aa ISO/IEC JTC 1/SC 29/WG 03 N0872

**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:** 2023-04-28

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

**No. of pages:** 22 (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 N0872**

**April 2023, Antalya, TR**

|  |  |
| --- | --- |
| **Title** | **Technologies under Consideration for Network Based Media Processing** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **22614** |
| **Editor** | **Iraj Sodagar** |

This Technology under Consideration document collects candidate technologies for inclusion into ISO/IEC 23090-8 Network Based Media Processing Standard and/or ISO/IEC 23090-11 Network-based media processing implementation guidelines.

Additionally, this document captures the possible defects of the current specification.

Contents

[1 Improvement of Security descriptor (proposed in m50768) 4](#_Toc109206399)

[1.1 Introduction 4](#_Toc109206400)

[1.2 Previous contribution for supporting protected function usecase 4](#_Toc109206401)

[1.3 Authorization for the protected resource 4](#_Toc109206402)

[1.3.1 Authorization done by NBMP source 4](#_Toc109206403)

[1.3.2 Authorization done by NBMP Workflow Manager 4](#_Toc109206404)

[1.4 Proposed update for supporting private function usage 5](#_Toc109206405)

[1.4.1 Proposed Usecase 5](#_Toc109206406)

[1.4.2 Proposal for update current security parameter 5](#_Toc109206407)

[9.6.1.13 Security parameters 6](#_Toc109206408)

[2 Update of Dynamic resource update on NBMP 8](#_Toc109206409)

[2.1 NBMP Workflow and task Scheduling 8](#_Toc109206410)

[2.1.1 Scheduled-time (m56729) 8](#_Toc109206411)

[2.1.2 Task Group 11](#_Toc109206412)

[3 Synchronized metadata handling (m 56728) 12](#_Toc109206413)

[3.1 Introduction 12](#_Toc109206414)

[3.2 Proposed design 13](#_Toc109206415)

[3.3 Discussion at MPEG#134 14](#_Toc109206416)

[4 Defects under investigation 15](#_Toc109206417)

[4.1 Port mapping 15](#_Toc109206418)

[4.1.1 Problem 15](#_Toc109206419)

[4.1.2 Possible solutions discussion at MPEF#139 15](#_Toc109206420)

# 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 Client 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.

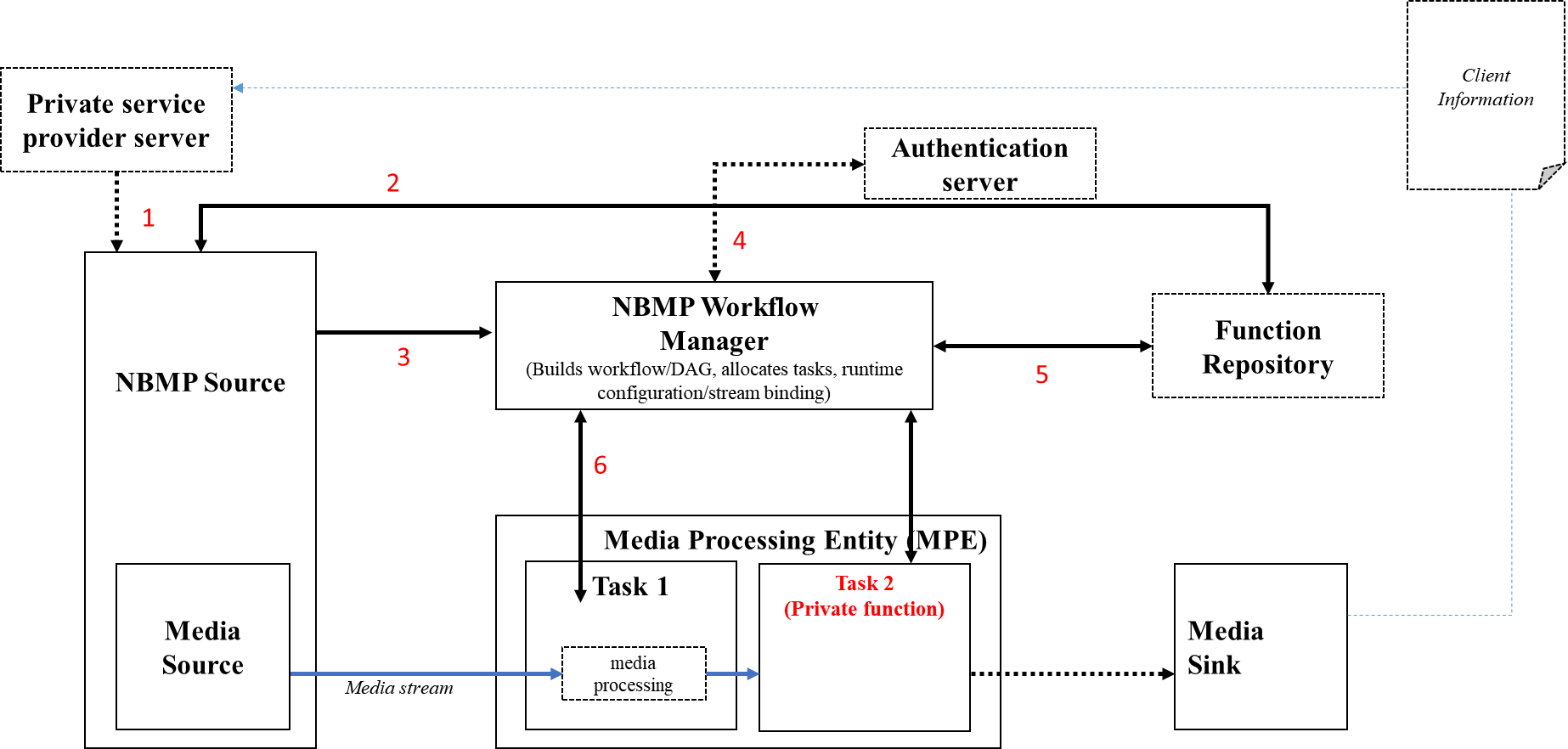
This descriptor provides details about the underlying resource (e.g., workflow, task) in which this is included. Following are the list of parameters in this descriptor.

