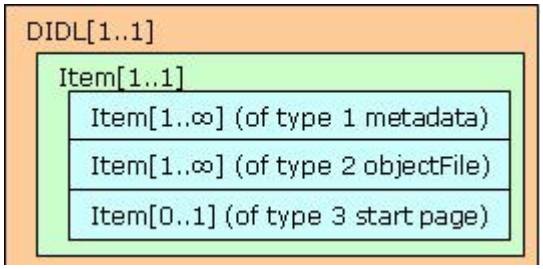
DIDL as wrapper

The DIDL XML container, as defined in DRIVER, is a document with one top-level Item element. The Item contains several child Item elements. These child item elements appear in three different kind of types. Between the straight brackets the cardinality of the XML elements are shown:

```
<metadata>
  <didl:DIDL ...>
    <didl:Item>
      <didl:Item>...</didl:Item>
      <didl:Item>...</didl:Item>
      <didl:Item>...</didl:Item>
    </didl:Item>
  </didl:DIDL>
</metadata>
```



Root Element: DIDL document Identification attribute

The DIDL root element contains *one* attribute; namely DIDLDocumentID. This attribute provides information about the Identifier of the DIDL wrapper as an autonomous entity. This Identifier is NOT to identify the intellectual work, but to Identify the serialisation of the DIDL XML.

```
<didl:DIDL
    DIDLDocumentId="urn:nbn:nl:ui:10-15290" <!-- Identification -->
    ...
>
    ...
</didl:DIDL>
```

The DIDLDocumentId attribute contains the ID of the DIDL wrapper. This CAN be the same as the OAI-Identifier that is being used to get a record. The DIDL wrapper can be used as an autonomous entity out of the OAI-PMH context, therefore a DIDL is not the same thing as an OAI record. There is a demand for Persistent Identifiers assigned to digital objects in the future (mandatory for the OAI-ORE project.). For libraries it is recommended to use

```
urn:nbn:<country code>:<isil library code>-<object id>.
```

Where <country code> is the ISO 3166-1 country code, <isil library code> is the ISO/NP 15511 International Standard Identifier for Libraries and Related Organizations, and <object id> is a persistent database number.

Remarks

1. This DIDLDocumentId has in the first place a different Identifier than the OAI identifier for this record. The rationale behind this is that a DIDL document is an autonomous entity that can exist outside and separate of an OAI-record. However for easing the operational implementation, it is allowed to use the Identifier that is used for the OAI record when both, OAI record and DIDL document are inextricably bound together.
2. {ISIL library code} ISO/NP 15511: International Standard Identifier for Libraries and Related Organizations (ISIL)http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=52666
3. {object id} could be the database number. It is recommended to store this number in a separate field and not to auto generate from the database id because a database update in the future will change these numbers and the persistency could be lost.

Item Descriptor Elements (optional)

The Item elements can **OPTIONALLY** contain *two* or *three* Descriptor elements. One Descriptor element describes the modification date of the Item element. To compare similar harvested Item elements on modification date, an identifier must be added.

Example on level one:

```
<didl:DIDL ...>
  <didl:Item>
    <didl:Descriptor>...</didl:Descriptor> <!-- Identification -->
    <didl:Descriptor>...</didl:Descriptor> <!-- Modification date -->
    <didl:Item>...</didl:Item>
    <didl:Item>...</didl:Item>
    <didl:Item>...</didl:Item>
    ...
  </didl:Item>
</didl:DIDL>
```

Example on level two; Object type added:

```
<didl:Item> <!-- Level 1 Root Item -->
    <didl:Item> <!-- Level 2 Child Item -->
      <didl:Descriptor>...</didl:Descriptor> <!-- Identification -->
        <didl:Descriptor>...</didl:Descriptor> <!-- Modification date -->
        <didl:Descriptor>...</didl:Descriptor> <!-- Object type -->
      ...
      </didl:Item>
        <didl:Item>...</didl:Item>
        <didl:Item>...</didl:Item>
        <didl:Item>...</didl:Item>
    ...
  </didl:Item>
</didl:DIDL>
```

