 ISO/IEC JTC 1/SC 29/WG 03 N0287

**ISO/IEC JTC 1/SC 29/WG 03  
MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Technologies under Consideration for Network Based Media Processing

**Status:** Approved

**Date of document:** 2021-07-23

**Source:** ISO/IEC JTC 1/SC 29/WG 03

**Expected action:** None

**Action due date:** None

**No. of pages:** 43 (with cover page)

**Email of Convenor:** young.L @ samsung . com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 03 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 03 N** **0287**

**July 2021, Virtual**

|  |  |
| --- | --- |
| **Title** | **Technologies under Consideration for Network Based Media Processing** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **20541** |
| **Editor** | **Iraj Sodagar** |

This Technology under Consideration document collects candidate technologies for inclusion into ISO/IEC 23090-8 Network Based Media Processing Standard.

Contents

[ISO/IEC JTC 1/SC 29/WG 03 N0287 1](#_Toc77768177)

[**Expected action:** None 1](#_Toc77768178)

[**Action due date:** None 1](#_Toc77768179)

[1 Improvement of Security descriptor (proposed in m50768) 5](#_Toc77768180)

[1.1 Introduction 5](#_Toc77768181)

[1.2 Previous contribution for supporting protected function usecase 5](#_Toc77768182)

[1.3 Authorization for the protected resource 5](#_Toc77768183)

[1.3.1 Authorization done by NBMP source 5](#_Toc77768184)

[1.3.2 Authorization done by NBMP Workflow Manager 5](#_Toc77768185)

[1.4 Proposed update for supporting private function usage 6](#_Toc77768186)

[1.4.1 Proposed Usecase 6](#_Toc77768187)

[1.4.2 Proposal for update current security parameter 6](#_Toc77768188)

[9.6.1.13 Security parameters 7](#_Toc77768189)

[2 Update of Dynamic resource update on NBMP 9](#_Toc77768190)

[2.1 Dynamic resource update (proposed in m50771) 9](#_Toc77768191)

[2.1.1 Introduction 9](#_Toc77768192)

[2.1.2 Dynamic resource update during NBMP services 9](#_Toc77768193)

[2.1.3 Proposed update of current processing descriptor 10](#_Toc77768194)

[9.6.1 Processing Descriptor parameters 10](#_Toc77768195)

[2.2 NBMP Workflow continuity (m56712) 11](#_Toc77768196)

[2.2.1 Introduction 11](#_Toc77768197)

[2.2.2 Use case 11](#_Toc77768198)

[2.2.3 Proposed solutions 11](#_Toc77768199)

[2.3 NBMP Workflow and task Scheduling 13](#_Toc77768200)

[2.3.1 Scheduled-time (m56729) 13](#_Toc77768201)

[2.3.2 Task Group 16](#_Toc77768202)

[2.3.3 NBMP Workflow scheduling (m57293) 17](#_Toc77768203)

[2.3.4 Notes of MPEG#134 23](#_Toc77768204)

[2.3.5 Notes of MPEG#135 (on m57293) 23](#_Toc77768205)

[3 Exploration on split rendering with NBMP 25](#_Toc77768206)

[3.1 Introduction 25](#_Toc77768207)

[3.2 use-case (as defined N19030) 25](#_Toc77768208)

[3.3 Requirements (as defined N19030) 26](#_Toc77768209)

[3.4 Potential solutions based on m53671, m53673, m53675) 26](#_Toc77768210)

[3.4.1 Splitting workflow 26](#_Toc77768211)

[3.4.2 Workflow Reconfiguration 27](#_Toc77768212)

[3.4.3 Source and Sink System Events 30](#_Toc77768213)

[3.4.4 Split rendering between different network element or different cloud nodes 30](#_Toc77768214)

[4 Signalling push/pull and other characteristics of inputs and outputs 31](#_Toc77768215)

[4.1 Proposed Improvements to Input and Output Descriptor (m55963) 31](#_Toc77768216)

[4.1.1 Introduction 31](#_Toc77768217)

[4.1.2 [NBMP] New parameters to NBMP Input/Output Descriptor 31](#_Toc77768218)

[4.1.3 JSON Schema 34](#_Toc77768219)

[4.1.4 *References* 35](#_Toc77768220)

[4.1.5 Discussion at MPEG#133 35](#_Toc77768221)

[4.2 On improvements to NBMP input/output descriptors (m 56711) 37](#_Toc77768222)

[4.2.1 Introduction 37](#_Toc77768223)

[4.2.2 Proposed improvements 37](#_Toc77768224)

[4.2.3 Proposed improvements 39](#_Toc77768225)

[4.3 Discussion at MPEG#134 40](#_Toc77768226)

[5 Synchronized metadata handling (m 56728) 40](#_Toc77768227)

[5.1 Introduction 40](#_Toc77768228)

[5.2 Proposed design 41](#_Toc77768229)

[5.3 Discussion at MPEG#134 43](#_Toc77768230)

# Improvement of Security descriptor (proposed in m50768)

## Introduction

The current NBMP DIS describes a security descriptor and its parameters that are applicable to NBMP Function/Task/Workflow. At the last Geneva meeting, we discussed about the authorization for using protected function. It was agreed to consider the use case for using the security descriptor. However, it is not fully supported in current NBMP DIS. This contribution proposes to add the description about the usage of the security descriptors for supporting the protected function usecase.

## Previous contribution for supporting protected function usecase

For accessing the protected resource of NBMP services, authorization mechanisms have to be considered in the NBMP system. This section describes the NBMP system that needs to be authorized and provides mechanisms to do so. If the 3rd party function provider provides specific NBMP functions and the use of these functions require authorization, the NBMP source or NBMP workflow manager can do the authorization by using client information or client grants.

## Authorization for the protected resource

### Authorization done by NBMP source

The NBMP source initiates the NBMP service based on the workflow description. If the NBMP source wants to use the specific function which needs authorization from the function provider, it can be done using authorization based on the client information or client grants. After finishing the authorization for accessing the protected function, the NBMP source can send its authorization information, such as an access token, to the NBMP workflow manager through the Authorization descriptor in the workflow description. The NBMP workflow manager can create the workflow which contains the protected resource (NBMP function) by using this access token.

### Authorization done by NBMP Workflow Manager

If the NBMP source wants to use the specific function which needs authorization from the function provider, the NBMP source sends the client information or client grants for authorization by using the authorization descriptor. After receiving the workflow description, the NBMP workflow manager recognizes that a protected resource/function is needed for creating the workflow. The NBMP workflow manager can do the authorization for accessing the protected resource or function based on client information or client grants.

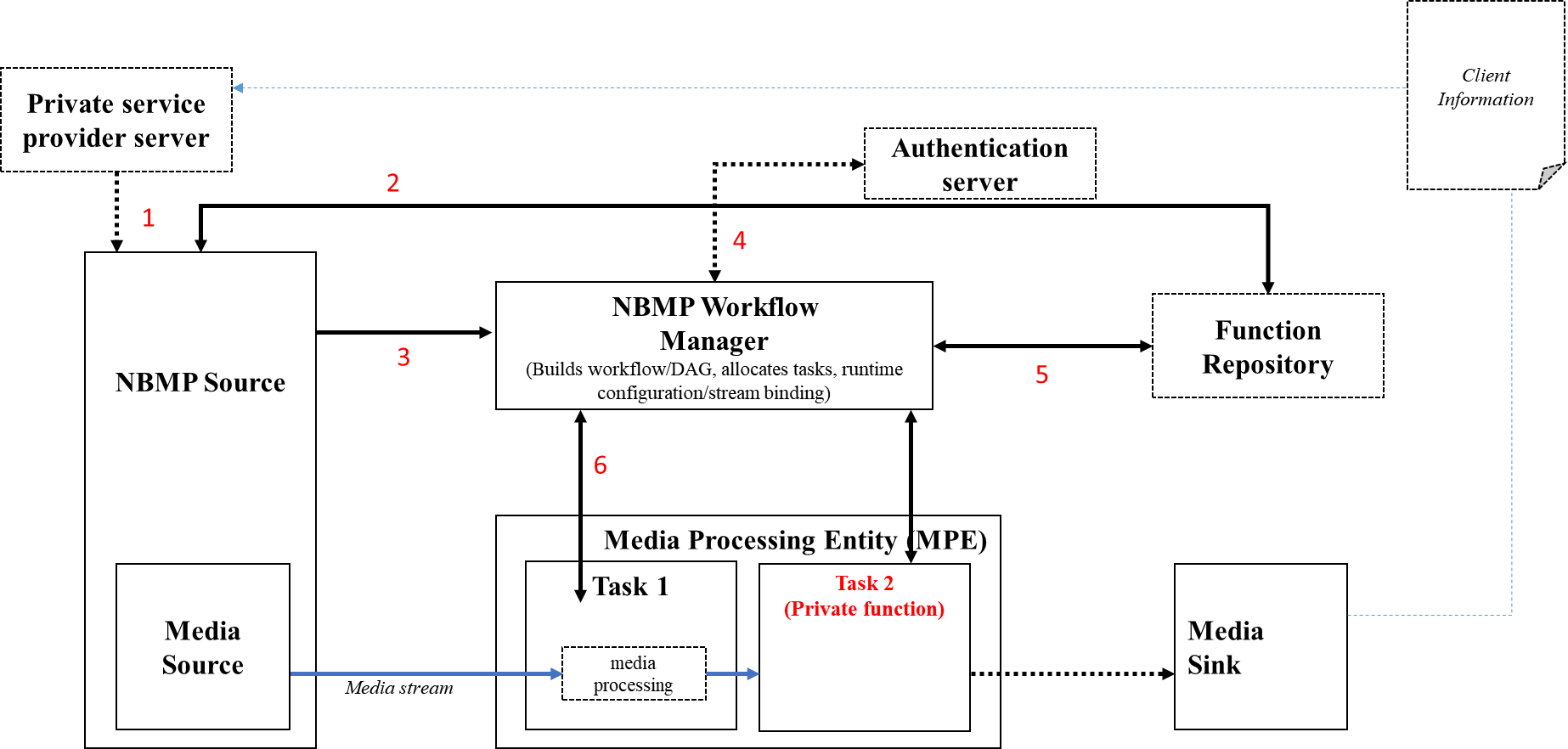
This descriptor provides details about the underlying resource (e.g., workflow, task) in which this is included. Following are the list of parameters in this descriptor.