|  |  |  |
| --- | --- | --- |
| **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.

## 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 |

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.

# Defects under investigation

## Port mapping

### Problem

The mapping of the port names to the input and output descriptors are not clear. The current issues are:

1. The term “port-mapping” is defined but not used.

2. The bind object has two items: stream\_id and name, both required. It is not clear why name is required.

3. It seems that “stream\_id” is used to bind a port-name to an input/output. But the text is not clear. Moreover, Table 1 prohibits the use of stream\_id in a FD. So it is not clear how in FD, the ports are bind to input/output descriptor. Note that in the stream\_id, the use of this parameter for Function is allowed and mentioned.

### Possible solutions discussion at MPEF#139

1. For tasks and workflows, stream\_id is used to bind the port-name to input/output.
2. The name item in bind is optional and carries a logical name for the input/output descriptor.
3. The input or output descriptor does not have the actual ports in a function descriptor, and they are filled when a task is instantiated.
4. We don't have agreement on how port names are mapped to the input/out descriptors in functions. Two different options:

a. use stream-id as it is used in tasks. If accepted, remove the stream-id expectation from TABLE 1.

b. use "name" but make it unique for only function description

1. "port-mapping" either needs to be used or removed as definition.

#### Proposed changes

Table 1 — Function description (FD)

| **Descriptor** | **Additional constraints** | **Cardinality** |
| --- | --- | --- |
| Scheme | None | 0-1 |
| General | Following parameters shall not be present:   * priority; * nonessential; * location; * task-group. * bind parameters: stream-id and name   The parameter id shall be a valid URI according to ETF RFC 3986. | 1 |
| Input | Following parameters shall not be present:   * stream-id; * timeout. | 1 |
| Output | Following parameters shall not be present:   * stream-id. | 1 |
| Processing | If the function is a function group, this descriptor shall contain a connection-map object.  Following parameters shall not be present:   * start-time; * connection-map. | 0-1 |
| Requirements | Following parameters shall not be present:   * proximity; * proximity-equation; * split-efficiency.   In function description, only maximum or minimum values should be specified. | 0-1 |
| Configuration | None | 0-1 |
| Step | If this descriptor is used, the function can be run in step operation.  If the case of stateless support, the input and output descriptors shall contain the sequence-number/start/duration metadata inputs and outputs for the corresponding input/outputs  Following parameter shall not be present:   * operating-units. | 0-1 |
| Client-Assistance | Following parameters shall not be present:   * measurement-collection-list; * source-assistance-information. | 0-1 |
| Assertion | None | 0-1 |
| Variables | The parameters shall not have the following item:   * value. | 0-1 |
| Events | None | 0-1 |
| Security | None | 0-1 |

