Virtual Documents in Documentum

What is a Virtual Document

Simplistically speaking Virtual document is a document that can contain another documents.
After reading the above statement first question that might arise to all is if it can contain documents how is it different from a folder?
Let me make few bullet points about Virtual Document here, which will clear most of the confusions that you might have about virtual documents.

A Virtual Document is a Document which can contain the type or the sub type of SysObjects **with an exception given below**

A Virtual Document cannot contain any folders or cabinets, or any sub types of these.

A Virtual Document object may or may not have a content (dmr_content) attached to it but a folder or a cabinet will never have any content attached to it. Though most of the time a Virtual Document might not be having a Content file attached to it.

Any Object type that extents from SysObject can be converted as a Virtual Document. The Attribute **r_is_virtual_doc (integer not a Boolean)** of SysObject determines whether that object is a virtual document or not. If the value is 1 then it's a Virtual document. If its value is 0 and the property **r_link_cnt** value is not higher than 0 then that object is not a Virtual Document

The Content Server Fundamental is misleading in this aspect, Page number 199 of guide says it's a Boolean property but in reality its not

The Contents of the Virtual document can be of different object types.

The Virtual Documents can be versioned and managed in the same way as you do with any other objects.

Components and Containment, the elements of Virtual Document

The Virtual documents are composed of various components, each components are nothing but individual objects. In other sense a component is a child of a Virtual Document. The containment objects (dmr_containtment) stores the information about the individual components of a Virtual Document. Every time when you add a new component to a virtual document a new **dmr_containtment** object will be created.

Now lets see the attributes of **a dmr_containment** object. This will give you a clear idea about what it does.

Name	Info	Description
parent_id	ID – Single	Object ID of the object that directly contains the component.
component_id	ID – Single	The Object Id of the Initial Object of the component. (I_chronicle_id). If the object has no versions, then its object ID and chronicle ID are the same.

<table>
<tr><td>copy_child</td><td>Integer – Single</td><td>Defines what a client should when the document containing the component is copied.<table><tr><th>Possible Values</th><th>Meaning</th></tr><tr><td>0</td><td>The decision whether to copy or reference the component is left to the user when the document is copied</td></tr><tr><td>1</td><td>When the document is copied, the component is referenced In the new copy rather than actually copied</td></tr><tr><td>2</td><td>When the document is copied, the component is also copied</td></tr></table></td></tr>
<tr><td>follow_assembly</td><td>Boolean - Single</td><td>If set to TRUE, directs the system to resolve a component using the component’s assembly</td></tr>
<tr><td>i_partition</td><td>Integer – Single</td><td>Not Currently used</td></tr>
<tr><td>order_no</td><td>Double -Single</td><td>Number representing the component’s position within the components of the virtual document identified by parent_id</td></tr>
<tr><td>a_contain_desc</td><td>string(255) Single</td><td>User-defined. Used by clients to manage XML documents</td></tr>
<tr><td>a_contain_type</td><td>string(255) Single</td><td>User-defined. Used by clients to manage XML documents</td></tr>
<tr><td>use_node_ver_label</td><td>Boolean - Single</td><td>If set to TRUE for early-bound components, the server uses the early-bound symbolic label to resolve late-bound descendants of the component during assembly</td></tr>
<tr><td>Version_label</td><td>string(32) Single</td><td>Version label for the component.</td></tr>
</table>

Now as you had seen the dmr_containment object attributes lets get into more details of Virtual Documents.
The component can be associated with Virtual documents in 2 ways with respect to its versions
You can either associate a particular version of component with the virtual document or you can attach the entire version tree of the component with Virtual document.

If you attach the entire version of a document you have a choice of selecting the version of component to attach with virtual document at the time of assembling a virtual document.
Another important part of the Virtual document is the order of components in the Virtual Document; Content Server manages this ordering by default by adding for removing numbers while adding or removing components. If you wish do this part you can do it manually too.

Component referential integrity

The Boolean property *compound_integrity* of server configuration object **dm_server_config** manage the referential integrity of virtual document. If this attribute is set to true the content server will not allow deleting an object, which is contained in a Virtual Document. This attribute is set to true by default. You have to have minimum *SysAdmin* privileges to change this value.
If this attribute is set to false one can destroy the components of an unfrozen Virtual Documents. In any case you can never destroy components of a frozen virtual document.

Assembly, Conditional Assembly and Snapshots

Selecting the set of components for a Virtual document for a operation is known as Assembly, Conditional assembly allows the user to choose to include all components or some of them. As mentioned above if the entire versions of a component we can choose which all documents and also which versions of document to be included in the assembly.