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Description** | **Data Type** |
|
| Authorization flag | Indicates whether the resource requires/supports authorization for accessing | Boolean |
| Authorization protocol type | Provides authorization protocol type (i.e SAML 1.0/2.0, Oauth 1.0/2.0) for the resource | String |
| protected resource | Provides protected resource (NBMP service or NBMP Function) information | String |
| Authorization Configuration | Configuration details of Authorization parameter required for resource  . Access token  . Client grants | Array |

## Proposed update for supporting private function usage

### Proposed Usecase

Below figure shows the detailed usage of the private\_function\_flag for supporting the restricted resources during creating the workflow.



1. NBMP client requests an NBMP media service to the service provider/NBMP source.
2. The NBMP source receives the request for a creation of the workflow from the NBMP client and sends it to the WM using WDD. If the NBMP client requests the creation of a NBMP workflow which includes at least one specific function (Private function), then it requires qualified information for accessing the restricted function (Private function).
3. If the NBMP source has access to such qualified information, and can do the authentication process via an obtained Auth-token, then the NBMP source creates a WDD which includes the private\_function\_flag set to ‘0’ in the processing descriptor for the associated function. That means that this request of the workflow creation contains a private function and its associated auth-token is contained in the WDD.

If the NBMP source does not have access to such qualified information (and only knows that it is a restricted resource) for the authentication process, then the NBMP source sets the private\_function\_flag to ‘1’ in the processing descriptor for the associated function and sends the WDD it to WM. This implies that this request of the workflow creation contains a private function and that the WM should do the authentication process for accessing the restricted function.

1. WM can perform the authentication process for gaining the qualited information for accessing the restricted function, if the private\_function\_flag in WDD for a requested function is set to ‘1’.
2. WM requests the NBMP functions and creates the NBMP workflow.
3. WM instantiates the functions which includes the private function(s) obtained by using the auth-token (gained in step 4) and completes the creation of the workflow.

### Proposal for update current security parameter

Please update current processing descriptor for supporting the private function usage.

## 9.6.1.13 Security parameters

Table 73 defines the parameters used in Security Descriptor.

Table 73 — Security parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Definition** | **Unit** | **Type** | **Valid range** |
| name | Identifier used by Input, Output, Processing descriptors | N/A | string | N/A |
| scope | The scope of the authentication, authorization and encryption on different resources. The values for example, are: Data, Function | N/A | string | N/A |
| authentification-method | Suggested authentication, authorization and encryption methods or protocols by names. Multiple methods or protocols can be used with specific parameters  Sample methods are Access Token, JSON Web Token (JWT), Single-Sign-On (SSO) like OAuth1/2, SAML1/2, Client Certificate, Server Certificate | N/A | string | N/A |
| authority-url | Authority URL for authentication and authorization, if provided | N/A | string | N/A |
| certificate | trusted certificate, X.509 certificate, if certificate method is specified | N/A | string | N/A |
| auth-token | Access token like HMAC, key wrapped[[1]](#endnote-2) key, or security key to a KMS (key management system), if token based method is specified. | N/A | string | N/A |
| client-grant | Client grants if token-based method is specified | N/A | string | N/A |
| auth-token-expires | Pptional period of Media Source for which the authentication token is applicable. It can be ISO 8601 format | N/A | string | N/A |
| auth-token-renew | Optional token to renew the auth-token after it is expired | N/A | string | N/A |
| auth-token-rotation | Optional flag whether or not an auth-token gets rotated and renewed | N/A | boolean | N/A |
| private\_function\_flag | Optional flag indicating whether or not qualified information for accessing a restricted resource exists, such as an auth-token, in the WDD. When the flag is ‘0’, a restricted resource is required for the workflow, and an auth-token exists in the WDD. If the flag is ‘1’, a restricted resource is required but the WM should get the qualified information for accessing the restricted resource using some other means (e.g. the source-assistance-information in the client-assistance descriptor.) | N/A | boolean | N/A |
| unsigned integer = [0, (2\*53)-1] | | | | |

# Update of Dynamic resource update on NBMP

In this section, few proposals are included on the dynamic resource allocation and update.

## Dynamic resource update (proposed in m50771)

### Introduction

The current NBMP DIS describes the creation workflow using several descriptors and its parameters that are applicable to NBMP Function/Task/Workflow. At the last Geneva meeting, we discussed about the dynamic resource update. This contribution proposes to add an update to the description of the processing descriptor and its parameter for supporting dynamic resource updates during NBMP services.

### Dynamic resource update during NBMP services

In the current NBMP DIS, when a task is added or deleted, another workflow should be created. However, resource allocation or update source should be necessary for supporting a seamless media service. In this contribution, we propose a new parameter on the processing descriptor for supporting dynamic resource updates.

For example, an NBMP service can be created as below,

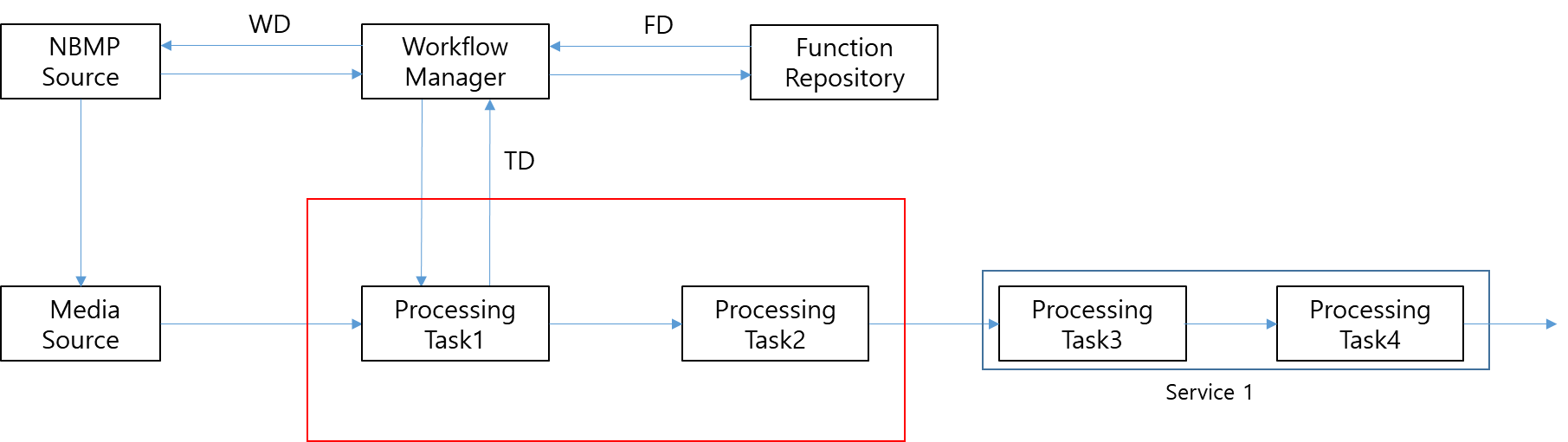


Figure 1. Example of the NBMP workflow creation (workflow 1)

If the number of service users are increased or if there is a need to support an extention of the current service, creation of the another workflow is needed as shown below;

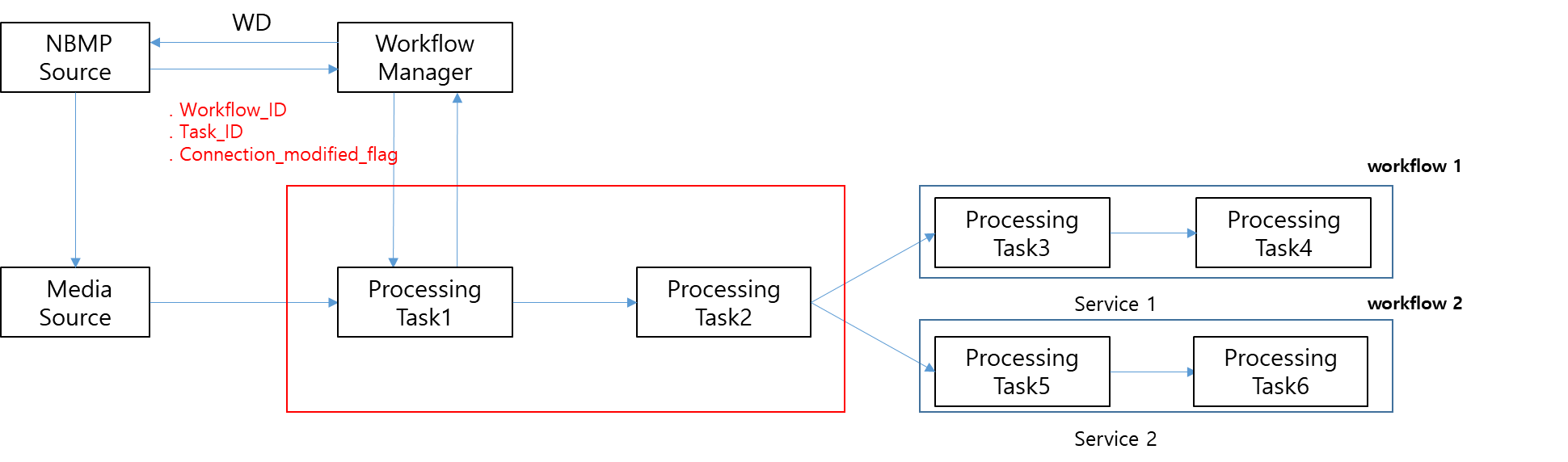


Figure 2. Creation of new workflow for exention of the service

Normally, the current NBMP system should create all of the tasks when it generates the new workflow. So if the workflow 1 consists of the tasks 1~4, then when workflow 2 is created, it should consist of tasks 5~8. However, the processing tasks 1 & 2’s output can be shared between workflow 1 and workflow as shown in figure 2. This contribution proposes the dynamic resource allocation of the current existing workflow.

