Supporting rights checking in an MPEG-21 Digital Item Processing environment

Frederik De Keukelaere[(i)], Thomas DeMartini[(ii)], Jeroen Bekaert[(i)], Rik Van de Walle[(i)]
[(i)]*Multimedia Lab, Ghent University-IBBT, Belgium*
[(ii)]*ContentGuard, USA*
frederik.dekeukelaere@ugent.be, thomas.demartini@contentguard.com,
jeroen.bekaert@ugent.be, rik.vandewalle@ugent.be

Abstract

Within the world of multimedia, the new MPEG-21 standard is currently under development. The purpose of this new standard is to create an open framework for multimedia delivery and consumption. MPEG-21 mastered the multitude of types of content and metadata by standardizing the declaration of Digital Items in an XML based format. In addition to standardizing the declaration of Digital Items MPEG-21 also standardizes Digital Item Processing, which enables the declaration of suggested uses of Digital Items. The Rights Expression Language and the Rights Data Dictionary parts of MPEG-21 enable the declaration of what rights (permitted interactions) Users are given to Digital Items. In this paper we describe how rights checking can be realized in an environment in which interactions with Digital Items are declared through Digital Item Processing. We demonstrate how rights checking can be done when "critical" Digital Item Base Operations are called and how rights context information can be gathered by tracking during the execution of Digital Item Methods.

1. Introduction

Currently, there are many types of digital content and probably just as many possible ways of describing them and the context in which they can be used. Within any system that proposes to facilitate a wide range of actions involving content, there is a need for a very precise definition of what exactly constitutes such a "Digital Item".

This need resulted in a strong challenge to design a powerful and flexible model for Digital Items (DIs) [1] that is able to accommodate the variety of forms that content can take and provide support for new forms of content that will be developed in the future. The success of such a model rests on its capability to express these forms, including the metadata, in an interoperable manner. This challenge has been tackled by the Moving Picture Experts Group (MPEG) in MPEG-21 [2] part 2 [3], [4], which expresses such a model for declaring Digital Items in an unambiguous and interoperable manner.

To be appealing to industry, multimedia content often needs to be combined with a mechanism that allows expressing what interactions are allowed with it. For example, a content distributor might want to give a license to a consumer to play a resource if the consumer paid a required fee. For this reason MPEG standardized the Rights Expression Language (REL) [5], [6] and the Rights Data Dictionary (RDD) [7], [8], enabling the creation and association of licenses for multimedia content.

Going beyond the declaration of Digital Items and expression of rights, MPEG-21 Part 10, Digital Item Processing (DIP) [9], [10], is standardizing a set of tools allowing the declaration of suggested uses of a Digital Item in an interoperable way. These suggested uses are described at a level that is typically more specific than the level at which rights are described. The suggested uses also typically refer to multiple resources and pieces of metadata, describing their joint use. For example, a suggested use might be playing a resource adapted for a particular environment described by a particular piece of metadata. These very specific uses are suggested by adding Digital Item Methods (DIMs), which are in fact scripts, to Digital Items or parts thereof. By analogy with the World Wide Web, a Digital Item Method is to a Digital Item as a JavaScript function is to a web page. In other words, Digital Item Processing enables active content (DIMs) to reside inside a Digital Item.

The combination of rights expressions that enumerate permitted interactions at one level with active content that enumerates suggested uses at a

0-7803-9332-5/05/$20.00 ©2005 IEEE

more specific level presents a problem. It needs to be possible to determine which sequences of specific instructions in the active content correspond to which generic interactions in the rights expressions. As will be shown in sections 2 and 3, either situation (rights expressions or active content) independently is relatively straight-forward. However, the combination can also be handled using the approach outlined in section 4.

2. Rights expressions that enumerate permitted interactions at one level with application code at a more specific level

When creating an application that interacts with Digital Items and that doesn't support DIP or any other means of active content, there is still the distinction between the level at which the rights are expressed and the level at which the code is written. However, in this case, the implementer understands, in advance, how their code interacts with Digital Items. Being well-behaved is simply a matter of finding the appropriate points in their implementation during its interaction with a Digital Item to check for the rights to do that interaction.

