1 Overview

For the release 1 , the **Object Manager** (OM) offers services in the following areas:

- **Content Items:** These are the objects in the eSciDoc system that comprises the institutional repository. A content item is a simple or complex object like an article, report or collection.
  The Object Manager allows versioning as well as revisioning of Content Items and will handle relations between Content Items.
  Content Items can have various statuses, which also have an influence on whether actions are applicable to them or not.
  The Content Items can be differentiated by their aggregation type (item or container). Containers offer the concept of aggregation, i.e. they can contain other (simple and complex) objects. Each container can include a structural map and an Admin Descriptor.

  In the first release, the Object Manager supports creation, retrieval, deletion, logical deletion, versioning, revisioning, publishing and withdrawal of Content Items as content relations and structural relations between Content Items.

  Release notes 0.1:
  In the release 0.1, the Object Manager supports creation, retrieval, deletion, logical deletion of Content Items with aggregation type “item”. Furthermore the content types(e.g. article, collection) are not defined for the release 0.1.

- **Content Components:** Are used when a Content Item with aggregation type “item” should not only store metadata about a specific content, but the content itself as digital objects. The Content Component will be handled as a separate part of the Content Item, enriched with metadata and licenses to facilitate handling and processing. A Content Component can not exist without a Content Item.

  In the first release, the Object Manager supports adding, retrieval, update and logically deletion of Content Components to / from Content Items.

  Release notes 0.1:
  In the release 0.1, the Object Manager supports adding, retrieval and logically deletion of Content Components to / from Content Items.
  Validating against a xml schema for the content component property “technical metadata” is not yet implemented. An example xml for “technical metadata” is provided in the testOMAcc8. The xml elements “fileName” and “mimetype” must be set to.

- **Admin Groups:** These are objects that are used to administrate Content Items. They will not be versioned but can also have various statuses, which also has an influence on whether actions are applicable to them or not.

  In the first release, the Object Manager supports the retrieval of Admin Groups as well as adding, retrieval and moving of Content Item to / from Admin Groups. However, there will be no interface for creating, deleting, opening and closing of Admin Groups, so this has to be done manually in the system.

  Release notes 0.1:
  In the release 0.1, the Object Manager supports retrieval of Admin Groups.

- **Metadata:** A Metadata Record describes a Content Item in a given Metadata Schema. A Content Item can have more than one Metadata Record. A Metadata Record contains references to Metadata Elements of the respective Metadata Schema and values for these Metadata Elements. Depending on the Metadata Element definition, a Metadata Record may have more values for a single Metadata Element. Admin Groups do not have their own internal metadata record but only an admin descriptor.
  Note: A Content Item has an internal MD Record as well as a set of properties. The differences between these two are that the MD Record is versioned while the properties are not.

  In the first release, the Object Manager allows updating, retrieval of Metadata records to / from Content Items.

  Release notes 0.1:
  In the release 0.1, the Object Manager supports only the Escidoc internal metadata record.

- **Relations:** The Items stored in the Object Manager can be related to each other. The relations have types, names and are directed. Structural Relations have no metadata but are versioned (e.g. the "is administrated by" relation between Content Item and Admin Group). Content Relations can have additional metadata and are versioned (e.g. the "is annotated by" between two Content Items.

  Release notes 0.1:
  In the release 0.1, the Object Manager does not support relations.

- **Identification:** Some of the object types in the eSciDoc framework will get assigned a persistent identifier (PID) at a certain point in their lifecycle, by which they can be identified no matter where they are currently located. These PIDs are also handled by the Object Manager.

  In the first release, the Object Manager supports creation and assignment of PIDs during the publishing process to Content Items and Content Components as well as retrieval of Content Items via PID or mapping of PID to the internal ID.

  Release notes 0.1:
  In the release 0.1, the Object Manager does not support PIDs.

- **Validation:** This means checking the validity of objects handled by the Object Manager.

  In the first release, the Object Manager will only validate Content Items by checking their assigned Metadata records, using the Metadata Modeler component. This will be done every time the metadata record is changed. There is no public interface for validation in release one.

