

Technical Note Concerning Space Data System Standards

|  |
| --- |
| XFDU in TGFT |

Draft Technical NOTE

CSSA 927.1-TN-0.3

February 2017

DOCUMENT CONTROL

|  |  |  |  |
| --- | --- | --- | --- |
| **Document** | **Title and Issue** | **Date** | **Status** |
| CSSA927.1-TN-0.1 | XFDU in TGFT | February 2017 | first draft. |
| CSSA927.1-TN-0.2 | XFDU in TGFT | February 2017 | Adds diagram for top-level XFDU structure as modified for TGFT. |
| CSSA927.1-TN-0.3 | XFDU in TGFT | February 2017 | Adds construction rules for TGFT XFDUs, example in Annex A |

CONTENTS

Section Page

[DOCUMENT CONTROL i](#_Toc476079431)

[CONTENTS ii](#_Toc476079432)

[1 Introduction 1-1](#_Toc476079433)

[1.1 Purpose OF THIS TecH Note 1-1](#_Toc476079434)

[1.2 Background 1-1](#_Toc476079435)

[1.3 Scope 1-1](#_Toc476079436)

[1.4 Document Organization 1-1](#_Toc476079437)

[1.5 Terminology 1-1](#_Toc476079438)

[1.5.1 Definsitons from XFDU Specification 1-1](#_Toc476079439)

[1.6 References 1-1](#_Toc476079440)

[2 Concept for using XFDUs in TGFT 2-1](#_Toc476079441)

[2.1 Use of XFDU by TGFT 2-1](#_Toc476079442)

[2.2 Derivation of the TGFT XFDU from the CCSDS-Standard XFDU 2-2](#_Toc476079443)

[2.3 SummAry of XFDU Concepts and Capabilities Used by TGFT 2-2](#_Toc476079444)

[2.4 Analysis of XFDU Element and Attribute Applicability to TGFT 2-4](#_Toc476079445)

[2.4.1 XFDU Manifest Attributes 2-6](#_Toc476079446)

[2.4.2 Package Header 2-7](#_Toc476079447)

[2.4.3 Information Package Map 2-10](#_Toc476079448)

[2.4.4 data Object Section 2-15](#_Toc476079449)

[2.4.5 Metadata Section 2-22](#_Toc476079450)

[2.4.6 Behavior Section 2-29](#_Toc476079451)

[2.5 XFDU PAckage File formats 2-29](#_Toc476079452)

[2.6 Applicability of TGFT XFDU Composition rules to services that use TGFT 2-30](#_Toc476079453)

[3 Composition Rules for XFDU Packages for TGFT 3-1](#_Toc476079454)

[3.1 General 3-1](#_Toc476079455)

[3.2 XFDU Manifest 3-1](#_Toc476079456)

[3.2.3 XFDU Manifest Attributes 3-2](#_Toc476079457)

[3.2.5 informationPackageMap element 3-2](#_Toc476079458)

[3.2.6 metadataSection element 3-3](#_Toc476079459)

[3.2.7 dataObjectSection Element 3-6](#_Toc476079460)

[3.3 Payload Data File enclosed within the XFDU Package 3-7](#_Toc476079461)

[3.4 Metadata File enclosed within the XFDU Package 3-7](#_Toc476079462)

[3.5 TGFT XFDU Package 3-7](#_Toc476079463)

[ANNEX A Example XFDU Manifest A-1](#_Toc476079663)

[ANNEX B Reference Bookmarks (to be deleted) B-1](#_Toc476079664)

Figure Page

[**Figure 2‑1: TGFT/Cross Support Service Entity Relationships** 2-1](#_Toc476079464)

[**Figure 2‑2: Top-Level XFDU Structure with TGFT Modifications** 2-3](#_Toc476079465)

[**Figure 2‑3: CCSDS-standard Top-Level XFDU Structure in XMLspy Graphical Representation** 2-5](#_Toc476079466)

[**Figure 2‑4: Components of the Top-Level XFDU Structure Used by TGFT, in XMLspy Graphical Representation** 2-6](#_Toc476079467)

[**Figure 2‑5: CCSDS-standard PackageHeaderType in XMLspy Graphical Representation** 2-8](#_Toc476079468)

[**Figure 2‑6: PackageHeaderType Components Used by TGFT,in XMLspy Graphical Representation** 2-10](#_Toc476079469)