Table 18— input-port and output-port object

|  |  |  |
| --- | --- | --- |
| **Parameter name** | **Type** | **Cardinality** |
| port-name | P | 1 |
| bind | O | 0-1\* |

The bind object specifies how to associate a port name to a stream, either input or output. For NBMP functions, they provide static information about the port names. The binding is not included in the functions. For NBMP Workflows and tasks, they provide information about the needs for connections between ports and input and output streams by NBMP workflow manager by using the same value for the stream-id parameters

Table 19 — bind object

|  |  |  |
| --- | --- | --- |
| **Parameter name** | **Type** | **Cardinality** |
| stream-id | P | 1 |
| name | P | 0-1 |
| keywords | Array of string | 0-1 |

### JSON schema

|  |
| --- |
| {    "title": "General Descriptor Schema",    "type": "object",    "required": ["id", "name", "description", "input-ports","output-ports", "state"],    "properties": {      "id": {"type": "string"},      "name": {"type": "string"},      "description": {"type": "string"},      "rank": {        "type": "integer",        "minimum": 0      },      "nbmp-brand": {        "type": "string",        "format": "uri",        "pattern": "^urn:mpeg:mpegi:nbmp:(2([0-9]{3})):([a-zA-Z0-9\_]+)$",        "additionalProperties": false      },      "published-time": {        "type": "string",        "format": "date-time"      },      "priority": {"type": "number"},      "location": {"type": "string"},      "task-group": {        "type": "array",        "minItems": 1,        "uniqueItems": true,        "items": {          "type": "object",          "required": ["group-id", "task-id"],          "properties": {            "group-id": {"type": "string"},            "task-id": {              "type": "array",              "minItems": 1,              "uniqueItems": true,              "items": {"type": "string"}            },            "group-type": {              "type": "string",              "enum": ["distance", "sync", "virtual"],              "default": "distance"            },            "group-mode": {              "type": "string",              "enum": ["synchronous", "asynchronous"],              "default": "synchronous"            },            "net-zero": {              "type": "boolean",              "default": "false"            }          }        }      },      "input-ports": {        "type": "array",        "minItems": 1,        "uniqueItems": true,        "items": {          "type": "object",          "required": ["port-name"],          "properties": {            "port-name": {"type": "string"},            "bind": {              "type": "object",              "required": ["stream-id"],              "properties": {                "stream-id": {"type": "string"},                "name": {"type": "string"},                "keywords": {                  "type": "array",                  "minItems": 1,                  "uniqueItems": true,                  "items": {"type": "string"}                }              }            }          }        }      },      "output-ports": {        "type": "array",        "minItems": 1,        "uniqueItems": true,        "items": {          "type": "object",          "required": ["port-name",],          "properties": {            "port-name": {"type": "string"},            "bind": {              "type": "object",              "required": ["stream-id"],              "properties": {                "stream-id": {"type": "string"},                "name": {"type": "string"},                "keywords": {                  "type": "array",                  "minItems": 1,                  "uniqueItems": true,                  "items": {"type": "string"}                }              }            }          }        }      },      "is-group": {        "type": "boolean",        "default": false      },      "nonessential": {        "type": "boolean",        "default": false      },      "state": {" type": "string"}    }  } |

## Function restrictions

The processing descriptor defines function restriction array for each function as:

“The entity function-restrictions, as shown in Table 26, is an array. Each element of this array describes the additional restriction for one function instance used in a function group, as shown in Table 32.

Table 32 — function-restrictions array element

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Cardinality** | |
| instance | instance identifier | P | 1 | |
| general | general descriptor restriction of this instance as defined in Table 16 | O | 0-1 | |
| processing | general descriptor restriction of this instance as defined in Table 26 | O | | 0-1 |
| requirement | requirement descriptor restriction of this instance as defined in Table 33 | O | | 0-1 |
| configuration | configuration descriptor restriction of this instance as defined in Table 42 | O | | 0-1 |
| client-assistance | client-assistance descriptor restriction of this instance as defined in Table 44 | O | | 0-1 |
| failover | fail-over descriptor restriction of this instance as defined in Table 46 | O | | 0-1 |
| monitoring | monitoring descriptor restriction of this instance as defined in Table 50 | O | | 0-1 |
| reporting | reporting descriptor restriction of this instance as defined in Table 51 | O | | 0-1 |
| notification | notification descriptor restriction of this instance as defined in Table 52 | O | | 0-1 |
| step | Step Descriptor restriction of this instance as defined in Table 62  When specified, this step descriptor is applied to this function instance. | O | | 0-1 |
| security | security descriptor restriction of this instance as defined in Table 61 | O | | 0-1 |
| blacklist | The list of excluded WDD descriptors for this function instance | Array | | 0-1 |

In Table 32, the descriptors define allowed values for the function instance used in the function group.”

