java Script 作为 HTML 的头文件进行加载时，页面的其他加载都将受其影响而延迟。这是由于在同步加载过程中，HTML 正文必须在其头文件完成全部加载后才可继续；而当 Java Script 复杂度很高时，其大小通常也会增加，若在头文件中使用，将会影响页面渲染速度，响应时间过长，降低用户体验。

因此，App 在页面加载过程中，JavaScript 脚本应采用异步加载。

4.3.4 JS/CSS 加载顺序

同样情况，CSS 文件和 Java Script 的加载顺序也会影响到 HTML 页面的响应时长。若 CSS 在 Java Script 前加载，则可与页面渲染同步进行，提高页面显示速度；反之，下载和渲染只能在 Java Script 完成加载后才能依次进行，降低加载效率。加之，有些 Java Script 的执行本身就依赖于 CSS 文件中的某些属性设定，故只能等待 CSS 文件完全加载后才能全部执行。

App 在页面加载过程中，如出现同时加载 CSS 和 JS 文件的情况，应先加载 CSS 文件。

4.3.5 HTTP1.0 使用

HTTP1.1 相对于 HTTP1.0 有很多新特性和优势。

因此，App 在网络交互以及页面加载过程中，不应有任何 HTTP1.0 的使用。

4.3.6 HTML 空属性

一般来说，将 HTML 标签中的无值属性称为“空属性”。在 HTML5 中，明确规定了对此类空属性的处理方法；而对于非 HTML5 的情况，浏览器通常会依然向服务器发起请求，造成不必要的资源浪费。

综上所诉，建议开发者可以：

- 在使用 Java Script 作为 HTML 头文件时，尽量使用异步加载；
- Java Script 的加载顺序应置于 CSS 文件之后；
- 若非 HTML5，在标签中避免出现空属性。

App 在页面加载过程中，不应出现两个以上的属性设置为空的情况。

4.3.7 CSS 中 DisplayNone 的使用

CSS（层叠样式表）中的”display:none”属性是用来隐藏 HTML 脚本中不希望显示在页面上的对象。然而，这些被隐藏的对象通常也会被下载，引起一定的延迟和资源消耗。

App 在页面加载过程中，不应出现在 CSS 中使用“display: none”的情况。

4.3.8 Flash 文件使用

由于 Android4.0 以上以及 IOS 系统都不再支持 Flash，应避免在应用中使用。

App 在页面加载过程中，不应检测到任何 Flash 的使用。

5 测试方法

5.1 传输内容

5.1.1 内容压缩

测试编号：5.1.1
测试目的： 压缩后的文本可减少下载流量、提高下载速度、提升用户体验。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的业务场景； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： App 在网络传输过程中发送超过的未压缩文本类文件应不超过所有文本文件的 5%。

5.1.2 重复内容

测试编号：5.1.2
测试目的： 反复下载重复内容造成耗时、带宽、电池资源的浪费。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1) 启动被测应用； 2) 使用被测应用能触发联网交互操作的业务场景； 3) 被测场景应包含回退之前访问过的页面的操作； 4) 持续使用联网交互操作的业务场景不低于 5 分钟； 5) 抓取网络交互包并对其进行分析。
预期结果： 检测到的重复下载的资源类文件应不超过 3 项。

5.1.3 缓存使用

测试编号：5.1.3
测试目的： 未加入缓存头对内容进行标记，会造成反复下载重复内容造成耗时、带宽、电池资源的浪费。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1) 启动被测应用； 2) 使用被测应用能触发联网交互操作的业务场景； 3) 被测场景应包含回退之前访问过的页面的操作； 4) 持续使用联网交互操作的业务场景不低于 5 分钟； 5) 抓取网络交互包并对其进行分析。
预期结果： App 在报文交互过程中，所有下载内容均应使用缓存控制信息。

5.1.4 过期缓存

测试编号：5.1.4
测试目的： 反复下载重复内容造成耗时、带宽、电池资源的浪费。

预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1) 启动被测应用； 2) 使用被测应用能触发联网交互操作的业务场景； 3) 被测场景应包含回退之前访问过的页面的操作； 4) 持续使用联网交互操作的业务场景不低于 5 分钟； 5) 抓取网络交互包，并对其进行分析。
预期结果： 对于服务器返回 304 代码，应直接从缓存内读取信息，而非依然向服务器请求完整的资源重下载。