For example, suppose we have created an application that adapts a JPEG 2000 [11] picture by reducing the resolution, stores the adapted version of the picture in memory, and then displays the picture. A possible example of an appropriate point to check rights could be the time at which the adapted resource is to be displayed. At that point the application can check the availability of rights required to play an adapted version of that resource. Such a check could not be done earlier, since the application could have the right to adapt the resource and store it to disk, but not the right to adapt the resource and play it. Therefore in this case the appropriate point in time to check for a “play adapted resource” right is just before the playing of the adapted resource.

3. Active content (suggested uses separate from application code)

An implementer who is not worried about creating a well-behaved application in terms of rights does not need an intimate understanding of how their implementation is interacting with Digital Items. Supporting DIP is a matter of leaving a certain amount of the decision of how to interact with a Digital Item up to the DIM author. The implementer can program the application to blindly follow the instructions in the DIM without any care as to what is actually happening. Although this possibility is not of much interest due to the pervasiveness of script interpreters, which make this possibility common-place, we have included it here for the sake of completeness.

4. Rights expressions that enumerate permitted interactions at one level combined with active content that enumerates suggested usues at another level

An implementer who supports DIP and also desires to be well-behaved in terms of rights needs to address the challenge of being able to check for the rights to interact with the Digital Item in the way suggested by the DIM author at the appropriate times.

By using DIP Digital Item Methods (DIMs) and Digital Item Base Operations (DIBOs) an author can express his suggested uses of a Digital Item.

A **Digital Item Method (DIM)** expresses a suggested use of a Digital Item by an MPEG-21 User [1] (e.g., a human consumer). It contains calls to Digital Item Base Operations and describes the sequence in which those calls are executed.

A **Digital Item Base Operation (DIBO)** is the smallest possible elementary action that can be performed on a Digital Item. The Digital Item Base Operations available at an MPEG-21 peer can be seen as a set of functions that can be used for authoring a Digital Item Method. The set of Digital Item Base Operations is therefore a library of functions for the Digital Item Method Language.

Since the DIBOs are more detailed than the base rights (i.e., Adapt, Delete, Diminish, Embed, Enhance, Enlarge, Execute, Install, Modify, Move, Play, Print, Reduce, and Uninstall) in the REL and RDD and since the implementer does not know ahead of time the sequences of DIBOs that the DIM author will use, the implementer needs to somehow, while processing the DIM, aggregate the effect of a sequence of DIBOs to determine the appropriate rights to check.

For example, the DIBO sequence of "`firstChild`" [12], "`firstChild`", "`DIA.AdaptResource`", and "`DIP.PlayResource`" might correspond to a rights check for the "Play" right. Another DIBO sequence of "`firstChild`", "`firstChild`", "`DIA.AdaptResource`", "`replaceChild`", and "`writeToURI`" might correspond to a rights check for the "Adapt" right. Even though both DIBO sequences begin with the same first three DIBOs, they result in different rights checks because they end with different "critical DIBOs" (in this case, "`DIP.PlayResource`" and "`writeToURI`", respectively). Critical DIBOs are those DIBOs that trigger rights

```
//Get the item out of the current DIDL Document
var itemA=didDocument.firstChild.firstChild;
//Load up another DIDL Document and get the item out of it
var registry=DOMImplementationRegistry.getDOMImplementation("LS");
var builder=registry.createLSParser(DOMImplementationLS.MODE_SYNCHRONOUS,null);
var document2=builder.parseURI("did2.xml");
//Get the item out of the other DIDL Document
var itemB=document2.firstChild.firstChild;
//Import the second item into the current DIDL Document
itemB=didDocument.importNode(itemB, true);
//Get the metadata and resource from the second item
var component=itemB.firstChild;
var metadata=component.firstChild;
var resource=component.lastChild;
//Adapt the resource according to the metadata and update the component
var adaptedResource=DIA.AdaptResource(resource, metadata);
component.replaceChild(adaptedResource, resource);
// Create a new descriptor with a statement with the text "cool"
var statement=document.createElement("Statement");
var cool=document.createTextNode("cool");
statement.appendChild(cool);
var descriptor=document.createElement("Descriptor");
descriptor.appendChild(statement);
// Add the descriptor
component.appendChild(descriptor);
// Add the second item as a child of the first item
itemA.appendChild(itemB);
// Write out the DIDL Document
var domWriter = registry.createLSSerializer();
domWriter.writeToURI(didDocument,"http://mydidlocation/did.xml");
```

