Notes on CCOM Version Compatibility

MIMOSA

February 21, 2019

# Notes on backwards/forwards compatibility

CCOM 4.1.0 introduces some changes that are not strictly backwards compatible due to their importance. However, the schema has been adjusted to maximize backwards/forwards compatibility to avoid issues when exchanging data with applications/adaptors implemented against CCOM 4.0.x. This document describes some considerations that need to be made if full backwards compatibility needs to be maintained within an environment. MIMOSA recommends upgrading to 4.1.0 and above.

Note: where optional elements have simply been *added* in CCOM 4.1.0, it is considered backwards compatible assuming the parsers ignore elements that they do not understand.

A simple [XSLT transformation](XSD/4.1-to-4.0.xslt) is also provided, which can be used to ensure XML content that may be CCOM 4.1.0 conforms to CCOM 4.0. Where this transformation is lossy, it is mentioned in the text below.

## Versioning

An attribute, ccomVersion, has been added to the schema to support versioning. This attribute should be used in the CCOMData element of Compound Documents and in the ApplicationArea/UserArea of BODs to indicate the version compatibility. For example,

<CCOMData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   
 xmlns="http://www.mimosa.org/ccom4"   
 ccomVersion="4.1.0">  
 <Entity xsi:type="...">  
 ...  
 </Entity>  
</CCOMData>

for a Compound Document, or

<SyncAssetSegmentEvents xmlns:oa="http://www.openapplications.org/oagis/9"  
 xmlns="http://www.mimosa.org/ccom4"  
 releaseID="1.2.1"   
 versionID="1.0"> <!-- == BOD version, not CCOM Schema version, -->  
 <oa:ApplicationArea>  
 <oa:Sender>  
 <oa:LogicalID>3939cddb-fba2-4e77-bdd7-ac0cae069741</oa:LogicalID>  
 </oa:Sender>  
 <oa:CreationDateTime>2019-01-09T03:13:09Z</oa:CreationDateTime>  
 <oa:BODID>56f60441-cc4c-4fc2-9524-0df921187974</oa:BODID>  
 <oa:UserArea>  
 <CCOMData ccomVersion="4.1.0" />  
 </oa:UserArea>  
 </oa:ApplicationArea>  
 <DataArea>  
 <oa:Sync/>  
 <AssetSegmentEvents>  
 <Asset>  
 <UUID>df3cb180-e410-11de-8a39-0800200c9a66</UUID>  
 <ShortName>3Z84G32AA0-4 AC Induction Motor</ShortName>  
 <RegistrationSite>  
 <UUID>833d1881-80df-4fcc-be80-bbc5f2395e58</UUID>  
 <ShortName>Juneora</ShortName>  
 </RegistrationSite>  
 </Asset>  
 <Segment>  
 <UUID>3f8f5618-3ee7-4c30-afe9-0f80e24d4f45</UUID>  
 <ShortName>MTR-101</ShortName>  
 <RegistrationSite>  
 <UUID>833d1881-80df-4fcc-be80-bbc5f2395e58</UUID>  
 <ShortName>Juneora</ShortName>  
 </RegistrationSite>  
 </Segment>  
 <AssetSegmentEvent>  
 <UUID>76238289-57a9-4ef5-888f-ea131b46dd60</UUID>  
 <Type>  
 <UUID>ecc99353-412b-4995-bd71-1cbc6fc16c7c</UUID>  
 <ShortName>Installation of Asset on Segment</ShortName>  
 </Type>  
 <End>2019-03-30T13:21:00Z</End>  
 <Asset>  
 <UUID>df3cb180-e410-11de-8a39-0800200c9a66</UUID>  
 <ShortName>3Z84G32AA0-4 AC Induction Motor</ShortName>  
 </Asset>  
 <Segment>  
 <UUID>3f8f5618-3ee7-4c30-afe9-0f80e24d4f45</UUID>  
 <ShortName>MTR-101</ShortName>  
 </Segment>  
 </AssetSegmentEvent>  
 </AssetSegmentEvents>  
 </DataArea>  
</SyncAssetSegmentEvents>

for a BOD. In the latter, an empty CCOMData element is included in the BOD’s UserArea with the attribute set to the CCOM version.

