#	Sec.	EUMETSAT Comment	GMV answer
1	2	Is a mapping foreseen between the schema packages and their contents? I.e., how would an application, which is to create a new EO Product or Auxiliary Package, know which package (and thus containing file) to reference for its representation information, i.e. how does this application identify which logical name to refer to?	No mapping is foreseen. Note that the relationship between the schema package and its files is performed through the Manifest file of this package. Anyway, certain knowledge of the relationship between packages is assumed when the products are archived i.e. the representation information should be stored in the archive before the EO Product /EO Auxiliary package. Therefore this reference could be driven by a configuration file developed for each specialisation.
2	3	I cannot see that a schema file stored in a package can become easily obsolete. It does this only when ultimately all references to it infer that this is the new applicable version. How can this be ensured? Example: EO Product Av1 refers to e.g. DFLD schema Dv1. As described, if the version of an EO Product changes, a new EO Product needs to be archived, let's call it Av2. This one now wants to reference DFLD schema Dv2. Both DFDL schemas are in the same package, without any version information. By adding Dv2 one invalidates Dv1 and automatically Av1 – and this is mostly, IMO, not what one wishes. Instead, both should still be usable. The proposed versioning schema prohibits this. I.e., as described in the document "Obsolete schema version...", I don't really think those become obsolete, they simply become not the most recent but might be still applicable for "older" versions of products, unless one really plans to replace something, e.g. because of errors... I propose to rethink this approach. From the top of my head one could include version information in filenames, or via directories inside the SAFE packages. e.g. .../3.2/schema.xsd. I seem to remember as well that SAFE asked for identifying applicable versions in references to internal or externally kept files, and this is common practice.	Perhaps "obsolete" is not the best word to use in this context because it can lead to confusion. Therefore, the "old" word should be used instead of "obsolete" because the DFDL schema could be still usable by others products. The idea to include the file version in the filename is a good approach for us too. It can be added as mandatory before the ISO8601.
3	3	ISO 8601 requires the "T" as a separator between date and time! The proposed naming extension for renamed files is not ISO 8601 compliant.	Agree. The "T" was not introduced by mistake but it was initially considered in an early stage of the document. The file naming criteria will be reviewed accordingly.
4	3	The logical identifier for files inside packages is not understood. The proposal reads that a file is identified only by a number. I fail to understand how this number maps to a physical file. What is the mapping applied here? Sort by date/time, sort by filename, other? So I don't really understand how one would be able to refer from one, say, EO package to an externalised file. Obviously one would need the package-id as well as the file-id. If this is solely managed by the mapper as one can see in the example as of section 5.1, IMO, a certain risk is introduced. I propose to identify both packages and names by logical identifiers which are somehow meaningful. The given example in section 5.1 as well places a big burden on the mapper as it requires to map not only physical positions of packages but as well of single files. I would expect	The number used in the URN is a simple way to identify a file inside a package. This number does not locate a physical file because the URN<->Location mapper is used for this instead. The only restriction is that the number assigned for each file should be unique inside the package. With respect to your approach "…to transfer the role of mapping the filenames to the manifest of the package itself…" In this

		that this is over the top and that it can be solved differently, as each single file can be identified inside a package, e.g. by its (file)name. An alternative approach could be, e.g., to transfer the role of mapping the filenames to the manifest of the package itself, i.e. in the manifest of a referenced package it is defined how a logical identifier of a file used elsewhere is related to a physical file inside a package. This would be enough as it is not planned to move single files of packages around. The atomic entity wrt. archiving is "package" and not "file".	case, the reference would be performed via URN package (without file identifier). Therefore, if there are several "old" versions of the same schema in the package… It is not possible to identify which is the applicable schema version (or old version) for this file. However, the file identifier would not be needed if we assume that the schema versions are backward compatible (although, we consider that this assumption can be hardly done for operational missions). Please, consider the following cases: LTDP case: - Different versions of DFDL and metadata schemas can be assumed to be backward compatible Operational case: - DFDL: if the structure one product type is changed, it is assumed that it will be considered as a different product type. In this case, each version of the DFDL can be assumed as backward compatible. - If different schemas versions are adopted, it is under responsibility of the archive to assure that the new schema is backward compatible. One metadata schema version applying to one product type, but it could not be applicable to a "new" product of the same type.
5	3	In section 4.3.2 it is said that the mapper should be able to associate the logical identifiers of several packages, i.e. implement a 1..* relationship. I fail to understand why this is necessary. In my understanding the references from e.g. EO package to AUX data is done via the manifest of the EO package only, by using the logical identifiers as described.	One of the objectives of the current design is to allow the addition of Auxiliary files (associated to an EO Product SAFE package) once that SAFE package has been created, without changing the archived EO Product package. Consider for example, that you have received the mandate to preserve a set of auxiliary files in your archive. Following your approach, the manifest of the EO Packages should include a logical reference to each one of those auxiliary files (packages) which is not a problem at all. However, if in the future you have a new mandate to preserve an additional auxiliary file type (not

			considered previously) you should update the manifest of all EO Products packages to include a link to reference the new file in the archive… and this modification is exactly what we wanted to avoid.
6	3	The examples given in section 4.3.3 are wrong if one would take the comments made above into consideration and would need to be reworked.	Agreed.
7	5	The examples need to be revisited if any of the above made comments will be considered for changing the presented specification/design.	Agreed.