In figure 2, the NBMP source can send the workflow document (WD) which includes a previous workflow ID (workflow 1’s ID) and its task information (Task ID) to the workflow manager for creating workflow 2. Then workflow manager can then create the new workflow based on a previous workflow’s task information and also new function information (keyword on processing descriptor). In addition, the NBMP source should be notified of the modification of the previous task connection. For example, it does not need to update any connection in the case of task 1, but the output connection is changed from task 3 to task 5 in the case of task 2 . So if the connection of a task needs to be updated, connection\_modified\_flag should be set to ‘1’, if not, is it set to ‘0’.

### Proposed update of current processing descriptor

## 9.6.1 Processing Descriptor parameters

Table 54 defines processing parameters, which are used as part of Processing Descriptor.

Table 54 — Processing parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Definition** | **Unit** | **Type** | **Valid range** |
| keywords | list of keywords that can be used to execute a search in function repository  Functions are described using a human-readable description and included in the Function Repository. Either Workflow Manager or NBMP Source can use function names as mentioned in Function Repository for inclusion in this field. | N/A | array of strings | N/A |
| image | pointers to implementation images | N/A | array of objects | N/A |
| start-time | the resource’s start time | As defined by ISO 8601 | string | N/A |
| identifier | specifies Function’s id | N/A | string | N/A |
| instance | Specifies identifier for one Instance of a Function. An Instance of a Function shall have unique restrictions in a Function Group. This identifier shall be unique for each Instance in the same Function Group.  Note: If a Function is used more than once in one Function Group with identical restrictions, these restrictions can be defined by one Instance of that Function. | N/A | string | N/A |
| port-name | specifies Function’s logic port name | N/A | string | N/A |
| parameters | Specifies IDs of the parameters passed to tasks for the task instantiation | N/A | array of strings | N/A |
| Workflow-id | Specifies the ID of the workflow | N/A | string | N/A |
| Task-id | Specifies ID of the task’s id in a specific workflow | N/A | string | N/A |
| Connection-modified-flag | Specifies the modification of the previous workflow connection map | boolean | N/A | N/A |
| N/A: not applicable | | | | |

## NBMP Workflow continuity (m56712)

### Introduction

This contribution proposes a use case and a solution for it. In the use case, the workflow needs to be updated to a (partially) different workflow. However, no media data must be lost during the update. The switching to the new workflow needs to occur at a certain time.

### Use case

A workflow may have been working for a while. But due to the increase in the volume of data, one or more tasks of the workflow may need to be split to cope with the load increase. In this case, no data needs to be lost due to switching to the new workflow.

In some deployments, the current tasks can be used. So there the current tasks can be used in the updated workflow.

In some application, switching to a new workflow need to happen at the exact time for collecting data purposes or billing.

### Proposed solutions

#### Workflow continuity

The Workflow API for updating workflow in the spec does not have any notion of continuity, which means that the NBMP Workflow Manager can stop the current workflow, update the workflow with the new one and start again and data may be lost during this update.

We propose to add a continuity flag to the General Descriptor. In the case of continuity = true, the NBMP Workflow manager shall switch between the two workflows without losing any data. Furthermore, the switching time may be requested to be a certain time. This is useful to know the exact time of switching for reports.

Table 12 — General Descriptor

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Type** | **Cardinality** |
| id | P | 1 |
| name | P | 1 |
| description | P | 1 |
| rank | P | 0-1 |
| mpeg-compatibility | P | 0-1 |
| published-time | P | 0-1 |
| priority | P | 0-1 |
| execution-time | P | 0-1 |
| input-ports | Array of object | 1 |
| output-ports | Array of object | 1 |
| is-group | P | 0-1 |
| continuity | P | 0-1 |
| switching\_time | P | 0-1 |
| state | P | 1 |

Table 53 — General parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Definition** | **Unit** | **Type** | **Valid range** |
| id | unique string in the scope of Repository/Workflow of the resource | N/A | string | N/A |
| name | name for identifying the resource | N/A | string | N/A |
| description | a human-readable description for the resource | N/A | string | N/A |
| rank | The rank of Function/Function Group among Functions with the same functionality.  A higher number means a higher rank. | N/A | number | unsigned integer |
| mpeg-compatibility | URN indicating the compatibility with a reference Function/Function Group defined in Annex A and B. | URI | string | N/A |
| published-time | date and time of publication of this resource | As defined by RFC3339, section 5.6 | string | N/A |
| priority | priority information for the resource | N/A | number | unsigned integer |
| port-name | unique string among all port-names of this resource defining the logic name for input or output | N/A | string | N/A |
| is-group | value ‘true’ indicates containing Descriptor describes a Function Group or Task Workflow. If the value is ‘true’, a connection-map object shall exist in this Description.  The default value is ‘false’. | N/A | boolean | N/A |
| continuity | Value ‘true’ indicates that this workflow update is a continuation of the previous workflow and no data shall be lost during switching.  The default value is ‘false’. | N/A | boolean | N/A |
| switching\_time | Switching time of the workflow to its update. If this value is in the past, then the switch should happen immediately and is updated with the exact switching time.  If a lossless transition can not occur at this time, then the switch should happen at the first possible time and is updated with the exact switching time.  The value is ignored is ‘continuity’ is ‘false’. | As defined by RFC3339, section 5.6 | string | N/A |
| state | current state of the resource in its lifecycle  The value of this parameter shall be one of the followings in a NBMP Operation’s response:   * null * instantiated * idle * running * in-error * destroyed   If this parameter is included in a request from the NBMP Source or Workflow Manager, the value of this parameter shall be one of the followings (subject to constraints in subclause ‎7.2‎7.3):   * instantiated * idle | N/A | string | N/A |

The two new parameters can be used only with Workflow Update or Task Update.

#### Partial Workflow replacement

In some cases, running a completely parallel workflow requires lots of resources and the Cloud platform prefers to maintain some of the tasks in the running workflow during the update.

Since the NBMP Client provides the new WDD, the Workflow Manager must identify the difference between the running workflow and the new WDD.

The NBMP Client can ease up this task by providing the following information:

1. The list of tasks that are changed.
2. The list of connections that are changed.

Since each task has an id in the WDD, changing a task, i.e. removing and replacing it with a new task, or introducing the new task is easy:

1. Every existing task has an id in the running workflow.
2. The location of a task in the running workflow is identified with the value of ‘instance’ in the connection map.
3. Any new task in the workflow update doesn’t have ‘instance’ set.

Consider the above, it is easy for the NBMP Workflow Manager to identify which tasks/function instances in the updated workflow are new and which ones are the old ones.

As for providing the list of connections, each workflow is defined by an array of connection-map objects. The workflow manager can compare each object in this array from the workflow update to the running workflow and identify the new or updated connection-map objects. However, to simplify the process, we use the connection id in the connection map of AMD2.

## NBMP Workflow and task Scheduling

### Scheduled-time (m56729)

#### Introduction

A Task Group is a collection of tasks or function instances that are expected to run on the same cloud node/cluster. When a set of tasks are grouped, it means that they are closer to each other than other tasks, i.e. they have a smaller distance. It is a coarser way of defining the proximity of tasks together compared to the distance parameter.

The NBMP Client may define the task groups or even the distance parameters based on the characteristics of the workflow. For example, two tasks in a workflow should be run closer since the two tasks working together is beneficial (e.g., in terms of efficiency or performance) for the workflow execution compared to the execution of the other tasks in the workflow. The use of task groups is optional, if an NBMP Client has no idea about grouping tasks.

#### Motivation for new design

When scheduling a workflow, NBMP workflow manager estimates the total computing resources (e.g. CPUs, GPUs, memories etc) and creates necessary MPEs (Media Processing Entities) from the infrastructure provider, i.e. typically the Cloud control entity, or orchestrator. The process of MPE creation is part of the known process called cloud resource provisioning. Typically, under-provisioning should be avoided to minimize any potential risks and delays caused by insufficient resources when deploying a workflow. it is common to allow some over-provisioning in practice to provide or request extra capacities in terms of resources like CPUs, memory, and storage. Such normal provisioning or over-provisioning scenarios for the whole workflow may not use the resources cost-efficiently, if NBMP can allow any delayed resource allocation; or use the available resources for partial workflows (certain tasks at certain time of period). We believe task groups are one such design option for flexible resource planning and utilization.

#### Proposed new design for task groups

The proposed new design of task groups enables new possible executions of workflow and its sub-workflows. Like the stepwise execution of individual tasks specified by the Step Descriptors, a workflow with task groups can benefit from stepwise execution or batch execution for delayed sub-workflows, plus other parallel execution modes. The enhancement can make NBMP workflow more cost-effective, as groups of tasks can be scheduled individually and the MPE resource could be used more efficiently. Over-provisioning issue as described above, could then be avoided.