[**Figure 2‑7: CCSDS-standard InformationPackageMapType in XMLspy Graphical Representation** 2-11](#_Toc476079470)

[**Figure 2‑8: InformationPackageMapType Components Used by TGFT,in XMLspy Graphical Representation** 2-12](#_Toc476079471)

[**Figure 2‑9: CCSDS-standard DataObjectSectionType in XMLspy Graphical Representation** 2-16](#_Toc476079472)

[**Figure 2‑10: DataObjectSectionType Components Used by TGFT, in XMLspy Graphical Representation** 2-18](#_Toc476079473)

[**Figure 2‑11: CCSDS-standard MetadataSectionType in XMLspy Graphical Representation** 2-23](#_Toc476079474)

[**Figure 2‑12: MetadataSectionType Components Used by TGFT, in XMLspy Graphical Representation** 2-24](#_Toc476079475)

Table Page

[A-1 Example XFDU Manifest A-2](#_Toc476079616)

# Introduction

???

## Purpose OF THIS TecH Note

To define the subset of XFDU capabilities that must be used to package payload data and metadata for transfer via CCSDS TGFT.

## Background

???

## Scope

???

## Document Organization

???

## Terminology

### Definsitons from XFDU Specification

1. Data Object
2. [more]

## References

The following documents are referenced in this Report. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Report are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS documents.

[1] *XML Formatted Data Unit (XFDU) Structure and Construction Rules*. Recommendation for Space Data System Standards, CCSDS 661.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, September 2008.

[2] Terrestrial Generic File Transfer. Recommendation for Space Data System Standards, CCSDS 927.1-W-x. White Book. Issue x. Washington, D.C.: CCSDS, [TBD].

[3] *Reference Model for an Open Archival Information System (OAIS)*. Recommendation for Space Data System Practices, CCSDS 650.0-M-2. Magenta Book. Issue 2. Washington, D.C.: CCSDS, June 2012.

# Concept for using XFDUs in TGFT

## Use of XFDU by TGFT

TGFT provides an underlying service to a class of cross support services by which the data exchanged between service provider and service user is in the form of a file, possibly accompanied by the metadata necessary to interpret that file.

There are two TGFT entities: the *File Source* and the *File Recipient*. Because TGFT is a push-only protocol, the transmission of the file is always initiated by the File Source. The triggering mechanism for the initiation of the file transfer is defined by the cross support service that is using TGFT. The TGFT entities have one of two types of relationships with the service that uses it (TGFT). In the *provider source* configuration, the Service Provider has the data, and it uses TGFT to push the XFDU Package containing that data to the Service User. The transfer of Delta-DOR files from the Delta-DOR service provider to the user is an example of the provider source configuration. In the *user source* configuration, the Service User has the data, and it uses TGFT to push the XFDU Package containing that data to the Service Provider. The transfer of spacecraft-destined files from the Forward File service user (in the MOC) to the Forward File service provider (n the ground station) is an example of the user source configuration. Figure 2‑1 illustrates these two configurations.



**Figure 2‑1: TGFT/Cross Support Service Entity Relationships**

As illustrated in the figure, it is the service entity that sits atop the File Source that generates the XFDU Package, and the service entity that sits atop the File Recipient that receives and processes the XFDU Package.

EDITOR’S NOTES

1 Does TGFT do anything to enforce any part of the format requirements? E.g., what if a File Source tried to send a file to the File Recipient that had an extension other than “.zip” or “.tar”?

2 According to the TGFT White Book (reference [2]), all services using TGFT will share the same file directories (“in-tray” and “out-tray”). How are the files distinguished for the purposes of directing the files to their appropriate services (i.e., applications)? The TGFT White Book currently puts no semantic requirements on the names of the files transferred by TGFT, but one possible approach would be to create a file-naming convention that reflected the TGFT-using service, and a corresponding registration mechanism for the service-specific file name substrings. Another approach would be to mandate the use of the (currently optional) packageType attribute of the informationPackageMap component of the XFDU Manifest (see 2.4.3.1 (2) and Figure 2‑8), where the values for packageType are registered. For this draft of this Tech Note, the mandatory use of the packageType attribute with registered values is assumed.

