**COM Book Update**

Solving the “History” problem

**Problem: No tracking of History of COM Objects**

Currently, the history of COM Objects is not being tracked natively. This means that previous versions of a COM Object are not captured and only the latest version is available.

In the M&C book, a workaround for this problem was implemented but it is not simple and it has unnecessarily increased the size of the M&C book by 30%:

* **Before:** draft 3 was **153 pages**
* **After:** final book is **202 pages**

The workaround is known as the “Identity model” and basically, separates the name of the definitions into dedicated COM Objects known as Identity COM Objects.

Example for the Parameter service: Previously, the name of the definition would be held directly in the Definition COM Object. After the “Identity model” workaround, the name of the parameter was moved to a dedicated ParameterIdentity COM Object. Every time a change to a definition happens, a new ParameterDefinition COM object is created and the related link is set to point to the ParameterIdentity.

The “Identity model” workaround proved to be bad in actual projects!

**Proposed solution: Add an additional history link in the COM Objects and Create a History service**

Underlying principle of the solution: By storing the deltas that happen during each update of a COM Object, it is possible to generate its previous versions.

The proposed solution for this problem involves two steps:

1. Add COM Object history link
2. Create a History service
3. **Add COM Object history link**

The history link will be pointing to History COM Object.

By adding a history link to the COM Object Model, it is possible to keep track of the history of the COM Objects. Any change to a COM Object is captured as a delta and stored in the History COM Object.



**Figure** - New “Common Model Object structure” with one more link

The "Common Model Object Structure" (section 2.2.2), on Figure 2-2 of the COM book, will need to be updated with a history link as presented in the figure above. This link shall be used to point to a "History COM Object".

Table 1-1: History Service COM Objects

|  |  |  |  |
| --- | --- | --- | --- |
| Object Name | Object Number | Object Body Type | Related points to |
| History | 1 | List<HistoryDelta> | Not used |

### Composite: HistoryDelta

The HistoryDelta structure holds the deltas that happens to a COM Object in time. The structure has a timestamp field that contains the time of the update and holds a list of changes that happened in the COM Object fields.

|  |  |
| --- | --- |
| Name | HistoryDelta |
| Extends | MAL::Composite |
| Short Form Part | 1 |
| Field | Type | Nullable | Comment |
| timestamp | MAL::Time | No | The timestamp of the update. |
| updatedFields | List<HistoryFieldUpdate> | No | List of fields that were updated. |

### Composite: HistoryFieldUpdate

The HistoryFieldUpdate structure holds the update that occurred to a certain field.

|  |  |
| --- | --- |
| Name | HistoryFieldUpdate |
| Extends | MAL::Composite |
| Short Form Part | 1 |
| Field | Type | Nullable | Comment |
| field | MAL::String | No | The name of the field that was updated. |
| from | MAL::Blob | No | The previous value of the field. |
| to | MAL::Blob | No | The new value of the field. |

1. **Create a History service**

The History service allows consumers to request the history of COM Objects and to request previous specific versions of a COM Object.

Table 1-2: History Service Operations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Area Identifier | Service Identifier | Area Number | Service Number | Area Version |
| COM | History | 2 | 4 | 2 |
| Interaction Pattern | Operation Identifier | Operation Number | Support in Replay | Capability Set |
| REQUEST | getHistory | 1 | No | 1 |
| REQUEST | getSpecificVersion | 2 | No | 2 |

### High Level Requirements

The History service shall provide:

* the capability for retrieving the history of updates of a COM Object
* the capability for retrieving the different versions of a COM Object

Additional Requirements:

* The "History COM Object" shall hold the deltas of the COM Object that it is pointing to it. The history link shall always be null when a new COM Object is created.
* The COM Archive needs a new requirement on the "update" operation. Basically, it must state that, after an update, the respective "History COM Object" needs to be created or updated. The "deltas" field must be complemented with a new HistoryDelta on every new change. The new requirements must all be "SHOULD" requirements in order to avoid having a "dependency" on the History service. So, in theory, the Archive service can be implemented without the History service.
* To prevent the duplication of COM Objects during the first store, all the stored COM objects shall start without a history link (pointing to null). This first "History COM Object" must be created only if the update operation is called in the Archive service.

**10/09/2020**

Limitation of the scope:

* Short time to middle time past => Long time is done in the OIA WG
* We don’t want to retrieve too many date (so that it doesn’t take 5 days to provide the data).

Parameter

* MNEMO
* Definition
* Value [ raw, converted, validity state ]

GetValue(MNEMO)

getDef(MNEMO)

monitorValue(MNEMO)

changes from definitions between t1 and t2

values of MNEMO from time t1 to t2

givevalue/def(MNEMO, t1)

Definition<MNEMO> 🡨 Value<MNEMO> vs Service Object “Parameter”

**Parameter with version**

The only thing that would increase the version is a definition change.

|  |  |
| --- | --- |
| Value object | Definition object |
| * MNEMO (, Version)
* Value [ raw, converted, validity state ]
 | * MNEMO (, Version)
* Definition
 |

Remark: how to deprecate a MNEMO or a version that will never be used again?

Two options:

* Create a versioning service external
* Service Object Type <Parameter> = { name/id=MNEMO , versionTag<integer>=1 }