In addition, NBMP TuC [2] includes one technical design about dynamic resource update on NBMP workflow (Section 2.2 “Dynamic resource update during NBMP services”) . It states that “If the number of service users are increased or if there is a need to support an extension of the current service, (the) creation of another workflow is needed”. Without interrupting the existing workflow for “a seamless media service”, we would use the concept of task groups to allow another simple but effective parallel feature to support the increased load (the number of service users are increased) and increased processing capability by the increased number of service users.

The new design can be called as task group replication. A task group can be instructed by the Workflow Manager at run-time to increase the instance number of same task group. The instances share the same input and output characteristics of the task group. It is particularly useful for the task group at the end of a workflow that provide externally facing output interfaces, e.g. “caching-server-url” for media pulling from consumers, e.g. NBMP media sinks.

#### Task Group Descriptor Design

The Task Group Descriptor (TGD) can be supported and defined in the workflow description document (WDD). When the TGD is initialized by the Workflow manager, based on proximity, for example, NBMP Source can get the TGD as part of WDD from the workflow manager. Alternatively, a TGD can be defined according to the connection conditions between tasks. In a workflow graph, the DAG edges (in the “connection-map” object) can provide hints for the boundary of the task groups. For instance, the parameter “*breakable*” determines whether or not the connected tasks can be splitable into different task groups.

Task groups in different MPEs should follow the working modes of MPEs. In WD of NBMP AMD2, MPEs can have different dynamic capabilities and tasks can be re-grouped to different MPEs. It is useful to indicate the different execution modes per task groups. Without further fragmenting or conflicting the workflow state, we can simply have 2 modes: synchronous and asynchronous. Synchronous mode is the default and tacit one if not specified. The whole tasks or task groups are scheduled, and resource planned, and executed together. Asynchronous mode enables task groups to be scheduled for execution differently. This means that the resource allocation for the task groups can be very flexible.

To support parallelism and horizontal scalability[[2]](#footnote-2) on the task group level, it can be specified as a new pair of parameters:

*replicable*: indicates the replicable capability of a task group.

*replicable-number*: Indicates the number of instances a task group can be replicated.

The replication feature can be seen as a clone of some tasks together as a group, especially when they are in one single MPE. After cloning the tasks, the newly created tasks can be added to the same task group. In order to trace those newly created tasks from existing ones, their function ID or function instance IDs need to be kept in the Descriptor with the flag to indicate they are “cloned” instances. When the task group needs to scale in from, for example, the “replicable-number” 2 down to 1, those tasks which are marked as “replicated” can be stopped and removed.

We propose to change the “task/function instance id list” in the WD [4] to a new structure where task IDs contains a property named “replicated”. Those tasks whose “replicated” parameters are true can be stopped and released when the task group scales in (e.g. the replica number decreases). Another property “replicated-from” can help to trace its original task ID.

NBMP Source can update the “replicable-number”. The workflow manager can then create or remove tasks and update the workflow’s “connection-map” accordingly.



Figure Task group before scaling



Figure Scale out Task group by task cloning via replication

When a workflow can have multiple logical sub-flows like task groups, late or delayed allocation can bring benefits like utilizing resources with different charging/pricing modes in peak times (with respect to different pricing models). In another example, it is very common in multimedia processing to utilize HW-acceleration like GPU, in particular, in machine/deep learning related tasks (e.g. the neural network training and inferencing tasks). MPEs with GPU requirements are relatively expensive. It would be good if a workflow with GPU requirements can finish the tasks as fast as possible. A workflow supporting live ingestion may, however, would be in running state for a long time. But not all tasks in the workflow require GPU-powered MPEs. It would be economically efficient to group tasks and delay the execution of some tasks by groups. When task groups can be scheduled independently, those associated GPU-powered MPEs could be scheduled at the right time when the input data for the relevant task groups becomes available, without pre-occupying those expensive resources during the whole life cycle of the workflow. We suggest using task groups to represent the sub-workflows and to allow flexible scheduling and execution like we can do with individual tasks. This would enable leveraging the step mode workflows (which is already supported) in a more efficient manner with task groups. We propose to add datetime parameters to postpone certain task groups from execution.

#### Proposed changes

The WD of NBMP AMD2 [4] defines the Task Group as follows. The existing text in the WD are greyed and the other texts in black color to show the proposed changes.

### Task Group

A Task Group is a logical group of tasks that are expected to be deployed on the MPEs as close as possible, possibly on the same MPE.

A Workflow Description with a collection of tasks may assign different subsets of tasks to one or more Task Groups as shown in Table X.

Table X — Relationship of Task/Function instance IDs and task group IDs

|  |  |
| --- | --- |
| Task Group Ids | Task/Function Instance Id list |
| G0 | Id00, Id01, ..., Id0m0 |
| G1 | Id10, Id11, ..., Id1m1 |
| … | … |
| G(k-1) | Id(k-1)0, Id(k-1)1, …, Id(k-1)m(k-1) |

In the above table:

* K is the number of Task Groups;
* mi is the number of Tasks in Group Gi;
* Gi is the id of the Task Group i, and
* Idij is the id of Task or Function instance j in Task Group i.

A Task Group is identified with a unique identifier that is unique between Task Groups and Tasks and Function instances IDs in WDD and FDD. When defined in WDD, the list contains the task IDs with the property “replicated” to indicate whether or not the task is replicated from another task; and the property “replicated-from” to indicate the source ID of the newly replicated task ID.

Task Group Descriptor describes task groups in more detail.

8. NBMP Descriptors

8.22 Task Group Descriptor

This descriptor provides high-level details about the group-based task processing of a workflow. Table XX defines the parameters used in this descriptor:

(Note: this table contains the parameter descriptions following the structure in NBMP Section 9: NBMP Parameters. We can provide correct structure for Section 8 NBMP Descriptors)

Table XX Task group object

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Unit** | **Type** |
| Id | Group id | identifier | String |
| Tasks | (see previous Table X) | Task identifier | Array of String |
| Mode | Specifies task group mode. There are 2 modes: 1) synchronous mode; 2) asynchronous mode for flexible task group-based execution and scheduling. The default is synchronous mode. | N/A | String |
| Replicable | Specifies the scaling capability applicable to asynchronous mode task group only. Default value is False. | N/A | Boolean |
| Replicable-number | Specifies the number of replicas that can be run concurrently, when the replicable parameter is True. Default value is 0 | Number of replicas | Number |
| Scheduled-time | Specifies exact scheduled execution time of the task group. When the upstream tasks or task groups fail or does not complete by the scheduled time, this task group shall not run and report the error. | As defined by ISO 8601 (e.g. 2021-03-24) | String |
| Delayed-time | Specifies the relative execution time to outputs from upstream tasks or task groups in batch/stepwise execution mode. | seconds | Number |

### NBMP Workflow scheduling (m57293)

#### Use case

In some use cases, the workflow can be run in parts, task by task, or a group of tasks at each time. In such an application, real-time processing is not a requirement and for reasons such as limited computational resources allocated for the workflow or to avoid peak traffic time on the cloud, a workflow may be needed to be scheduled.

#### Schedule schemes

We consider the following schemes for each task:

1. By order: each task is run once for its entire inputs and then the next task is run.
2. By duration: each task in the workflow is run such that one duration of input/output is consumed/produced, e.g. 1 min of inputs, or 5 mins of output(s).
3. By event: each task is run with a start event and stops with a stop event.
4. By timeslot: each task starts according to a specific schedule.

The above schemes can be run for a group of tasks (or a task group). Schemes 2-4 can be run for the entire workflow.

#### By order

The Workflow Manager needs to run each task at the time. Therefore, it has to start from tasks that have the input(s) of the workflow, generate the outputs, buffer them and then start the next tasks in a moving wave fashion toward the outputs. There are two possibilities:

1. The NBMP Client defines the order of tasks to perform
2. The Workflow Manager deducts the order.

In both cases, the Workflow Manager needs to reconfigure the workflow by adding buffer/storage for each output of the task. The Workflow Manager either needs to have the information about the maximum size of these buffers/storages, or each input size is provided so that the Workflow Manager estimates the size of buffer/storage.

Note that the ‘complete’ input which was previously proposed is useful here since every task receiving the ‘true’ flag at its ‘complete’ input knows that its task is completed and can generate an event for the Workflow Manager that it is the time of the next task.

##### Requirement

* From NBMP Client
  + The scheme mode
  + Optionally the order of tasks
  + Maximum size of storage for each output in workflow or maximum size of the inputs.
* From NBMP Workflow Manager
  + Capability signaling of the supported:
    - If it supports this mode
    - If it can derive the order
    - If It can derive the required buffer sizes from inputs sizes
  + Acknowledge if it can run the request

#### By duration

The Workflow Manager needs to run each task for a specific duration of input or output. Therefore, it has to start from tasks that have the input(s) of the workflow, generate the outputs for that duration, buffer them and then start the next tasks in a moving wave fashion toward the outputs. There are two possibilities:

1. The NBMP Client defines the order of tasks to perform
2. The Workflow Manager deducts the order.

In both cases, the Workflow Manager needs to reconfigure the workflow by adding buffer/storage for each output of the task. The Workflow Manager either needs to have the information about the maximum size of these buffers/storages for that specific duration, or each input size is provided, so that the Workflow Manager estimates the size of buffer/storage.