This attribute aids in version management and assists applications in determining whether they can process a particular CCOM XML instance. While the aim is to have compatibility between versions of the CCOM schema, it is possible for patches and other updates to break compatibility, so it is important to be able to identify the version of CCOM with which a particular instance is strictly compatible.

Note: when new optional elements are added in new releases of CCOM, it is assumed to be backwards compatible as long as the parsers are configured to ignore what they do not understand; however, this may not always be the case for parsers that strictly validate against the expected schema. Moreover, a later version of the schema successfully validates instances conforming to prior schemas; therefore, an application implemented to a later schema version **must** be able to read an instance generated by an application implemented to an earlier version (unless the specification indicates otherwise). Very large breaking changes should result in a major version increment and a new namespace; therefore, the use of this attribute does not have to be considered across major versions (such as CCOM 4 v.s. CCOM 5).

The value of this attribute should contain the lowest version of the CCOM schema to which the instance strictly conforms. For example, instance data serialized according to the 4.0.0 schema should specify “4.0.0”, while data serialized according to the 4.1.0 schema may specify “4.1.0” **or** “4.0.0”: the latter only if the data has been explicitly serialized using the to preserve compatibility with 4.0.0. Each revision should contain guidance on what is and is not compatible between versions.

## Properties and Attributes

CCOM 4.1.0 deprecated the use of the term Attribute in favour of the term Property. As such the UML model and schema were updated to reflect the preferred terminology. The following describes the changes to the XML Schema that support backwards compatibility.

(The ’\*’ in the following means a wildcard match, e.g., Attribute\* could refer to AttributeSet)

* Renamed Attribute\* complex types to Property\*
* Added Attribute\* complex types as extensions to Property\* complex types. This is required as top-level entities may have their xsi:type defined as one of the Attribute\* types
  + when serializing out top-level Property\* types for backwards compatibility, they must generate a xsi:type="Attribute\*" attribute
* Elements with types referring to Attribute\* complex types now refer to their Property\* counterparts
* Elements named Attribute\* are added to an xs:choice giving the option of the Attribute\* element or a Property\* counterpart.

To ensure a CCOM XML instance is backwards compatible, the Property\* data **must** be serialized using the Attribute terminology. Instances using the Attribute terminology should be read by a CCOM 4.1.0 conforming application in an identical fashion to the Property terminology.

## Asset/Entity Lifecycles (LifecycleStatus)

CCOM 4.1.0 has expanded the ability for different entities to have lifecycles, whereas it was only Asset and Segment previously. This is in large part to allow Request to have lifecycles but other Entity types may also have their own lifecycles, e.g., Models. Moreover, the lifecycle model was incomplete and has now been expanded to define groups of lifecycle statuses under a type, clarified the past/present/future status history, etc. This allows an entity to have multiple lifecycle statuses at the same for different categories of status, allowing different organization or units to have relevant lifecycle categories that may differ in the individual lifecycle stages.

To maintain backwards compatibility, Asset and Segment maintain the old elements LifecycleStatus and LifecycleStatusType, which map respectively to: \* the types LifecycleStatusHistory and LifecycleStatusType of the new model; and \* the new elements LifecycleStatusHistory and PresentLifecycleStatus of Entity

Moreover, some of the constraints visible in the UML model have been relaxed for compatibility.

To ensure a CCOM XML instance is fully backwards compatible with 4.0, the Entity lifecycle fields, PresentLifecycleStatus and LifecycleStatusHistory, **must** **not** be used. Instead the fields LifecycleStatus and LifecycleStatusType of Segment and Asset **must** be serialized according to the mapping described above. This means that in a 4.0 compatible exchange, only segments and assets may have associated lifecycle statuses and they may have only 1 ‘present’ status each. The Entity and StatusFromDate fields of LifecycleStatusHistory **must** **not** be serialized to achieve full 4.0 conformance. Finally, when serializing subtypes of BaseType or a CCOMClass the ValidLifecycleStatusKind field **must** **not** be serialized