5.1.5 合并请求

测试编号：5.1.5
测试目的： 对同一类型文件的多次请求会影响页面读取速度。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1) 启动被测应用； 2) 使用被测应用能触发联网交互操作的业务场景； 3) 持续使用联网交互操作的业务场景不低于 5 分钟； 4) 抓取网络交互包并对其进行分析。
预期结果： 在 2s 内检测到不高于 2 个单独的 CSS 或 JS 请求。

5.1.6 图片处理

测试编号：5.1.6
测试目的： 如果下载了过大的图片，渲染时还要再缩小为适合的填充大小，既浪费流量也降低用户体验。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1) 启动被测应用； 2) 使用被测应用能触发图片下载的业务场景；

3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 所有下载图片的规格都应在填充区域大小的 150%以内。

5.1.7 信息精简

测试编号：5.1.7
测试目的： 文件中通常会添加空格等方便人类阅读的标识；然而对于机器来说他们毫无意义，还会浪费资源。
预置条件： a） 被测应用具备能触发联网交互操作的业务场景； b） 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的业务场景； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 可精简的信息应不超过总信息量的 5%。

5.1.8 图片组合

测试编号：5.1.8
测试目的： 建立快速网络连接的最佳方式为减少 http 请求数量，因此将多个小图文件打包为一张大图，可减少网络请求数量，甚至节省一定流量。
预置条件： a） 被测应用具备能触发联网交互操作的业务场景； b） 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发图片下载的业务场景； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： App 网络交互过程中不应由有可组合的小图片流。

5.2 传输过程

5.2.1 建立连接

测试编号：5.2.1
测试目的： 网络连接建立时，会产生 TCP 数据流（TCP burst），其后会依次产生若干数据流，这会增加应用响应时间和能量消耗。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的业务场景； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中数据请求的发起动作相对集中。

5.2.2 周期连接

测试编号：5.1.1
测试目的： 周期性数据传输如果处理不当，会造成能量浪费和应用性能下降。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的业务场景； 3） 在主页面停留，不进行任何操作，保持此状态不低于 20 分钟； 4） 抓取整个测试过程中的网络交互包，并对其进行分析。
预期结果： 测试过程中未检测明显的周期性网络连接模式。

5.2.3 并发连接

测试编号：5.2.3
测试目的：

持续连接的 TCP 连接，在性能上优于多个同时的 TCP 连接。可以减轻网络负载，消除建立不必要网络连接的时间。
预置条件： c) 被测应用具备能触发联网交互操作的业务场景； d) 测试终端运行的其他 app 进程关闭。
测试步骤： 5） 启动被测应用； 6） 使用被测应用能触发联网交互操作的业务场景； 7） 持续使用联网交互操作的业务场景不低于 5 分钟； 8） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中未检测到同时并发的 TCP 连接。

5.2.4 屏幕旋转

测试编号：5.2.4
测试目的： 设备屏幕旋转会造成页面重新加载和数据采集信息的传送，如果过于频繁的话会造成能量浪费和网络拥挤。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。 c) 终端旋转屏幕锁定为关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的业务场景； 3） 操作过程应包含多次旋转手机，并触发不少于 5 个不同页面的横竖屏切换动作； 4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中应未检测到屏幕旋转或存在屏幕旋转但未造成网络活动。

5.2.5 关闭连接

测试编号：5.2.5
测试目的： 如果数据传输完成后没有关闭连接，超时后的关闭连接请求会占用信道，并造成能量浪费。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤：

1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的业务场景； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 检测到用于超时后的关闭连接请求的能量占比应少于 5%的传输耗能。

5.3 加载性能

5.3.1 响应错误

测试编号：5.3.1
测试目的： 客户端请求异常、服务器异常的错误会导致网络使用效率降低；资源重定向会导致网络使用效率降低。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的业务场景； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中，未检测到任何 301、302、400、500HTTP 响应状态码；负载均衡原因引起的 3xx 重定向代码除外。

