1 Introduction

MPEG-4 supports two forms of user interaction with a presentation (scene); local interactivity and remote interactivity. Local interactivity can be fully implemented using the native event architecture of BIFS, as well as the BIFS update mechanism for events generated by a hosting application, scripts, and AAVS. Remote interactivity is interacting with a remote user (client and/or server). Remote interactivity is supported by means of control messages over the MPEG-4 systems back-channel. This section describes the use of CommandDescriptor and CommandROUTE mechanism to support user interaction. The control message that is sent back to the server is formatted into a Command that is encapsulated in a timestamped ALPDU. In this discussion, server refers to a peer in the session.

The syntax and semantics of the CommandDescriptor, CommandROUTE and Command is given below.

2 Command Descriptor

class CommandDescriptor: bit(8) commandDescriptorTag = 0x06 { 
    bit(10) CommandDescriptorID;
    // stream count; number of ES_IDs associated to this message
    unsigned int (8) count;
    // ES_Id of the streams
    unsigned int (16) ES_ID[count];
    // application-defined parameters
    do { 
      unsigned int (8) paramLength;
      char (8) commandParam[paramLength];
    } 
    while (paramLength!=0);
}
CommandDescriptorID: 10-bit value uniquely identifying this command descriptor. The value of the command descriptor determines how the server interprets the Command.

count: number of objects the CD applies to. Only nodes that have their ES_ID listed may use the CD. A value of 0 indicates it applies to all media objects.

ES_ID: ES_ID of the objects affected by this message. This information can be useful, for example, for simple implementation of commands such as stop, pause etc.

paramLength: length of the command parameter.

commandParam: application-specific parameter string.

3 Command ROUTE

The ROUTEs mechanism is extended by specifying a CommandROUTE that is used to route control messages to a server (client). A CommandROUTE can be updated using CommandROUTE update messages.

class CommandROUTE {
    bit(1) isUpdateable;
    if (isUpdateable)
        bit(10) CommandROUTEid;
    bit(10) outNodeID;
    outID outFieldReference;

    bit(10) CDid; // command descriptor ID
}

isUpdatable: is the server route updateable
CommandROUTEid: route id. A maximum of 1024 routes are possible.
OutNodeID: see ROUTE definition
CDid: command descriptor ID. Specifies the ID of the command descriptor communicated to the server.

TBD: Define CommandROUTE add, remove, update.

4 Command

This section documents the syntax and semantics of the command, as it is transmitted back to the server. Transmission is performed using AL packetization. This allows timestamping to be used (if the AL is configured as such using DMIF facilities when setting up the back channel(s)).

class Command {

    // which node generated this message
    unsigned int(16) nodeID;

    // command descriptor ID
    bit(10) CDid;

    // the following are simply copied from the command descriptor
    // stream count; number of ES_IDS associated to this message
    unsigned int (8) count;
}
// ES_Id of the streams
unsigned int (16) ES_ID[count];

// application-defined parameters
do {
    unsigned int (8) paramLength;
    char (8) commandParam [paramLength];
}
while (paramLength!=0);

nodeID: The nodeID of the node associated with the command. This field must be filled by the MPEG-4 terminal based on the node that directly pointed to the command descriptor.

5 Usage

A CommandDescriptor is associated with a command rather than a node. The implication is that several objects could use the same CommandDescriptor. The parameters in the CommandDescriptor allow the server to save the state information in a similar way a ‘COOKIE’ does in web browsers. This is desirable as it is difficult for a server to maintain state for all the objects in all the sessions running on it. The usage of CommandDescriptor, CommandROUTE and Command is described in this section.