Figure 1. Example DIM code

checks, i.e. the DIBOs that have rights associated with them. A list of those DIBOs can be found in Table 1.

Table 1. DIBO mapping to RDD verbs

Digital Item Base Operation	Rights Data Dictionary verb
`DIA.AdaptResource`	Adapt, Diminish, Enhance
`DIP.ExecuteResource`	Execute
`DIP.PlayResource`	Play
`DIP.PrintResource`	Print
`Write`	Adapt, Diminish, Enhance, Enlarge, Modify, Reduce
`writeToURI`	Adapt, Diminish, Enhance, Enlarge, Modify, Reduce
`writeToString`	Adapt, Diminish, Enhance, Enlarge, Modify, Reduce

Another example to consider is the DIBO sequence of "`firstChild`", "`firstChild`", "`DIP.PlayResource`". This sequence ends in the same "critical DIBO" as the first sequence (namely "`DIP.PlayResource`") and might also result in a rights check for the "Play" right. However, the context information for this rights check would differ from the context information for the rights check for the first example. The rights check for this example would have context information indicating that the resource will be played in its original form while the rights check for the first example would have context information indicating that the resource will be played in some adapted form (for instance, at reduced resolution).

The information necessary to perform the appropriate rights check using the appropriate context information can be collected by adding information tracking into the implementation of each of the DIBOs such as "`firstChild`" and "`DIA.AdaptResource`". The information that needs to be collected includes what the operation operated on and how the operation affected the things it operated on.

Let us have a closer look at a tracking mechanism based on the code from Figure 1. The above section of DIM code gets a first item from the current DIDL document. It then loads up a second item, changes it a bit, and then adds it to the first item. In the end, when the updated first item is written out, the first item was "enhanced" and the second item was "adapted". So before writing out the updates a well-behaved implementation should check for "Enhance" rights to the first item and "Adapt" rights to the second item. Figure 2 shows an example of the tracking information that can be stored during the execution of the DIM code to make it possible to figure out the correct rights to check.

"#document_9" represents the global DIDL document object. "Item_28" is the item that is stored out. Looking at the line before last of the tracking information, we can see that "Item_28" came by adding "Item_16" into "Item_11". So "Item_11" (itemA from Figure 1) was enhanced. We can also see that parts of "Item_16" were heavily changed and that "Item_16" originally came from "Item_14". So "Item_14" (itemB from Figure 1) was adapted. These two rights (enhancing of ItemA and adaptation of ItemB) would be checked. Thus, by using the information represented in Figure 2, we were able to determine the appropriate rights to check. If necessary, context information describing, for example, the number of nodes changed in ItemB could also be made available for the rights checking.

```
Pseudo code
#document_9.firstChild
DIDL_10.firstChild
parseURI
#document_12.firstChild
DIDL_13.firstChild
#document_9.importNode(Item_14)
Item_16.firstChild
Component_17.firstChild
Component_17.lastChild
DIA.adaptResource(Resource_19, Descriptor_18)
Component_17.replaceChild(Resource_20, Resource_19)
#document_9.createElement()
#document_9.createTextNode()
Statement_22.appendChild(#text_23)
#document_9.createElement()
Descriptor_25.appendChild(Statement_24)
Component_21.appendChild(Descriptor_26)
Item_11.appendChild(Item_16)
writeToURI
```