5.3.2 第三方脚本使用

测试编号：5.3.2
测试目的： 加载第三方脚本会影响应用的响应速度。如果第三方脚本加载失败，会极大地影响用户体验。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的业务场景； 3） 持续使用联网交互操作的业务场景不低于 5 分钟；

4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中，任何页面中都不应存在多于两个第三方脚本的调用。

5.3.3 JS 异步加载

测试编号：5.3.3
测试目的： 位于加载效率文件 HEAD 部分的 JavaScript 文件如果同步加载会影响整个加载效率文件的加载。
预置条件： a） 被测应用具备能触发联网交互操作的业务场景，且使用到了 HTML 编写 app 界面； b） 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的 HTML 页面； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中，JavaScript 脚本应采用异步加载。

5.3.4 JS/CSS 加载顺序

测试编号：5.3.4
测试目的： CSS 和 JavaScript 的加载顺序影响整个页面的渲染时间。
预置条件： a） 被测应用具备能触发联网交互操作的业务场景，且使用到了 HTML 编写 app 界面； b） 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的 HTML 页面； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中，如出现同时加载 CSS 和 JS 文件的情况，应先加载 CSS 文件。

5.3.5 HTTP1.0 使用

测试编号：5.3.5
测试目的： HTTP1.1 相对于 HTTP1.0 有很多新特性和优势。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景，且使用到了 HTML 编写 app 界面； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的 HTML 页面； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中，不应检测到 HTTP1.0 的使用。

5.3.6 HTML 空属性

测试编号：5.3.6
测试目的： CSS 和 JavaScript 的加载顺序影响整个页面的渲染时间。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景，且使用到了 HTML 编写 app 界面； b) 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的 HTML 页面； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中，不应检测到两个以上的属性设置为空的情况。

5.3.7 CSS 中 DisplayNone 的使用

测试编号：5.3.7
测试目的： CSS 中的“display: none”规则可以不显示某个对象，但是仍然会下载，有可能影响 app 的速度。
预置条件： a) 被测应用具备能触发联网交互操作的业务场景，且使用到了 HTML 编写 app 界面；

b） 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的 HTML 页面； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中，不应检测到 CSS 中使用“display: none”的情况。

5.3.8 Flash 文件使用

测试编号：5.3.8
测试目的： 最新的 Android 系统和 IOS 系统都不支持 Flash，应避免使用 Flash 内容，可以使用加载效率 5 标签代替。
预置条件： a） 被测应用具备能触发联网交互操作的业务场景，且使用到了 HTML 编写 app 界面； b） 测试终端运行的其他 app 进程关闭。
测试步骤： 1） 启动被测应用； 2） 使用被测应用能触发联网交互操作的 HTML 页面； 3） 持续使用联网交互操作的业务场景不低于 5 分钟； 4） 抓取网络交互包并对其进行分析。
预期结果： 测试过程中，不应检测到任何 Flash 的使用。

6 评价方法

6.1 指标集

6.1.1 概述

本标准指标集共分为两个层次，指标集的具体释义及要求请参照第四章技术要求。

6.1.2 第一层指标集

$$U = (U_1, U_2, U_3) = \{\text{传输内容，传输过程，加载性能}\}$$

6.1.3 第二层指标集

$$U_1 = (U_{11}, U_{12}, U_{13}, U_{14}, U_{15}, U_{16}, U_{17}, U_{18})$$

$=\{$文本压缩，重复内容，缓存使用，过期缓存，合并请求，图片处理，信息精简，图片组合$\}$

$U_2 = (U_{21}, U_{22}, U_{23}, U_{24}, U_{25}) = \{$建立连接，周期连接，并发连接，屏幕旋转，关闭连接$\}$

$$U_3 = (U_{31}, U_{32}, U_{33}, U_{34}, U_{35}, U_{36}, U_{37}, U_{38})$$

$=\{$响应错误，第三方脚本使用，JS 异步加载，JS /CSS 加载顺序，HTTP1.0 使用，HTML 空属性，CSS 中的 DisplayNone 的使用、Flash 文件使用$\}$