- **Licenses**: are referenced from Content Components. They are stored in the Object Manager, to provide a list of all possible Licenses.

  Release notes 0.1:
  In the release 0.1, the Object Manager supports retrieval of licenses. However, there will be no interface for creating and deleting of licenses, so this has to be done manually in the system.

- **Versioning**: a subset of object stored in the Object Manager are versioned. Specific versions can be identified by a version number (simple number like 1,2,3,…)

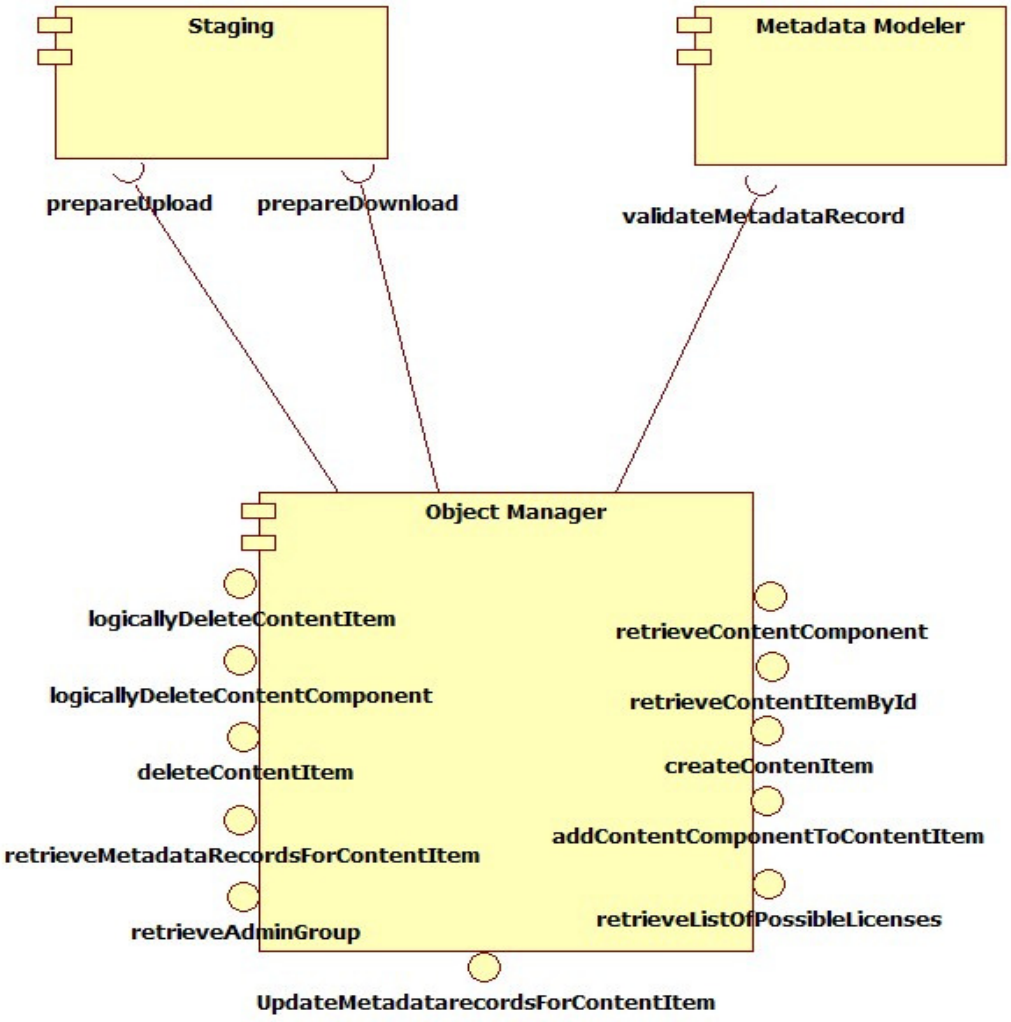
  In the first release, the Object Manager doesn't support versioning.

