Enhancing live editing commands by media content injection

Marcelo F. Moreno
Computer Science Department
Federal University of Juiz de Fora
Juiz de Fora MG Brazil
moreno@ice.ufjf.br

ABSTRACT

Since its conception, Ginga-NCL supports live editing commands that allows broadcasters and their applications to change the behavior of NCL document presentations. This can be achieved via stream events inserted into the broadcast stream, or via Lua scripts in the same application. Recently, editing commands may be posted via apps running on smart devices in the home network. Considering this last mode, one may imagine a use case where smart devices can be used to change the behavior of Ginga-NCL apps including not only the command itself, but also new media content related to the command, like photos and videos. This contribution proposes the addition of media injection accompanying live editing commands.

KEYWORDS

Live editing commands, media injection, interactivity, NCL

1 Introduction

Ginga applications are collected within a data structure known as a private base [1]. Private base applications can be initiated, paused, resumed and stopped and can refer to one another.

The core of the Ginga-NCL presentation engine [1] consists of the NCL formatter. The NCL formatter is in charge of receiving an NCL document and controlling its presentation, trying to guarantee that the specified relationships between the media objects are respected. The formatter deals with NCL documents that are collected inside a private base. Ginga associates one or more private bases with each TV channel.

The Private Base Manager is the Ginga module in charge of receiving commands to manipulate the private bases and the applications they contain. In particular, for receiving NCL documents, editing commands and to edit active NCL documents (documents being presented)

Since its conception, Ginga-NCL supports live editing commands that therefore allows broadcasters and their applications to change the behavior of NCL document presentations. This can be achieved via stream events inserted into the broadcast stream, or via Lua scripts in the same application. Recently, editing commands may be posted via apps running on smart devices in the home network.

In: Future of Interactive Television Workshop (V WTVDI), Rio de Janeiro, Brasil. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Porto Alegre: Sociedade Brasileira de Computação, 2019.

ISSN: 2596-1683

2 Proposal

In order to support media content injection related to the editing command, this contribution proposes a piggyback approach where the media file content is included in the JSON message posted according to Ginga-CC Webservices API 8.3.15 (see ABNT NBR 15606-11 [2]).

The current specification of the API for sending editing commands is proposed to be modified as follows. Changes are highlighted in italic and bordered paragraphs.

2.1 Changes taking ABNT 15606-11 as a basis

==== Beginning of ABNT NBR 15606-11 text excerpt ====

8.3.15. Sending editing commands to a running Ginga-NCL application

...

Request Format: `http(s)://<host>/dtv/current-service/apps/<appid>/edit[/<document-id>]/<command>/`

Type of Operation: `POST`

Description: Allows editing commands to be sent to a running Ginga-NCL application. Editing commands allow alteration of the content and behavior of a Ginga-NCL application during runtime. This API offers the same behavior as the "edit" events class of the Lua "event" module, as described in ABNT NBR 15606-2. Thus, edit commands sent via this API only alter the presentation of the document, not the document itself. If <document-id> is omitted, one assumes the application's main document (entry point).

Query Parameters: –

Message Body: In case of removal commands:

```
{
   "elementId": "<id>"
}
```

where

"elementId" specifies the identifier of the element to be removed.

In case of addition commands:

```
{
   "parentId": "<id>",
   "xml": "<xml-code>"
   "data": "<mime-type>; <filename>;
           <media content in base64 format>"
}
```

where

"parentId" specifies the identifier of the node under which the new element is to be inserted, and "xml" specifies the NCL code corresponding to the new element. The "parentId" field is for specific use of certain edit commands and, in these cases, if omitted, assumes the document body identifier (see ABNT NBR 15606-2). "parentId" shall be ignored if a command is specified for which no use is entailed.

"data" is optional and may be used only for the addNode command, in case the command includes the content of the new media to be added to the NCL application. "data" is a string that contains the media type, in conformance with MIME specifications, followed by semicolon, followed by the filename, another semicolon and, finally, the serialized content of the media file to be added in base64 binary-to-text encoding scheme.

==== End of ABNT NBR 15606-11 text excerpt ====

REFERENCES

[1] ABNT. *ABNT NBR 15606-1: Data coding and transmission specification for digital broadcasting. Part 1: Data coding specification.* 2018.

[2] ABNT. *ABNT NBR 15606-11: Data coding and transmission specification for digital broadcasting. Part 11: Ginga CC WebServices - Ginga Common Core WebServices specification.* 2018.