6.2 评价集

6.2.1 定量评价集

定量评价适用于评价集为是否的双元素类指标集。

$$V_1 = (V_{11}, \overline{V}_{11}) = \{\text{符合要求，不符合要求}\} = \{1,0\}$$

6.2.2 非定量评价集

非定量评价适用于评价集未给出明确定量判定指标，使用者可根据具体使用场景和测试需求自行定义的评价集。

此处给出非定量评价集参考建议，供本标准使用者参考。由于本标准指标评判主要依据为资源使用效率，此处以对资源的效率和浪费程度作为评价参考。

$V_2 = (V_{21}, V_{22}, V_{23}, V_{24}, V_{25}) = \{$优秀，良好，尚可，低效，无效$\} = \{1,0.75,0.5,0.25,0\}$

——优秀：资源利用效率极高，无浪费现象；

——良好：资源利用效率较高，存在轻度浪费现象；

——尚可：资源利用效率尚可，存在中度浪费现象；

——低效：资源利用效率较差，存在严重浪费现象；

——无效：资源无任何有效利用。

6.3 权重集

6.3.1 概述

权重反应了指标集中不同指标的重要性。使用者可根据自身需求调节权重系数，以制定满足评价需求的计算模型。由于本标准指标集共分为两个层次，每层次指标均有相应的权重系数。

权重系数取值范围为 0 到 1 ，且同一层级权重总和为 1。

6.3.2 第一层权重集

$$\mathrm{A} = (A_1, A_2, A_3)\ ,\ A_i \in [0,1]\ 且\ \sum A_i = 1$$

对应指标集：

$$\mathrm{U} = (U_1, U_2, U_3)$$

6.3.3 第二层权重集

$$A_1{}' = (A_{11}, A_{12}, A_{13}, A_{14}, A_{15}, A_{16}, A_{17}, A_{18})\ ,\ A_{1j} \in [0,1]\ 且\ \sum A_{1j} = 1$$

对应指标集：

$$U_1 = (U_{11}, U_{12}, U_{13}, U_{14}, U_{15}, U_{16}, U_{17}, U_{18})$$

$$A_2{}' = (A_{21}, A_{22}, A_{23}, A_{24}, A_{25})\ ,\ A_{2j} \in [0,1]\ 且\ \sum A_{2j} = 1$$

对应指标集：

$$U_2 = (U_{21}, U_{22}, U_{23}, U_{24}, U_{25})$$

$$A_3{}' = (A_{31}, A_{32}, A_{33}, A_{34}, A_{35}, A_{36}, A_{37}, A_{38})\ ,\ A_{3j} \in [0,1]\ 且\ \sum A_{3j} = 1$$

对应指标集：

$$U_3 = (U_{31}, U_{32}, U_{33}, U_{34}, U_{35}, U_{36}, U_{37}, U_{38})$$

6.4 结果集

6.4.1 概述

依据本标准第五章测试方法对所有测试集进行测试，建立当次测试结果评价集 V′，并依据测试需求，对权重集取值，得到权重集 A′，经计算可得出本次测试的评价结果 R。

其中：

$\mathrm{V}' = (V'_1, V'_2, V'_3)$，V′为一级评价集，$V'_i$为二级评价集；

$\mathrm{A}' = (A'_1, A'_2, A'_3)$，V′为一级评价集，$A'_i$为二级评价集。

计算结果集时应从第二层开始，依次向上一级得出最终评价结果。

6.4.2 第二层结果集

$$R_1 = A'_1 V'_1$$
$$R_2 = A'_2 V'_2$$
$$R_3 = A'_3 V'_3$$

6.4.3 第一层结果集

$$\mathrm{R} = \mathrm{A}'\mathrm{V}' = (A_1, A_2, A_3)\begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix}$$

其中，R 值为最终结果，且$R \in [0,1]$。

R 值越高，说明被测软件网络资源使用效率越高。软件的评价等级可参考下表。

移动智能终端应用软件网络资源使用效率评价参考建议表				
优秀	良好	尚可	有待提高	不合格
[1,0.8]	(0.8,0.6]	(0.6,0.4]	(0.4,0.2]	(0.2,0]

附录 A
（规范性附录）
标准修订历史

表 A.1标准修订历史

修订时间	修订后版本号	修订内容
2017.07.05	V1.0.0	第一次版本发布