Implemented Services in the Release V.01:

> createContentItem
> addContentComponentToContentItem
> updateMetadataRecordsForContentItem
> retrieveMetadataRecordsForContentItem
> retrieveAdminGroup
> retrieveContentItemById
> retrieveContentComponent
> retrieveListOfPossibleLicenses
> logicallyDeleteContentComponent
> logicallyDeleteContentItem
> deleteContentItem

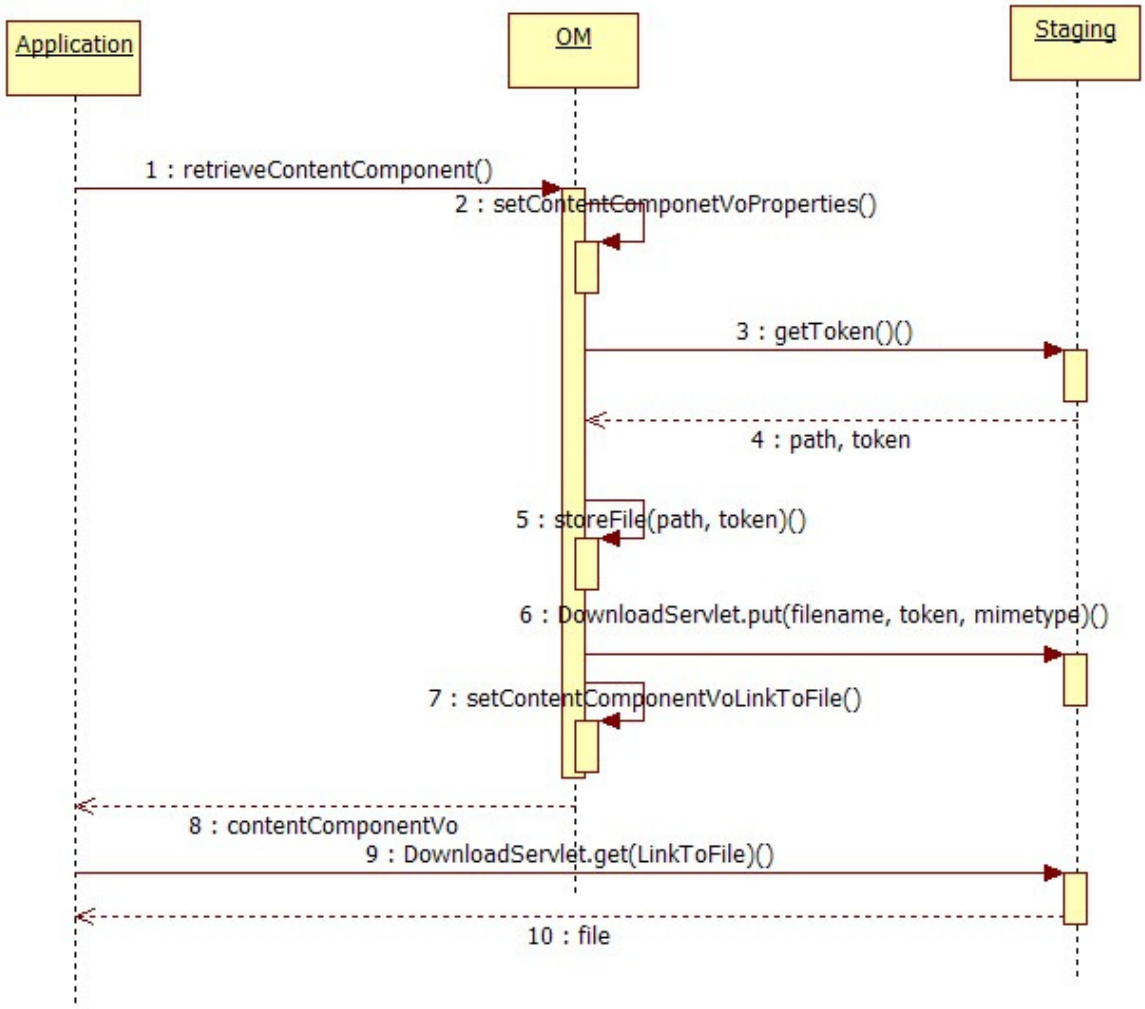
2 Interaction with other Components (release 0.1)

Figure 1 shows the interfaces which are needed for the interaction of the object manager with other components in the framework.