Descriptor Statement: Item 'Identifier'

The *first* Descriptor contains the ID of the Item elements. This is mostly used to uniquely identify the digital object (e.g. with a DOI). This ID is wrapped in a Statement with a DII Identifier element. For example:

```
<didl:Item>
  <didl:Item>
    <didl:Descriptor>
      <didl:Statement mimeType="application/xml">
        <dii:Identifier>urn:nbn:nl:ui:10-6748398729821</dii:Identifier>
      </didl:Statement>
    </didl:Descriptor>
    ...
  </didl:Item>
  ...
</didl:Item>
```

Remarks

1. For child Item elements of the root Item element accounts that this Identifier is NOT equal to the used OAI identifier or DIDL identifier
2. The Identifier in the root Item element CAN be the same as the DIDL or OAI Identifier, but this is not recommended
3. The namespace for dii has had to be declared in the DIDL tag
4. The Identifier in the HAS TO BE described as an URI when applicable

Descriptor Statement: Item 'modified'

The *second* Descriptor contains a modification date. When something changes inside an Item, this modification date element has to be up-dated. This modification date is being specified by the modified element from the dcterms namespace:

```
<didl:Item>
  <didl:Item>
  ...
    <didl:Descriptor>
      <didl:Statement mimeType="application/xml">
        <dcterms:modified>2006-12-20T10:29:12Z</dcterms:modified>
      </didl:Statement>
    </didl:Descriptor>
    ...
  </didl:Item>
  ...
</didl:Item>
```

Remarks

1. Declare the dcterms namespace in the DIDL tag
2. The format of the date is Zulu-time; which means that it can be sorted as text
3. There can be only one Statement element in a Descriptor element, which means that dii:identifier and dcterms:modified reside in separate Descriptor elements

Descriptor Statement: Item 'ObjectType'

The *third* descriptor contains the object type. This Object type appears on the second level of Item elements. In other words; this applies only on child Item elements of the root Item. This object type is being specified by the ObjectType element from the MPEG-21 Digital Item Processing (DIP) namespace that specifies an architecture pertaining to the dissemination of Digital Item Documents (DIDs).

```
<didl:Item>
  <didl:Item>
  ...
    <didl:Descriptor>
      <didl:Statement mimeType="application/xml">
        <dip:ObjectType>info:eu-repo/semantics/descriptiveMetadata</dip:ObjectType>
      </didl:Statement>
    </didl:Descriptor>
    ...
  </didl:Item>
  ...
</didl:Item>
```

In the section *Compound Element: representation of the complex work* the representation of the complex work this ObjectType statement will be further eleborated upon.

Remarks

1. Declare the dip namespace in the DIDL tag
2. The ObjectType in the Descriptor Statement HAS TO BE described as an URI
3. The processing architecture we use for dissemination will be for General European repositories. The URI used is placed at the info namespace as info:eu-repo. (http://info-uri.info/) Meanwhile it is used as an un-official standard within the driver community.

Compound Element: representation of the complex work

The top-Item element contains at least *two mandatory* Item element ObjectTypes. These Item-ObjectTypes are expressions of the root Item: one for the metadata and one for the digital object file, e.g. a PDF, as described by the metadata. *Optionally* there can be a third Item element ObjectType for a jump-off-page. The jump-off page is an html intermediate page that is used for human readable presentations when an Item has more than one digital object file. This situation typically occurs with theses that have separate object files (for example, when the thesis consists of a set of previously published articles). It also occurs when the content provider has a PDF, MS Word DOC and a HTML version of the same article.

```
<didl:DIDL ...>
  <didl:Item>
    <didl:Item>...</didl:Item> <!-- metadata -->
    <didl:Item>...</didl:Item> <!-- objects -->
    <didl:Item>...</didl:Item> <!-- jump-off-page -->
  </didl:Item>
</didl:DIDL>
```