##### Requirement

* From NBMP Client
  + The scheme mode
  + The duration of input(s) or the required duration of the output(s).
  + The order of tasks if needed/provided
  + Maximum size of storage for each output in workflow or maximum size of the inputs.
* From NBMP Workflow Manager
  + Capability signaling of the supported:
    - If it supports this mode
    - If it can derive the order
    - If it can derive the required buffer sizes from inputs sizes
    - If it can derive the duration of inputs required for the generation of the given duration of the output(s).
  + Acknowledge if it can run the request

#### By event

The start and stop of each task are triggered with an event. The events are issued by the Workflow Manager. Since any task has the ‘run’ state, the Workflow Manager can start the task by changing its state to ‘run’. To pause a task, the Workflow Manager can change the tasks state to ‘idle’. However, ‘idle’ does not distinguish between the lack of input vs the request by Workflow Manager to stay as ‘idle’. Therefore, we propose adding a new state, ‘paused, in which the task can be changed from ‘run’ to stay idle even if the inputs are available.

The Workflow Manager either can schedule events based on its own internal logic, or it can receive the events from the NBMP Client. Since also in this scheme, the output buffers/storages of each task should be managed. There are two possibilities:

1. The NBMP Client defines the maximum size of intermediate buffers/storages if the events are driven by the NBMP Client.
2. The Workflow Manager deducts the size if it schedules its own events.

##### Requirement

* From NBMP Client
  + The scheme mode
  + If the event is driven by the NBMP Client or a custom signal for the workflow Manager to start its own event scheme.
  + Maximum size of storage for each output in workflow if it is driven by the NBMP Client.
* From NBMP Workflow Manager
  + Capability signaling of the supported:
    - If it supports this mode
    - If it supports, the Workflow driven events
  + Acknowledge if it can run the request
* From NBMP task
  + Support of the ‘paused’ state
  + Support of changing the state between ‘run’ and ‘paused’

#### By timeslot

The NBMP Client provides a schedule for each task and the Workflow Manager run each task according to the given timeslot.

Since intermediate buffers/storage needs to be allocated, there is one possibility:

1. The NBMP Client defines the size of buffers.

##### Requirement

* From NBMP Client
  + The scheme mode
  + The scheduling mode of each task
    - Explicit: the schedule has the exact times
    - Indexed: A start, end, increment, duration scheme (like a for loop).
  + Maximum size of storage for each output in workflow or maximum size of the inputs.
* From NBMP Workflow Manager
  + Capability signaling of the supported:
    - If it supports this mode
    - If it supports the schedule mode
  + Acknowledge if it can run the request

#### Proposed solution

##### Schedule Descriptor

We define a schedule descriptor with the following parameters:

Table X1 — Scale Descriptor

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Type** | **Cardinality** |
| id | P | 1 |
| description | P | 0-1 |
| schedule-type | P | 0-1 |
| order | Array of string | 0-1 |
| duration | P | 0-1 |
| number-of-segments | P | 0-1 |
| io-flag | P | 0-1 |
| run-mode | P | 0-1 |
| timeslot | O | 0-1 |
| status | P | 1 |

Table X2 — time-slot object

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Type** | **Cardinality** |
| mode | P | 1 |
| explicit | Array of objects | 0-1 |
| start | P | 0-1 |
| end | P | 0-1 |
| increment | P | 0-1 |
| duration | P | 0-1 |
| timescale | P | 0-1 |

Table X3 — explicit object

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Type** | **Cardinality** |
| start | P | 1 |
| end | P | 1 |

Table X4 — scale parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Definition** | **Unit** | **Type** | **Valid range** |
| id | unique string indicating the schedule request in the scope of Workflow | N/A | string | N/A |
| description | a human-readable description for the schedule request | N/A | string | N/A |
| schedule-type | type of schedule request, one of the following values:  ‘order’: with a specific order  ‘duration’: with a specific order and for a specific duration  ‘segment’: for a specific number of segments  ‘event’: event-driven  ‘timeslot: with a given timeslot schedule  The default value is ‘order’. | N/A | string | N/A |
| order | The list of task ids in the order of execution for ‘schedule-type’=’order’ or schedule-type’=’duration’ or schedule-type’=’segment’ | N/A | string | N/A |
| duration | The duration of inputs/outputs in the unit of the timescale for schedule-type’=’duration’ | ticks | unsigned integer | N/A |
| timescale | The number of ticks per second used for the duration unit for schedule-type’=’duration’ | N/A | unsigned integer | N/A |
| number-of-segments | The number of inputs/outputs segments to be processed for schedule-type’=’segment’ | N/A | unsigned integer | nonzero |
| io-flag | If ‘TRUE’, then the duration or number of steps are for the outputs, otherwise they are for the inputs, for schedule-type’=’duration’ or ‘schedule-type’=’segment’  The default value is ‘FALSE’. | N/A | boolean | N/A |
| run-mode | The state change when ‘schedule-type’=’event’  ‘run’: change to the ‘run’ state.  ‘pause’: change to the ‘pause’ state | N/A | string | N/A |
| status | Status of the schedule request, one of the following values:  ‘capabilities’: request the capabilities  ‘consider’: investigate whether such schedule is possible to accommodate  ’request’: request scheduling  ‘passed’: accommodated/possible to accommodate  ‘failed’: failed/not possible  The default value is ‘failed’ | N/A | string | N/A |
| mode | timeslot mode:  ‘explicit’: explicit schedule with specific start and end time  ‘indexed’: implicit by defining start, end, increments, and duration.  The default value is ‘explicit’. | N/A | string | N/A |
| start | Start time of the timeslot for the task in ‘run’ mode. |  |  |  |
| end | end time of the timeslot for the task in ‘run’ mode. |  |  |  |
| increment | The increments of time to the next start time in the scale of timescale | N/A | unsigned integer | N/A |
| duration | The duration of time for this increment, in the scale of timescale | N/A | unsigned integer | N/A |

##### Use of the Schedule descriptor

The NBMP Client can include scale descriptor in a WDD update call for the following:

* Get capabilities of Workflow Manager by including it in WDD and ‘status’=’capabilities’
* Scheduling the Workflow by including it in WDD
* Scheduling a task by including it in a TDD
* Scheduling a group of tasks by including the descriptor in a Task group object
* Consider scheduling any of the above by requesting with ‘status’=’consider’ to see if the workflow can manage the schedule if it is requested.

### Notes of MPEG#134

Discussion on the contributions:

<http://mpegx.int-evry.fr/software/MPEG/Systems/NBMP/Spec-Development/-/issues/66>

<http://mpegx.int-evry.fr/software/MPEG/Systems/NBMP/Spec-Development/-/issues/70>

* For the design of "replication", it is applied only to the tasks at runtime. We intend to bring the scaling in/out concept to the Tasks/Task Groups/MPEs. It may require support from available NBMP elements/functions such as Step descriptor, splitter, and merger functions.
* The usecase of being able to schedule task/task groups with more sophisticated schedules (every 10 mins, at time T1, T2, ..., or an event driven) should be studied.

### Notes of MPEG#135 (on m57293)

The discussion can be found at: <http://mpegx.int-evry.fr/software/MPEG/Systems/NBMP/Spec-Development/-/issues/85>

Summary:

1. use corn instead of explicit/implicit scheduling
2. add a note that the paused state might be too complex and need to be discussed.

Details:

* Opinion 1 :
* NBMP Task lifecycle defines transitions like “running” and “idle”. The transition from “running” state to “idle” state is done by “onStop” or “onCompletion”. It is not defined how the transition occurs but the implementation may use events to inform WM. So the WM implementation would assume that the state changes are known by the WM via any mechanisms provided, according to 10.3 Task Requirements (NBMP FDIS). So at any given moment, WM knows the states of workflows and their tasks. For those tasks not in “running” states, there’s no need to add buffer/storage.
* The “paused” state is unnecessary but the “onPause” transition can be introduced, instead. So a task/workflow can change its state from “running” to “idle” with the “onPause” transition (changes can be added to the state diagrams and 10.3 Task Requirements). Alternatively, a new state “scheduled/hibernation” may be more useful than “paused” to indicate the situation: the workflow is scheduled but resources like MPEs shall not be allocated yet, unless their states change to “instantiated”. Other states can transit to the new state and resources can be freed for other purposes, temporarily.
* The “time-slot” design is complex. Please consider the well-known cron job schedule format: <https://en.wikipedia.org/wiki/Cron>
* The “order” is also questionable, as the workflow graph defines the data flow order which is somehow correspondent to the execution order, e.g.
  + A -> B -> C (graph DAG)
  + But the order may define “B, C, A”. So the conflict appears.
* In general, the contribution addressed the needs for a schedule/scheduler descriptor and we agree to add it to the TuC for further studies.
* Adding new state like "paused" would bring more complex situations for stateless and stateful tasks. For the time being, the suggestion (option 1) would be more preferable.
* Opinion 2 (author) :
  + The above approach only works for stateless tasks since idle state does not keep the state of the tasks (the internal memory and the result of processing previous data). For stateful tasks, we need that a task to be paused but maintain its own internal state. So we either 1) make the idle stateful, i.e. add the requirement that a task in idle should maintain its state. In this case, when the task goes from running to idle with "onPause" transition, then the task shall maintain its state. 2) or add a onpause state.
  + The purpose of the "paused" state is to maintain the state of a previously running task. So a hibernated task still has its memory, i.e. maintain it state.
  + A scheduled task, could be a task with either idle or paused state and with a schedule to change its state to running. So to me, scheduled is a property and not a state.
  + The order is for when there are existing parallel tasks, i.e. based on the graph DAG you can schedule either one first. I agree that the order shall not contradict the graph DAG requirements, i.e. if task A's output is connected to task B's input, then A comes before B in "order", as shown in the above example.