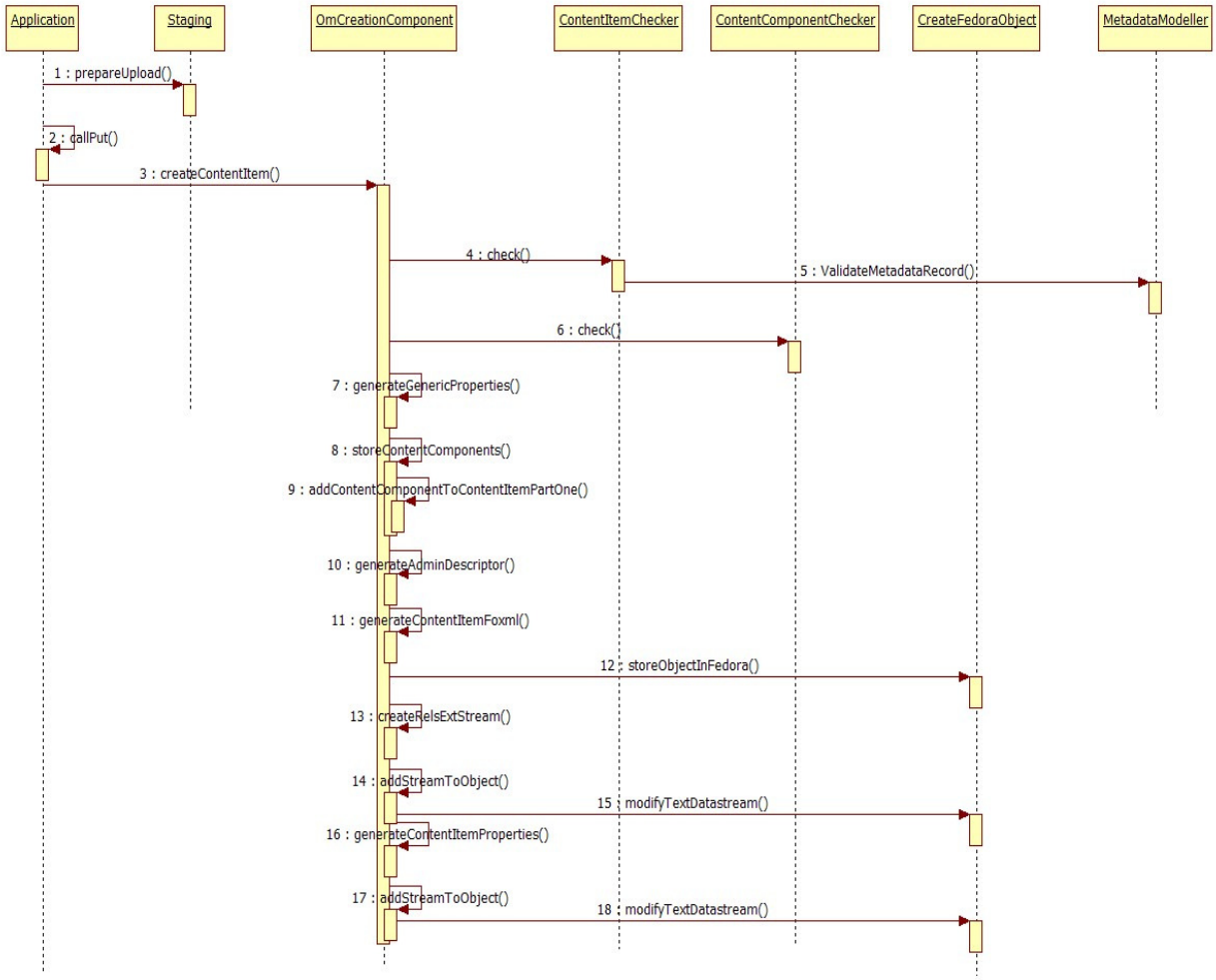
The sequence diagrams below show these interactions in more detail:

Sequence Diagramm: retrieveContentComponent: the diagram should give a general idea of the content component retrieval process. To simplify matters the sequence diagram does not show the actual method calls of classes of the OM component, but rather emphasises important steps in the retrieval process.