The first Item contains the metadata as Unqualified Dublin Core (DC) (mandatory) which is normally used in the OAI_DC format according to the DRIVER metadata guidelines that belongs to a Digital Item Processing architecture. The second Item(s) contain links to the digital objects, and the third Item contains a link to a jump-off page.

```
<didl:Item>
  <didl:Item> <!-- one or many occurrences -->
    <didl:Descriptor>
      <didl:Statement mimeType="application/xml">
        <dip:ObjectType>info:eu-repo/semantics/descriptiveMetadata</dip:ObjectType>
      </didl:Statement>
    </didl:Descriptor>
  ...
</didl:Item>
<didl:Item> <!-- one or many occurrences -->
 <didl:Descriptor>
    <didl:Statement mimeType="application/xml">
      <dip:ObjectType>info:eu-repo/semantics/objectFile</dip:ObjectType>
    </didl:Statement>
  </didl:Descriptor>
  ...
</didl:Item>
<didl:Item> <!-- zero or one occurrences -->
  <didl:Descriptor>
    <didl:Statement mimeType="application/xml">
      <dip:ObjectType>
        info:eu-repo/semantics/humanStartPage</dip:ObjectType>
      </didl:Statement>
    </didl:Descriptor>
   ...
  </didl:Item>
</didl:Item>
```

The URI's will be processed case un-sensitive. It is recommended to use camelCase writing. It is VERY important to use the exact combinations of characters, otherwise automatic processing will not be possible. To make it very clear the following URI's are used:

- info:eu-repo/semantics/**descriptiveMetadata** (This Item occurs 1 or many times)
- info:eu-repo/semantics/**objectFile** (This Item occurs 1 or many times)
- info:eu-repo/semantics/**humanStartPage** (This Item occurs 0 or 1 time)

Remarks

- The info:eu-repo namespace is used with the following syntax: info:eu-repo/*type*/*identifier* For more information see http://info-uri.info/registry/OAIHandler?verb=GetRecord&metadataPrefix=reg&identifier=info:eu-repo/
- The semantics of the ObjectTypes mean for example that this Item states that the first sub-Item has or contains Descriptive Metadata.

ObjectType: Metadata Item

The first Item ObjectType element contains the metadata. The metadata is put in a Resource element. Every Resource element contains the namespace of a metadata format that has been used. This way the format will be recognised by service providers. According to the OAI protocol it is mandatory to use 'oai_dc'. For ease of implementation one can use the OAI_DC as metadata, since OAI_DC is a basic requirement of OAI-PMH. Every metadata item can *optionally* have its own Identifier and modified element in a Descriptor element:

```
<didl:Item>
```

```
<didl:Descriptor>
  <didl:Statement mimeType="application/xml">
    <dip:ObjectType> info:eu-repo/semantics/descriptiveMetadata</dip:ObjectType>
  </didl:Statement>
</didl:Descriptor>
```

1

```
<didl:Descriptor> <!--  This metadata instance has its own ID number -->
  <didl:Statement mimeType="application/xml">
    <dii:Identifier>info:doi/10.1705/74836724783</dii:Identifier>
  </didl:Statement>
</didl:Descriptor>
```

2

```
<didl:Descriptor> <!-- This record has its own Modification date -->
  <didl:Statement mimeType="application/xml">
    <dcterms:modified>2006-12-20T10:29:12Z</dcterms:modified>
  </didl:Statement>
</didl:Descriptor>
```

```
<didl:Component>
```

3

```
<didl:Resource mimeType="application/xml">
<!--  the DC data -->
  <oai_dc:dc
       xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
       xmlns:dc="http://purl.org/dc/elements/1.1/"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/OAI
/2.0/oai_dc.xsd">
    <dc:creator>...</dc:creator>
    <dc:creator>...</dc:creator>
    <dc:title> ... </dc:title>
    ...
  </oai_dc:dc>
</didl:Resource>
```

```
</didl:Component>
```

```
</didl:Item>
```

Remarks

1. (Mandatory when applicable) It is recommended to identify every separate component, for future reference or re-assemble purposous. This metadata set has its own identifier, which is NOT the same as the DIDL identifier.
2. If the date of the metadata has been changed, make sure the modification date of the root level Item is also being changed.
3. Declare the dc namespace in the start-tag of the Resource element where you use Dublin core.