A snapshot is the state of a Virtual document at the time when it was taken. This means it has the components that were selected during the creation of the snapshot. Each Snapshot is saved in repository as an assembly (**dm_assembly**) object. Assembly objects are created when a user creates an assembly, a snapshot of the virtual document at particular point in time. Users must have at least Version permission for the object identified in the book_id property to modify an assembly object.

Binding a component (Early and Late)

Attaching a component to a Virtual Document is known as binding. There are 2 ways of binding a component to Virtual Document. They are **Early binding** and **Late Binding**

1) **Early Binding**
   If you bind a specific version of a Component to the Virtual Document when you create a virtual document then it's known as Early Binding. This makes sure that all the snapshots of virtual document have the same version of the component.
   Links – Absolute and Symbolic
   Absolute Link
   Before getting into Absolute links let's see what an **implicit version label** is? ***An Implicit version label is the version label assigned to a document when you version by the content server***. An implicit version label of the version of a document remains as it till that version is destroyed. Using an implicit version label to link component with a Virtual Document creates an Absolute link between component and Virtual Document.
   Symbolic Link
   Symbolic version label is the version label, which are user –defined. This allows a user to give meaningful labels to a document. This can be moved from one version to another of a object.
   When you use this type of linking no matter what the Virtual Document will have component, which has the specified symbolic version label. E.g. You define a Virtual Document to have a component which has a version label Published, the Virtual document will always have the published version of the component.
2) **Late Binding**
   When you don't specify a version label of a component when u add it to a virtual

document its known as late binding. In this case the Content server attach the entire version tree of the specified component to the virtual document. And when you assemble the document you specify which version of component you should use.

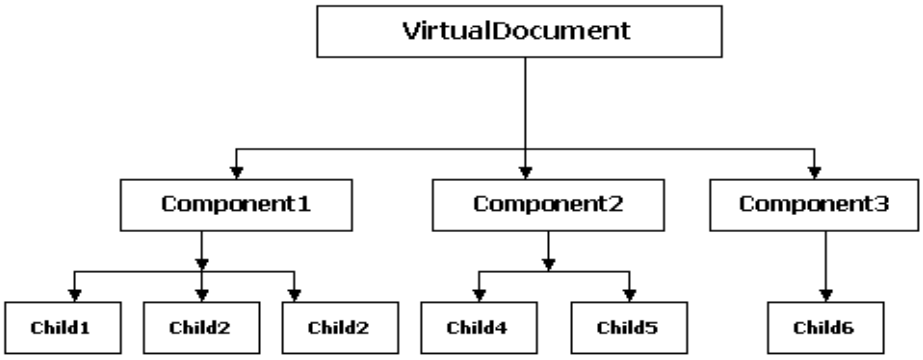
Freezing and Unfreezing a Virtual Document.

Frozen Virtual Documents are virtual documents, which are immutable. Once you make a Virtual Document or a snapshot frozen you cannot change or delete any attributes of the virtual document and you can also not add or remove components of virtual document. You can explicitly make a Virtual Document immutable by calling IDfSysObject.freeze()
When you call the freeze () **r_immutable_flag** will be set to true by content server also **r_frozen_flag** will also be set to true by the content server.

Calling IDfSysObject.unfreeze() on a frozen virtual document unfreezes the Virtual Document and make it modifiable again. In that case the content server sets the above-mentioned flags to false.

Virtual Document related DQLS

Consider this model All the queries explained below will be based on this model



To find all direct components in this Virtual document

```
SELECT 'object_name' in dm_sysobject IN DOCUMENT ID
('r_object_id_of_virtual_document')
```

This will returns the following in the exact same order VirtualDocument, Component1, Component2 and Component3 (**VirtualDocument is returned because Virtual Document itself is considered as a Component**)

To Filter Object Types in the above Query

```
SELECT r_object_id in dm_document IN DOCUMENT ID
('r_object_id_of_virtual_document')
```

This will return only dm_documents in the Virtual Document specified

Usage of DESCENT
Descent is used to return all components that contained in a virtual document.

```
SELECT 'object_name' in dm_sysobject IN DOCUMENT ID
('r_object_id_of_virtual_document') DESCENT
```

This returns the components in the following order
VirtualDocument, Component1, Child1, Child2, Child3, Component2, Child4, Child5, Component3, Child6,

Use the IN DOCUMENT clause with the ID scalar function to identify a particular virtual document in the query. The keyword DESCEND searches the virtual document's full hierarchy.

Usage of VERSION
The VERSION Keyword finds the components of a specific version of a virtual document.

```
SELECT 'object_name' in dm_sysobject IN DOCUMENT ID
('r_object_id_of_virtual_document') VERSION 1.1
```