It is not clear what “allowed values” means. Each of the restriction element use the descriptors used for function/task description. Does each element here repeats the same parameters and values and optionally limits the values, or whether any of these elements only include the updated limited parameters and values of the descriptors?

It seems that repeating the entire descriptor is not useful. However, at the same time using the same JSON objects make the use of descriptor easier.

Two possible solutions:

a) The descriptors in function restriction contain only updated values of parameter that are changed.

b) The descriptors replace the function descriptors before for each instance.

## Schema corrections

### Introduction

During the development of conformance tools for NBMP, we realized that the way various schemas are derived from the scheme-definition file, would not provide the constraints the specification requires.

### Problem

The NBMP specification, for each of FD, TD, WD and MD, defines some exception to the general descriptor schemas. For instance, the FD exclude some of the parameters from various descriptors:

Table 1 — Function description (FD)

| **Descriptor** | **Additional constraints** | **Cardinality** |
| --- | --- | --- |
| Scheme | None | 0-1 |
| General | Following parameters shall not be present:   * priority; * nonessential; * location; * task-group.   The parameter id shall be a valid URI according to ETF RFC 3986. | 1 |
| Input | Following parameters shall not be present:   * stream-id; * timeout. | 1 |
| Output | Following parameters shall not be present:   * stream-id. | 1 |
| Processing | If the function is a function group, this deC:\Users\Iraj Sodagar\OneDrive\Tencent\MPEG\MPEGGroup\NBMP-2ndEdition\NBMP\nbmp-schema-definitions.jsonscriptor shall contain a connection-map object.  Following parameters shall not be present:   * start-time; * connection-map. | 0-1 |
| Requirements | Following parameters shall not be present:   * proximity; * proximity-equation; * split-efficiency.   In function description, only maximum or minimum values should be specified. | 0-1 |
| Configuration | None | 0-1 |
| Step | If this descriptor is used, the function can be run in step operation.  If the case of stateless support, the input and output descriptors shall contain the sequence-number/start/duration metadata inputs and outputs for the corresponding input/outputs  Following parameter shall not be present:   * operating-units. | 0-1 |
| Client-Assistance | Following parameters shall not be present:   * measurement-collection-list; * source-assistance-information. | 0-1 |
| Assertion | None | 0-1 |
| Variables | The parameters shall not have the following item:   * value. | 0-1 |
| Events | None | 0-1 |
| Security | None | 0-1 |

The exclusions are shown in the yellow highlights.

The schemas for each document however are derived from the basic schema by referencing the schema for each descriptor:

 "properties": {

    "scheme": {"$ref": "nbmp-schema-definitions.json#/scheme"},

    "general": {"$ref": "nbmp-schema-definitions.json#/general"},

    "input": {"$ref": "nbmp-schema-definitions.json#/input"},

    "output": {"$ref": "nbmp-schema-definitions.json#/output"},

    "processing": {"$ref": "nbmp-schema-definitions.json#/processing"},

    "requirement": {"$ref": "nbmp-schema-definitions.json#/requirement"},

    "configuration": {"$ref": "nbmp-schema-definitions.json#/configuration"},

    "step": {"$ref": "nbmp-schema-definitions.json#/step"},

    "client-assistant": {"$ref": "nbmp-schema-definitions.json#/client-assistant"},

    "assertion": {"$ref": "nbmp-schema-definitions.json#/assertion"},

    "variables": {"$ref": "nbmp-schema-definitions.json#/variables"},

    "events": {"$ref": "nbmp-schema-definitions.json#/events"},

    "security": {"$ref": "nbmp-schema-definitions.json#/security"}

  }

In this method, the exclusions are not addressed.

During the development of the conformance schemas, we realized that the JSON scheme has additive properties, i.e. if a schema required a property, the schema can no later forbid it. In the updated conformance schema, we use the not property to exclude any forbidden property:

"not": {"required": ["priority", "nonessential", "location", "task-group"]}

It seems the electronic schemas included in the standard therefore are not correct and need to be updated with the conformance schemas.

The conformance schemas are attached to this document.

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