# Exploration on split rendering with NBMP

## Introduction

NBMP is designed to enable controlling media processing workflows in the cloud. With the emergence of 5G and Edge Computing, offloading media processing to the mobile edge enables realizing advanced XR and immersive media use cases on mobile handheld devices in a way that would otherwise not be possible with traditional client/server architectures or even with classical cloud platforms.

## use-case (as defined N19030)

With the emergence of cloud gaming services, split rendering has taken a different shape. In order to support real-time gaming on devices with limited graphics processing power, such as devices operating on battery, complex rendering operations of the game scene are performed in the cloud. This operation is denoted as scene baking. In the extreme case, the game scene is rendered completely and the result is encoded as a 2D video and streamed to the client. The client may perform some pose correction operations or may just render the received 2D video.

The following figure shows how split rendering is used for cloud gaming:



The client capabilities differ from client to client and may even change dynamically, e.g. based on current network connectivity or power level. The client is able to determine its current processing capabilities dynamically and should be able to control what gets offloaded and what is performed on the device.

It is also possible that the workflow split is flexible enough to be allocated between source, different MPEs and the sink.

## Requirements (as defined N19030)

We propose to consider the following requirements for the design and development of phase 2 of NBMP

* It shall be possible for the NBMP Sink to influence the NBMP workflow:
  + To control the workload split between the sink and the network
  + To dynamically adjust the workload split based on changes in client status and conditions
* It shall be possible to support streaming of media and metadata from the network to the sink in different formats that are appropriate to the different workload sharing strategies

## Potential solutions based on m53671, m53673, m53675)

### Splitting workflow

An NBMP Workflow consists of one or more tasks, as shown in Figure 3.1.

Input 1

Input 2

Output 1

Output 2

Output 3

Figure 3.1 — Example of Workflow

In split-rendering, some of the tasks of a workflow may be implemented in the Media Source and/or the Media Sink. Figure 3.2 shows an example of the split-rendering of Figure 3.1.

Figure 3.2 — Example of Workflow Directed Acyclic Graph (DAG)

Input 1

Input 2

Output 1

Output 2

Output 3

Source

Sink

Cloud

In Figure 3.2,

* Tasks 1 and 2 are implemented on the Source device/platform.
* Tasks 3 and 4 are implemented on the Cloud/Edge network, and
* Tasks 5,6, 7, and 8 are implemented on the Sink device/platform

### Workflow Reconfiguration

Depending on the workflow and source/sink capabilities, sometimes it would be more efficient if a workflow is reconfigured by replacing a part of the workflow or adding intermediate tasks in the workflow. The reason can be that a source or sink may have built-in functions that are more efficient, so the delivery format to them might be different than the workflow output format, or the battery constraint in a device may force the workflow to be replaced with a different workflow. Figure 3.3 and Figure 3.4 show two examples of such cases.

Figure 3.3 — Changing a workflow by adding Tasks 6 and 7 for reducing the bandwidth

Input 1

Output 2

Output 3

Sink

Output 2

Output 3

Sink

In Figure 3.3, the initial workflow is split between the cloud and the Sink device. But as the sink device connection bandwidth reduces, the workflow can work with the required throughput. The Workflow Manager updates the workflow by adding Task 6 and Task 7. Task 7 is a built-in decompression function in the Sink device. Using Task 6 as the compression function on the edge network, the bandwidth between Task 6 and Task 7 is reduced enough to fit in the available bandwidth to the device.

Figure 3.4 — Changing a workflow by moving processing to the edge and adding Tasks 6-9

Input 1

Output 2

Output 3

Sink

Output 2

Output 3

Sink

In Figure 3.4, an example of reconfiguring workflow is shown for the use-case in which the sink device is low in battery. Since Tasks 3-5 require significant processing power, to save the sink device battery, the workflow is reconfigured by moving Task 3-5 to the cloud and adding compression Tasks 6 and 8 so that the used bandwidth to deliver the content to the sink device is reasonable. Tasks 7 and 9 are build-in efficient decompression functions that decompress the input streams and render the results.

#### Signaling reconfigured workflows (m54496)

If a workflow is reconfigured, the new tasks added to the workflow are not essential in terms of the workflow description. They are added for the purpose of running the workflow due to the current conditions of sink/source/network resources. An example of such tasks are Task 6 and 7 in Figure 3 and Tasks 6, 7, 8, 9 in Figure 4.

In the modified workflow, such tasks can be marked as ‘virtual’ tasks. A workflow can operate seamlessly even if all its virtual tasks are removed. Signaling the virtual tasks in a modified workflow is beneficial, as it allows the NBMP Source or Workflow Manager to identify those tasks in the workflow and remove them if necessary, in future operations.

For signaling those tasks, we propose adding a parameter in the General Descriptor of the task:

Table 6 — General Descriptor

|  |  |  |
| --- | --- | --- |
| **Parameter Name** | **Type** | **Cardinality** |
| id | P | 1 |
| name | P | 1 |
| description | P | 1 |
| rank | P | 0-1 |
| mpeg-compatibility | P | 0-1 |
| published-time | P | 0-1 |
| priority | P | 0-1 |
| execution-time | P | 0-1 |
| input-ports | Array of object | 1 |
| output-ports | Array of object | 1 |
| is-group | P | 0-1 |
| virtual | P | 0-1 |
| state | P | 1 |

Table 7 — virtual parameter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Definition** | **Unit** | **Type** | **Valid range** |
| virtual | A number indicating all tasks in the workflow that may be removed with this task resulting to a functionally equivalent workflow, i.e. the workflow can be replaced with the resulting workflow. | N/A | number | unsigned integer |

The virtual parameter is only applicable for TDD.

### Source and Sink System Events

The source and sink descriptions allow listing the source/sink system events. These events are typically are:

1. Reduced processing power from a certain threshold
2. Reduced available memory from a certain threshold
3. Reduced available disk space from a certain threshold
4. Reduced battery level from a certain threshold

The workflow manager can set up notifications for all or a subset of the above system events using the notification descriptor. If such an event occurs, the workflow manager will receive the notification and can reconfigure the workflow to address the event.

### Split rendering between different network element or different cloud nodes

The above method can be used to split the workflow implementation among network elements, edge computing resources, between different nodes of a cloud platform or between multiple-cloud platforms.

# Signalling push/pull and other characteristics of inputs and outputs

## Proposed Improvements to Input and Output Descriptor (m55963)

### Introduction

The contribution presents an improved Input and Output Descriptor for efficient media/metadata processing, with respect to the transport characteristics of media data. Current I/O Descriptors (8.3 and 8.4) in NBMP FDIS [1] already contain some parameters covering some aspects such as accessibility (e.g. "\*-format", "codec-type", "protocol", and "caching-server-url"), capacity ("buffersize") and speed (e.g. "throughput"). One important character we believe current specification is missing is about the availability, that is, the indication about the access to the data when they are available; and how they can be fetched; and their potential retention duration for consuming. This type of characteristics is important particularly to the Workflow Manager to orchestrate the execution of the workflow (tasks) and allows more effective and efficient resource planning for different media processing modes such as real-time data streaming and batch processing. Different processing modes require different policies and requirements on data availability, and different deployment or setup for media and metadata transportation.

### [NBMP] New parameters to NBMP Input/Output Descriptor

The location of media and metadata is defined as the parameter "caching-server-url" in NBMP specification [1] as "URL of the server where the media will be sent from or the location from where the media can be fetched from". Given the versatility of NBMP use cases, the ways of media and metadata transportations shall be also versatile for both batch and stream processing. Media and metadata should be able to be pushed from upstream sources or tasks to downstream tasks or sinks where upstreaming tasks are the driver of data flow. Pulling should be able to work vice versa where downstream tasks are the driver of data flow.

Table 1 shows the new parameters proposed to NBMP Input/Output Descriptor. Each task can have one or multiple Inputs and Outputs. Each input/output defines one type of media streams (data).

Table 13 New parameters to Section 8.3 Input Descriptor and 8.4 Output Descriptor

Table 13 — Input Descriptor

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Cardinality** |
| media-parameters | Array of object | 0-1\* |
| metadata-parameters | Array of object | 0-1\* |
| \* This Descriptor shall contain at least one of the above objects. | | |

Input Descriptor consists of two arrays of objects: one for the media inputs and one for metadata inputs, as shown in Table 14 and Table 15 respectively.

Table 14 — Input media-parameters objects

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Cardinality** |
| stream-id | P | 1 |
| name | P | 1 |
| keywords | P | 1 |
| mime-type | P | 1 |
| video-format | P | 0-1 |
| audio-format | P | 0-1 |
| image-format | P | 0-1 |
| codec-type | P | 0-1 |
| protocol | P | 1 |
| throughput | P | 0-1 |
| buffersize | P | 0-1 |
| caching-server-url | P | 1 |
| mode | P | 0-1 |
| availability | O | 0-1 |
| priority | P | 0-1 |

The "availability" object defines further parameters about how and when the data can be accessible. It defined in Table XX.