1: OM.retrieveContentComponent(): orchestrates the process of retrieving a content component Vo
2: setContentComponentVoProperties(): reads out Properties of content component Vo from FOXML, which represents this content component Vo. Set Properties in the contentComponentVo.
3: Staging.getToken(): gets a token to enable upload of a local file in to a staging area
4: path to the directory in the staging area to store the file to be uploaded , token
5: storeFile: stores the file in the staging area
6: DownloadServlet.put(): notify the Download-Servlet about the file name, the file location in the staging area and the mimetype
7: sets the property LinkToFile of the content component Vo to the access-URL of the file in the staging area
8: DownloadServlet.get(): returns content component Vo
9: access the file in the staging area

Sequence Diagramm: OmCreationComponent The diagram should give a general idea of the content item creation process. For increasing the readability of the diagram the longish chains of input parameters (and return values) where left out. When needed, that information can be found in the javadoc.



1: Staging:prepareUpload: generates and gives back the upload token.

2: Staging:callPut: uploads a datastream to the Webserver.

3: OmCreationComponent:createContentItem: is orchestrating the process of creating a content item.

4: OmCreationComponent->ContentItemChecker:check: validates the content item.

5: MetadataModeller:validateMetadataRecord: validates MetadataRecords.

6: OmCreationComponent->ContentComponentChecker:check: validates the related content components.

7: OmCreationComponent:generateGenericProperties: is preparing a XML-Snippet, which is carrying the GenericProperties information.

8: OmCreationComponent:storeContentComponents: is adding all dependend ContentComponents to a ContentItem.

9: OmCreationComponent:addContentComponentToContentItemPartOne: This Method is a helper method. It does everything what should happen in the process of adding a ContentComponent to a ContentItem beside updating the ContentItem- Relations. Therefore this method can be used either the by storeContentComponents method or by the addContentComponentToContentItem method as a separate webservice call.

10: OmCreationComponent:generateAdminDescriptor: This Method is generating the AdminDescriptor-Stream for the ContentItem.

11: OmCreationComponent:generateContentItemFoxml: Generates the initial FOXML for the ContentItem.

12: CreateFedoraObject:storeObjectInFedora: executes the Fedora APIM and ingestes the foxml.

13: OmCreationComponent:createRelsExtStream: The Method is preparing the RelsExt-Stream for the ContentItem. It writes the specific ContentItem Properties, the Relations to ContentComponents and some escidoc-Metadata to the the Rels-Ext-XML.

14: OmCreationComponent: addStreamToObject: This Method writes any Stream(identified by Name) to the Fedora-Representation of the ContentItem. (Here the RelsExt Stream which was empty before is written)

15: CreateFedoraObject: modifyTextDatastream: modifies a stream of a Fedora Object. It is used by 14.

16: OmCreationComponent: generateContentItemProperties: This Method is generating the Properties-Stream for the ContentItem

17: OmCreationComponent: addStreamToObject: This Method writes any Stream(identified by Name) to the Fedora-Representation of the ContentItem. (Here the Properties Stream which was empty before is written)

18: CreateFedoraObject: modifyTextDatastream: modifies a stream of a Fedora Object. It is used by 17.

3 Comments about the implementation state:

CreateContentItem:

> As there is not yet a Content Type Modeller existing, the ContentItem can not yet be checked against a Content Type.

> We have agreed on , that the ContentItemVo can either represent items or containers. At the moment the container behaviour is not yet implemented. The aggregation type must always be set to item.

> Validating against a xml schema for the content component property “technical metadata” is not yet implemented. An example xml for “technical metadata” is provided in the testOMAcc8. The xml elements “fileName” and “mimetype” must be set.

LogicallyDeleteContentItem:

The notification of SB component about logically deleted content item is not yet implemented.
Since at the moment the container behaviour is not yet implemented, the method works only for content item with aggregation type “item”.

DeleteContentItem:

The notification of SB component about deleted content item is not yet implemented.
Since at the moment the container behaviour is not yet implemented, the method works only for content item with aggregation type “item”.