Original ID		New ID
xxxxxxxxxx	--->	DIDL_10
xxxxxxxxxx	--->	Item_11
xxxxxxxxxx	--->	#document_12
xxxxxxxxxx	--->	DIDL_13
xxxxxxxxxx	--->	Item_14
Item_14	--->	Item_16
xxxxxxxxxx	--->	Component_17
xxxxxxxxxx	--->	Descriptor_18
xxxxxxxxxx	--->	Resource_19
Resource_19	--->	Resource_20
Component_17	--->	Component_21
xxxxxxxxxx	--->	Statement_22
xxxxxxxxxx	--->	#text_23
Statement_22	--->	Statement_24
xxxxxxxxxx	--->	Descriptor_25
Descriptor_25	--->	Descriptor_26
Component_21	--->	Component_27
Item_11	--->	Item_28
#document_9	--->	#document_29

Figure 2. Example tracking information stored during execution of Figure 1

5. Conclusions

In this paper we have illustrated how various parts of the seminal MPEG-21 standard can be combined: MPEG-21 REL details a language to express rights pertaining to a Digital Item, MPEG-21 RDD details a set of clear and uniquely identified rights terms, and MPEG-21 DIP specifies the processing of user interactions associated with Digital Items. To guarantee rights are used in a correct way, an application must be well-behaved with respect to the permissions granted in those licenses.

The paper listed various possibilities for creating well-behaved implementations, both supporting DIP and not. Whenever static programs are created, it is possible to determine in advance what rights checks will be necessary to make sure that the application is behaving correctly. For non-static applications, such as DIP-enabled applications, the paper describes a tracking mechanism enabling collection of appropriate context information and run-time checking for appropriate rights at critical points during the execution of the DIM by the DIP application. Combining MPEG-21 DIP with other (MPEG-21) technology could eventually result in the creation of a trusted multimedia platform.

6. Acknowledgements

The research activities that have been described in this paper were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), the Belgian Federal Science Policy Office (BFSPO), the European Union, and ContentGuard.

7. References

[1] MPEG, "Text of DTR of ISO/IEC 21000-1 Second Edition," ISO/IEC JTC1/SC29/WG11 N6388, March, 2003.

[2] I. Burnett, R. Van de Walle, K. Hill, J. Bormans, and F. Pereira, "MPEG-21 Goals and Achievements," IEEE Multimedia, vol. 10 n°4, October-December 2003, p.60-70.

[3] ISO/IEC, "ISO/IEC 21000-2:2003 Information Technology - Multimedia Framework (MPEG-21) - Part 2: Digital Item Declaration," March 2003.

[4] I. Burnett, S. Davis, and G. Drury, "MPEG-21 Digital Item Declaration and Identification - Principles and Compression," IEEE Transactions on Multimedia, Special issue on MPEG-21, to be published in 2005.

[5] ISO/IEC, "ISO/IEC 21000-5:2004 Information Technology - Multimedia Framework (MPEG-21) - Part 5: Rights Expression Language," March 2004.

[6] X. Wang, T. DeMartini, B. Wragg, and M. Paramasivam, "The MPEG-21 Rights Expression Language," IEEE Transactions on Multimedia, Special issue on MPEG-21, to be published in 2005.

[7] ISO/IEC, "ISO/IEC FDIS 21000-6 Information Technology - Multimedia Framework (MPEG-21) — Part 6: Rights Data Dictionary," July 2003.

[8] C. Barlas, and G. Rust, "The MPEG Rights Data Dictionary," IEEE Transactions on Multimedia, in this Special issue on MPEG-21.

[9] MPEG, "ISO/IEC 21000-10 Information Technology - Multimedia Framework (MPEG-21) - Part 10: FCD Digital Item Processing," N6780, October 2003.

[10] F. De Keukelaere, S. De Zutter, and R. Van de Walle, "MPEG-21 Digital Item Processing," IEEE Transactions on Multimedia, Special issue on MPEG-21, to be published in 2005.

[11] M. Rabbani, and R. Joshi, "An overview of the JPEG 2000 still image compression standard," Signal Processing: Image Communication, vol. 17 n°1, January 2002, p.3-48.

[12] World Wide Web Consortium, "Document Object Model (DOM) Level 3 Core Specification Version 1.0," W3C Recommendation 07 April 2004.