Table XX— availability object

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Cardinality** |
| live | P | 0-1 |
| persistence | P | 0-1 |
| duration | P | 0-1 |
| latency | P | 0-1 |

Table 15 — Input metadata-parameters objects

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Cardinality** |
| stream-id | P | 1 |
| name | P | 1 |
| keywords | P | 1 |
| mime-type | P | 1 |
| codec-type | P | 0-1 |
| protocol | P | 1 |
| max-size | P | 0-1 |
| min-interval | P | 0-1 |
| caching-server-url | P | 0-1 |
| scheme-uri | P | 0-1 |
| mode | P | 0-1 |
| availability | O | 0-1 |
| priority | P | 0-1 |

Table 17 — Output media-parameters objects

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Cardinality** |
| stream-id | P | 1 |
| name | P | 1 |
| keywords | P | 1 |
| mime-type | P | 1 |
| video-format | P | 0-1 |
| audio-format | P | 0-1 |
| image-format | P | 0-1 |
| codec-type | P | 0-1 |
| protocol | P | 1 |
| throughput | P | 0-1 |
| buffersize | P | 0-1 |
| caching-server-url | P | 1 |
| mode | P | 0-1 |
| availability | O | 0-1 |
| priority | P | 0-1 |

Table 18 — Output metadata parameters objects

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Cardinality** |
| stream-id | P | 1 |
| name | P | 1 |
| keywords | P | 1 |
| mime-type | P | 1 |
| codec-type | P | 0-1 |
| protocol | P | 0-1 |
| max-size | P | 0-1 |
| min-interval | P | 0-1 |
| caching-server-url | P | 1 |
| scheme-uri | P | 0-1 |
| mode | P | 0-1 |
| availability | O | 0-1 |
| priority | P | 0-1 |

And changes to the Section 9 NBMP Parameters.

Table 54 — Input and Output parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Definition** | **Unit** | **Type** | **Valid range** |
| stream-id | unique identifier, with the scope of Function or Task or Workflow, to identify the media or metadata stream  For Functions, it is defined in Function Descriptor. For Tasks, it is assigned by Workflow Manager. For Workflows, it is assigned by NBMP Source. | N/A | string | N/A |
| …  (omitted for space saving) | … | … | … | … |
| mode | The value can be either "pull” or "push”. It should work with the "protocol” parameter together in NBMP Input/Output descriptors | N/A | string | N/A |
| priority | The value is an unsigned integer to indicate if the particular instance of the data format is made available or not depending on the resource availability while executing the workflow. The default value is 1 as essential, and any value greater than 1 shall be less essential. | Unsigned integer | number | 1..100 |
| live | A Boolean flag which indicates whether the content being streamed lively by the upstream entity (Media Source or task). Default value is false. | N/A | boolean | true or false |
| persistence | a Boolean flag which indicates whether the content shall be available for long term (e.g., even after the live streaming has ended). Default value is false. | N/A | boolean | true or false |
| duration | Time in milliseconds which indicates the duration for which the data is available. This value is expected to be present only if persistence flag is True. Default value is 0. | milliseconds | number | Unsigned integer |
| latency | Time in milliseconds which indicates additional delay in milliseconds. Default value is 0. | milliseconds | number | Unsigned integer |

Note: The same description will go to Table 55 Output parameters too.

### JSON Schema

The added schema would be added under the "properties" of the "input" and "output" objects.

|  |
| --- |
| ...  "mode": {  "type": "string",  "enum": ["pull", "push"],  "default": "pull"  },  "priority": {  "type": "integer",  "minimum": 1,  "maximum": 100,  "default": 1  },  "availability": {  "type": "object",  "properties": {  "live": {  "type": "boolean",  "defaut": false  },  "persistence": {  "type": "boolean",  "default": false  },  "duration": {  "type": "number",  "default": 0  },  "latency": {  "type": "number",  "default": 0  }  }  }  ... |

### **References**

1. W19062, "Text of ISO/IEC FDIS 23090-8 Network-based Media Processing", Brussels, Belgium, January 2020
2. W19015, "WD of Implementation Guidelines on Network-based Media Processing", MPEG 129 Meeting, Brussels, Belgium, January 2020

### Discussion at MPEG#133

The following is a summary of the discussion occurred at the MPEG#133 meeting. For more details, please see <http://mpegx.int-evry.fr/software/MPEG/Systems/NBMP/Spec-Development/-/issues/61>.

#### Comment 1

I agree with the shortcoming of the pull mode of the outputs. Thanks for finding this flaw.

For discussion, let's assume there are two tasks T1 and T2, the output of T1 is offered as pull-based. So the T2's input should pull the data from T1's output:

T1 output---------------->T2 input

I have the following questions for understanding the proposal better:

1. Priority: I could not understand how it is used. How does T2's input use the value of priority? what does "essential" mean? if some data can be ignored or processed late, then how late that would be?
2. Persistency: What does "a long time" mean? I believe if we want to define a lifetime for data, wouldn't be easier to define it with a number (i.e. number of seconds the data would remain after becoming available)?
3. live: if the cache-server-url exists, does it mean that the data is not streamed? i.e does the existence of cache-server-url cover live =false flag?
4. latency: it is not clear what additional delay is.
5. Why do we need these parameters for the input descriptor? For output, I understand that it defines the availability and lifetime of data. For input, if we add these parameters, wouldn't the definitions be different? do these parameters show how often the input can be updated (so is that a push characteristic and not a pull)?

Also brainstorming:

In the case of data becoming available at T1's output for a pull, I imagine that data is provided in fragments, i.e with having cache-server-url, it means the data is not continuously streamed but is becoming available as fragments/batches. So then we can define the following parameters for each data fragment/batch:

- cache-server-url (already exists): if this parameter has a value, it means the data is becoming available with pull.

- minimum update: the minimum interval between two consecutive available data fragments in msec.

- maximum lifetime (right now defined as duration): the maximum time for each data fragment to be available in msec.

with the minimum update, T2 knows how often it has to make requests to pull the data from T1, i.e. defines the minimum frequency that new data becomes available. With a maximum lifetime/duration, it knows how long the data will be available. Basically these two parameters define the maximum and minimum frequency for T2 to pull data fragments from T1 without losing any data.

Is there anything else is missing for covering the usecase?

#### Response to comment 1

Thanks for those questions. let us discuss them one by one.

*Priority: I could not understand how it is used. How does T2's input use the value of priority? what does "essential" mean? if some data can be ignored or processed late, then how late that would be?*

It is the job of WM to manage the connection between T1's output port and T2's input port. Assuming T1 has 2 outputs ("port1" and priority=1 for HEVC/5MB and "port2" (priority=2) for HEVC/2MB, WM can configure T2 to be connected to T1's "port1" by default. When the situation changes (we don't know or specify the situation yet, but WM makes the decision), WM can pause the workflow or T1/T2 by their states and re-configures T2 to T1's port2. The level of "essential" can be further defined. We are open for new proposals, e.g. the level of priority, etc.

*Persistency: What does "a long time" mean? I believe if we want to define a lifetime for data, wouldn't be easier to define it with a number (i.e. number of seconds the data would remain after becoming available)?*

Exactly. The next parameter "duration" serves the purpose when "persistency" is set to True. Of course, it works too if we combine "persistency" and "duration" into one, e.g. to "persistent-duration". when its value is zero (0), it is equivalent to "persistency" is set to False.

*live: if the cache-server-url exists, does it mean that the data is not streamed? i.e does the existence of cache-server-url cover live =false flag?*

Technically, the URL can point to anything, both live (e.g. a RTSP camera) or non-live (e.g. a network-hosted media file). We believe current "cache-server-url" is insufficient.

*latency: it is not clear what additional delay is.*

It is designed purposefully. Any delay can be added by the WM when realizing the workflow by linking tasks together. Default value is zero (0).

*Why do we need these parameters for the input descriptor?*

Good point. We believe following parameters are still valid to Input:

* mode: it defines which mode T2 runs (obtains the data)
* priority: it defines correspondent/different input port options for different data generated during the run-time (managed by the WM)
* live: it may influence the ways how T2 consumes the data.
* delay: it defines the timing skew/offset caused by possible internal media server. The value is typically assigned by the WM.

The "persistency/duration" indeed is more meaningful for outputs. we are open to brainstorm them wrt. inputs.

#### Agreement during the call

1. Mode is needed to clarify whether the output being pulled or pushed.
2. Priority: It is not clear how to use the number. The change or essentiality of an input/output should be described in the function description or the Task’s status of the output as part of the output (like bitrate reduction).
3. Presitency is not needed and the life time of the output data can be described with duration.
4. Live is not clear and seems not needed. Whether the data is produced concurrently or in past by the sending task is no concern of the receiving tasks.

## On improvements to NBMP input/output descriptors (m 56711)

### Introduction

This contribution proposes few changes to the TuC section on “Improvements to Input and Output Descriptor”.

### Proposed improvements

#### mode

It is needed and in fact, the NBMP spec needs to be fixed by adding this flag. The flag can be set to “pull” or “push”. However, the definition of pull or push is not clear. We suggest the following improvement:

|  |  |  |  |
| --- | --- | --- | --- |
| mode | The value can be either "pull” or "push”. It should work with the "protocol” parameter together in NBMP Input/Output descriptors.  For input, if this value is set to “push”, it means that the data should be pushed to this input. Otherwise, the data is pulled by this input.  For output, if this value is set to “push”, it means that the data is pushed from this output. Otherwise, the data is pulled from this output. | N/A | string |

We review the other flags for each case.

#### mode = push for input

If an input is pushed, then the data is pushed to this input. In this case, it seems the following parameters seems meaningless:

* Priority: the definition is not clear.
* Live: What does live streaming mean in the case of a push input? Does this input accept live streaming? Then the protocol can define the followings:
  + Maximum frequency of allowed pushes (minimum time between two consecutive pushes)
  + Maximum allowed order of ‘out of order’ pushes
  + Maximum number of allowed parallel pushes
* Persistence: since the content is being pushed to the input, the availability is meaningless. If it means that whether this input is essential or not, then it should be defined as part of the input descriptor as essentiality.
* Duration: same as above since the data is being pushed.
* Latency: same as above.