5.1 Pre defined Descriptors

To support common forms of user interaction, a set of CommandDescriptors are defined. All terminals shall process these CommandDescriptors in the same way. In addition to the standardized ones, content developers can define application specific CommandDescriptors. The following is a list of pre-defined CommandDescriptors.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X0000</td>
<td>MPEG4 reserved</td>
</tr>
<tr>
<td>0X0001</td>
<td>Start</td>
</tr>
<tr>
<td>0X0002</td>
<td>Stop</td>
</tr>
<tr>
<td>0X0003</td>
<td>Pause</td>
</tr>
<tr>
<td>0X0004</td>
<td>Forward</td>
</tr>
<tr>
<td>0X0005</td>
<td>Reverse</td>
</tr>
<tr>
<td>0X0006</td>
<td>Remove</td>
</tr>
<tr>
<td>0X0007</td>
<td>Add</td>
</tr>
<tr>
<td>0X0008</td>
<td>Transaction</td>
</tr>
<tr>
<td>0X0009 – 0X00FF</td>
<td>MPEG-4 reserved</td>
</tr>
<tr>
<td>0X0100 – 0XFFFF</td>
<td>User Defined</td>
</tr>
</tbody>
</table>

For the play control messages, the CommandDescriptor contains one default parameter. The BIFS renderer shall increment parameter count and add the value from the out field reference of the source of the CommandROUTE in the Command communicated to the server (other end of the session). The user added parameter if any, is the value used by the server.

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Parameter Count</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>0X0001</td>
<td>1</td>
<td>0 (start now)</td>
</tr>
</tbody>
</table>
Stop 0X0002 1 0 (stop now) Application added
Pause 0X0003 1 0 (pause now) Application added
Forward 0X0004 1 2 (rate; forward until end – expecting a pause or stop or play) Application added
Reverse 0X0005 1 2 (rate; reverse until end – expecting a pause or stop or play) Application added

A remove descriptor tells the server to remove a media object from a scene (by sending a node delete command) and stop transmitting the objects. Remove contains the ES_ID in the parameter list. Add descriptor adds a new object to a scene (server sends a node add command and the necessary object descriptors). Add contains the ID of the object that server adds as a result of this message.

The transaction CommandDescriptor is a generic descriptor to support transactions. This is geared towards e-commerce applications. For example, if a user wants to buy a pair of sunglasses like the ones James Bond is wearing in the movie, there must be a way for the user to select the sunglasses (as opposed to the jacket which is also available for sale). Such transactions can be performed using the transaction descriptor. Even though AAVS is expected to provide such functionality, a non-AAVS version still needs such capabilities. The parameter list is given below.

Parameter count: 6 (minimum) or more

Session ID: Current session ID (Eg. Item billed to your session account)
Name: Transaction name (Eg. PURCHASE)
Choice count: Number of available choices
Choices: Available choices (Eg. Jacket, Sunglasses)
Value: Eg. Cost
Information: Eg. You have to be 18 or older to make a purchase 😊

A BIFS renderer presents this information in a structured manner by popping up a window and upon any choices (if any) and OK from the user, communicates a CD to the server.

5.2 Application Defined Descriptors

Application designers may specify descriptors to perform different functionalities. However, a BIFS renderer knows only how to read a descriptor and add a field to the end of the parameter list. A BIFS renderer cannot render every user-defined descriptor differently, as the renderer is not ‘downloadable’ and should work across servers.

In application-defined descriptors, the server inserts the ‘state’ information that has to be acted upon when a command is received. For example, consider a multi-user session (Eg. a collaborative editing session). All the participants see the same objects on the screen. The changes made by one, for example by moving an object should be propagated to others in the session. This can be supported by a CommandDescriptor attached to the event that changes an object’s position. The CommandDescriptor can contain an the list of session IDs of others participating in the editing session. A change in the position of an object is communicated to the server, that generates BIFS updates for the remaining participants in the session. Another application is to add new objects to a session. There are many value added applications content creators can design using CommandDescriptors.
6 Communicating Command Descriptors and Commands

CommandROUTEs are be placed at the end of a scene description immediately following the ROUTEs. CommandDescriptors however may be communicated as a separate elementary stream. These are communicated on a separate channel. CommandDescriptors may be replaced by sending another one with the same ID. This acts as command descriptor replace/update.

A formatted command is sent to the server on the systems back channel. The commands in the back channel are ALPDU encapsulated. Each packet on the back channel consists of length of an ALPDU followed by AL PDU. All the complexities of ALPDUs may not be necessary here. Is there any issue with transmitting raw commands?