ObjectType: Object Item

The second Item ObjectType contains a link to one digital object. This is always "by-reference" to limit the file size, when used for metadata transfer purpouses. ("by-value" is possible but increases the file size and touches the issueon ownership, use base64 encoding, not exampled here), and the Item element has an ObjectType statement with an info:eu-repo/semantics/objectFile URI. An objectFile Item can occur more than once. See the following:

```
<didl:Item> ... <!-- Below this line one can find links to one or more digital objects -->
```

```
  <didl:Item> <!-- First Item for a File/Bitstream -->
    <didl:Descriptor>
      <didl:Statement mimeType="application/xml">
        <dip:ObjectType>info:eu-repo/semantics/objectFile</dip:ObjectType>
      </didl:Statement>
    </didl:Descriptor>
    ...
    <didl:Component>
       <didl:Resource mimeType="application/pdf" ref="http://my.server.nl/report.pdf"/>
    </didl:Component>
  </didl:Item>
```

```
  <didl:Item> <!-- Second Item for a File/Bitstream -->
    <didl:Descriptor>
      <didl:Statement mimeType="application/xml">
        <dip:ObjectType>info:eu-repo/semantics/objectFile</dip:ObjectType>
      </didl:Statement>
    </didl:Descriptor>
    ...
    <didl:Component>
      <didl:Resource mimeType="application/pdf" ref="http://my.server.nl/appendix.pdf"/>
    <didl:Component>
  </didl:Item>
```

```
  <didl:Item> <!-- Third Item for a File/Bitstream -->
    <didl:Descriptor>
      <didl:Statement mimeType="application/xml">
        <dip:ObjectType>info:eu-repo/semantics/objectFile</dip:ObjectType>
      </didl:Statement>
    </didl:Descriptor>
    ...
    <didl:Component>
      <didl:Resource mimeType="application/pdf" ref="http://my.server.nl/datasheets.xls"/>
    <didl:Component>
  </didl:Item>
```

```
</didl:Item>
```

As you can see in the above example, the Resource locations do not appear in several components within one Item, but each Resource location is wrapped in an Item element. The rationale behind this is that each Bit stream of file can have its own Identifier. On the three dots "..." (given in the examples) one may place the Identifier and modified tags, which is similar to the metadata Item.

Remarks

1. The order of the object components should be in a logical reading order! The Item with chapter 1 should be followed by the next sibling Item element that contains chapter 2, etc... This way the service provider can make a better presentation. Making the order explicit by placing sequence numbers is being specified in the next version of the specification.
2. If there are important modification dates for the Resource element, propagate these date changes upwards though out the parent Item elements that encapsulate the modified child Item element.
3. Only add Identifiers when there actually are any
4. If there are no Identifiers for the ObjectType Item elements, the Identifier of the DIDL element will be used by the service provider.
5. Use for a modified or Identifier element a separate <Descriptor> <Statement> element construction
6. The rule of thumb is that if a Bitstream or file has its own identifier, the wrapper is an Item element. To keep the possibility open for a Bitstream to have an Identifier, we use the Item element as default to wrap a resource location.

ObjectType: Jump-off-page Item

The third ObjectType Item element contains a link to the jump-off page or intermediate page. This is done in the same way as for the Object Item element. Currently this is restricted to 1 Item of this type; there are no identifier elements, nor modification date elements present. This Item element is optional:

```
<didl:Item> ... <!-- Below this line; an Item with a link to one optional Intermediate page -->
```

```
  <didl:Item>
    <didl:Descriptor>
      <didl:Statement mimeType="application/xml">
        <dip:ObjectType> info:eu-repo/semantics/humanStartPage </dip:ObjectType>
      </didl:Statement>
    </didl:Descriptor>
    ...
    <didl:Component>
      <didl:Resource mimeType="application/html" ref="http://my.server.nl/mypub.html"/>
    </didl:Component>
  </didl:Item>
```

```
</didl:Item>
```