#### mode = pull for input

If an input is pulled, then the data is pulled by this input. In this case,

* Priority: the definition is not clear. If it means that this data is not essential, then it should be a parameter for all inputs.
* Live: The definition shall be changed that the input pulls the content using live streaming. However, live streaming is not defined. The protocol can define The followings:
  + Maximum frequency of pulls (minimum time between two consecutive pulls)
  + Maximum order of ‘out of order’ pulls
  + Maximum number of parallel pulls
* Persistence: doesn’t seem to have meaning for a pull input. Please look at the parameter duration for the requirement that data must be available for a long time.
* Duration: if the data’s lifetime/age is meant, then this definition should be changed to the maximum time needed by this input for pulling data as soon as a pull-segment of data become available. This is a protocol parameter.
* Latency: not clear

#### mode = push for output

If the output is pushed, then the data is pushed by this output. In this case, it seems the following parameters seems meaningless:

* Priority: the definition is not clear. If the meaning is that the output may or may not become available, then the output should be marked as optional.
* Live: The meaning of live streaming varies depending on the protocol. The protocol can define the following parameters:
  + Maximum frequency of pushes (minimum time between two consecutive pushes)
  + Maximum order of ‘out of order’ pushes
  + Maximum number of parallel pushes
* Persistence: since the content is being pushed out, the availability of data can be indicated and the current definition is not meaningful.
* Duration: since the data is being pushed out, availability duration doesn’t have any meaning. If this is the maximum delay of pushing data, then it is a protocol parameter.
* Latency: same as above

#### mode = pull for output

If the output is pulled, then the data is pulled from this output. In this case,

* Priority: the definition is not clear. If the meaning is that the output may or may not become available, then the output should be marked as optional.
* Live: The meaning of live streaming varies depending on the protocol. A protocol can define the following parameters:
  + Maximum frequency of allowed pulls (minimum time between two consecutive pulls)
  + Maximum order of allowed ‘out of order’ pulls
  + Maximum number of allowed parallel pulls
* Persistence: doesn’t seem to have meaning for a pull input. Please look at the duration/availability if the data must be available for a long time.
* Latency: not clear

The duration makes sense if it is the maximum duration in which the data is available to be pulled once it becomes available. But then again this is a protocol’s characteristic.

### Proposed improvements

1. Add definition of pull and push to the parameter mode. Or change the parameter’s name to push and make it a Boolean flag.
2. Define the following parameters instead of priority, live, persistence, and latency and change the definition of duration:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mode | The value can be either "pull” or "push”. It should work with the "protocol” parameter together in NBMP Input/Output descriptors.  mode of transfer to/from input/output. The exact meaning of this value is defined by the “protocol” parameter.  Two values are defined: “push” and “pull”.  For input, if this value is set to “push”, it means that the data should be pushed to this input. Otherwise, the data is pulled by this input.  For output, if this value is set to “push”, it means that the data is pushed from this output. Otherwise, the data is pulled from this output. | N/A | string | N/A |
| nonessentiality | If this value is set to the ‘true’, the input (output) is not required/(may be not available from time to time).  The default value is ‘false’. | N/A | boolean | N/A |
| out-of-order | maximum length of out of order push/pulls  The default is 0. | N/A | number | Unsigned integer |
| parallel-requests | maximum number of push/pulls that is possible (allowed) for output/input (input/output).  The default is 0. | N/A | number | Unsigned integer |
| duration | availability of data in milliseconds from the moment it appears in output when it is pulled. | milliseconds | number | Unsigned integer |
| response-latency | maximum time to respond to a pull/push request. | milliseconds | number | Unsigned integer |

## Discussion at MPEG#134

<http://mpegx.int-evry.fr/software/MPEG/Systems/NBMP/Spec-Development/-/issues/72>

Accepted in the WD AMD2/CDAM2:

1. mode
2. availability-duration, availability-duration =0 means that the stream is live, i.e. continuous stream ready to be pulled or continuously pushed.
3. timeout (used to response-latency)

The rest of the proposed parameters be added to TuC for further discussion.

Synchronized metadata handling (m 56728)

## Introduction

NBMP allows two ways of passing parameters to tasks for making any changes at runtime when a workflow has been configured and started: 1) “reconfiguration of the task parameters” defined in the NBMP architecture; and 2) “metadata-parameters” in Input and Output Descriptors. The task reconfiguration can allow passing parameters from any sources (internal and external) through the Workflow Manager, for example, NBMP Source, or other NBMP tasks via event notifications. Timed metadata as the I/O allows a more frequent and deterministic way for tasks to consume the metadata together with the media I/O. Fig. 1 illustrates the 2 situations where dynamically generated metadata are delivered in the out-of-band approach.



**FIG. 1. Out-of-band parameter passing as metadata via task reconfiguration and metadata**

As media and metadata may be sent in different rates, or arrive to the task at different times even while corresponding to the same temporal intervals, the transport protocols (especially over the network) introduce differences in their arrival. When a metadata is generated dynamically by an upstream task, e.g. object detection with its bunding boxes, and transmitted to the downstream tasks for post-processing, the timestamps the metadata carries corresponding to the media (e.g. video frames) can sometimes cause relatively large delay in the out-of-band approach (e.g. via task reconfiguration through the Workflow Manager. The metadata is sent first to the workflow manager and then relay to all relevant tasks) when the post-processing requires accurate timing to couple with the metadata (object bounding boxes) and video frames. FIG 2 is an example illustrating the case where media rate (e.g. frame rates) and object detection rate is different.



**FIG. 2 Different metadata and media rates**

The metadata may be carried either in-band, meaning that they are delivered within the media format (e.g. a container) or multiplexed with the media stream, or out-of-band, meaning that they are delivered externally to the media stream. We believe a new transmission mode that supports in-band parameter (timed metadata) delivery can benefit NBMP and allow accurate and synchronized processing of dynamically generated parameters as timed metadata. This contribution proposes the new features to support the in-band carriage of the media and metadata in a new compound mode. The actual implementation can be specific to the transport protocols or media formats as auxiliary data, or the carriage of timed metadata as metadata tracks (ISO/IEC 23001-10), for example.

## Proposed design

According to the NBMP FDIS specification [1], there are “*media-parameters*” and “*metadata-parameters*” defined in the Input and Output Descriptors. The “input-ports” and “output-ports” specifies the logical endpoints for the connection map object to define the data flows from upstream functions/tasks to downstream functions/tasks. By default, the media and metadata streams are delivered using the transport protocols defined in the I/O descriptors.

It is understood that media and metadata can be sent in the out-of-band way when the parameters such as “stream-id”, “protocol” and “caching-server-url”, which makes the metadata stream different from the correspondent I/O media streams. It is required, in order to support in-band approach, those parameters should have the same value of the correspondent I/O media stream. Additionally, few new parameters are proposed in NBMP descriptors to indicate the in-band carriage of the metadata stream with the media streams over the same protocol used by the media streams.

A new parameter “*compound-mode*” is specified in the “connection-map” object where the connection between input and output ports (or external NBMP media source and sink). Another new parameter “*synchronized-mode*” is used to indicate that the two streams (media and metadata) need to be synchronized even when they have different data rates. Different tasks may have different requirements to the level of synchronization. In the example of real-time viewport-dependent streaming, for instance, it is essential to associate the viewport information with correct 360 video frames along the workflow pipeline whenever the viewport change occurs. In some steps, the processing demands both data (the video frame and metadata data) to be bundled together to be send to, for example, the OpenGL rendering context for shader-based fragment processing. In other scenarios, such strict synchronization may not be required or at least can be handled without any instructions.

**Table 1 New parameters to connection-map object**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Type | Cardinality |
| compound-mode | Specifies the in-band delivery mode of metadata and media data. When the mode is true, media and metadata streams are combined or multiplexed into one medium transferred using the same transport protocol. Default value is false | Boolean | 0-1 |
| synchronized-mode | Specifies the synchronous delivery of metadata and media data, when compound-mode is false. Default value is false. | Boolean | 0-1 |

The “synchronized-mode” addresses the scenario whether or not the media data and metadata need be strictly synchronized when the “compound-mode” is false by default. When the “synchronized-mode” is disabled (false), it would potentially lower the end-to-end latency in rea-time cases without buffering or any other synchronization techniques.

The “synchronized-mode” parameter can be conditionally enabled when the “compound-mode” is false. When the “compound-mode” is activated or true, the carriage of the metadata together with media data enables good data synchronization by default. The media and metadata are implicitly synchronized when the “compound-mode” is activated (true).

The compound mode can be supported by the transport protocols with mutually agreeable data format (specified by the workflow description), or by the function implementation natively. In the latter case, the sending function needs to indicate the compound delivery capability; and likewise, the receiving function need to indicate the capability of parsing the compound data comprising metadata and media data. The following Table 2 defines the new parameter in the General Descriptor.

Table 2 New parameter for General Descriptor

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Type | Cardinality |
| compound-mode-support | Specifies the capability of packing and unpacking of combined media and metadata streams. It is implementation-specific and independent to the transport protocols used by media and metadata streams. Default value is false | Boolean | 0-1 |

We believe that NBMP Implementation Guideline (IG) could further provide recommendations to the transport protocols and/or the carriage format of timed metadata, for example, using MPEG standards like ISO/IEC 23001-10.

## Discussion at MPEG#134

<http://mpegx.int-evry.fr/software/MPEG/Systems/NBMP/Spec-Development/-/issues/73>

We add a use-case, signaling mux media/metadata use case to TuC and we work on how we can improve the input/output descriptor to truly describe the mux format. A generalized input/output descriptor with mixed parameters of media and metadata may provide a more general approach for defining the muxed input/output characteristics

1. [↑](#endnote-ref-2)
2. <https://en.wikipedia.org/wiki/Scalability#Horizontal_(scale_out)_and_vertical_scaling_(scale_up)> [↑](#footnote-ref-2)