## Derivation of the TGFT XFDU from the CCSDS-Standard XFDU

The CCSDS-standard XFDU (reference [1]) has been designed with data archiving in mind. It therefore has features and capabilities that support transferring data into an archive, storing that data with its associated metadata for timespans up to years or even decades, and eventually making that data available to users, possibly in groupings, formatting, and encodings of the data that are different from those that were used in the acquisition of the data by the archive, and possibly drawn from multiple physical data repositories.

The needs of Terrestrial Generic File Transfer are only a subset of those needed for the larger data acquisition/archival/dissemination problem that the CCSDS-standard XFDU is designed to address. For instance, much of the complexity of the CCSDS-standard XFDU results from maintaining data relationships of “data objects” that are larger than individual files, whereas TGFT deals with an individual payload data file and the metadata associated with it.

As a result, the TGFT XFDU is a subset of the CCSDS-standard XFDU, with the elements needed to support the file transfer needs of cross support services for the use cases identified in the (currently draft) TGFT Specification (reference [2]).

The specification of XFDU capabilities for TGFT does not preclude TGFT-using services from using additional XFDU mechanisms, either optionally or to suit additional requirements of those TGFT-using services.

## SummAry of XFDU Concepts and Capabilities Used by TGFT

This section provides an overview of the CCSDS-standard XFDU, the applicability of the various aspects of that XFDU, and the rationale for restrictions, modifications, and/or refinements for the purposes of TGFT.

In general, a CCSDS XFDU is a logical entity that comprises “an XFDU Manifest, all files contained in the manifest, and all files and XFDUs referenced from it. Some or all of the referenced files may be contained in an XFDU Package, such as through the use of a ZIP file. However, there may still be references in the Manifest to files outside the XFDU Package. In this case, the XFDU is a logical entity and does not exist as a single physical entity.” (reference [1]).

EDITOR’S NOTE – Table 3-1 of [1] states that metadata/data may be included in the Manifest as base64 or XML. Presumably this would be in the Data Object Section of the Manifest. The Manifest schema shows a base64 byteStream:fileContent:binaryData element that appears to be the only possible location for an embedded binary file, but the description for that element states that it contains “base64-encoded *metadata*”. This appears to be a typo.

Figure 2‑2 illustrates the top-level structure of the XFDU Package with the simplifications that apply to TGFT.



**Figure 2‑2: Top-Level XFDU Structure with TGFT Modifications**

TGFT puts a further constraint on the structure of an XFDU in that the data (hereinafter referred to as the *payload data* to differentiate it from metadata) must be carried within the XFDU Package that is being transferred via TGFT. That is, TGFT cannot be used to transfer mere pointers to payload data. This constraint is a consequence of TGFT being defined as a push-only mechanism.

Designers of a TGFT-using service must decide whether the XFDUs for that particular service must embed the payload data XFDU Manifest itself, place it in a separate file within the same XFDU Package as the XFDU Manifest, or some combination of the two. If both options are supported by a service, the service specification must define the criteria by which the choice is made. Note that a TGFT-using service specification may allow File Source implementations to use either option, in which case the File Recipient implementations must be capable of processing both options.

A TGFT-compliant XFDU Manifest *may* contain references to metadata that exists outside the XFDU Package that contains the XFDU Manifest. The rationale is that some metadata (such as XML schema) may persist and be applicable to many instances of payload data, such that it is an inefficient use of resources to transfer the metadata with every instance of payload data transfer.

As with the payload data, the specification of a TGFT-using service must address which of the three possible options (embedded in the XFDU Manifest, enclosed in the same XFDU Package, or resident in an external repository) are to be used for the metadata the XFDU Manifests and XFDU Packages for that service. Because it is possible for multiple metadata files to be associated with one payload data file, some TGFT-using services may use multiple methods. For example, the payload data file may be formatted as an XML document using a standard (for that particular enterprise) XML schema, so the reference to that XML schema file would be an external reference. But the same payload file might also have metadata that applies to only that payload data file, and so that metadata file would be either embedded in the XFDU Manifest or enclosed in the same XFDU Package.

## Analysis of XFDU Element and Attribute Applicability to TGFT

Figure 2‑3 is the XMLspy graphic representation of the top-level CCSDS-standard XFDU Manifest schema. Figure 2‑4 is the corresponding diagram of only those top-level components that are used in TGFT.



**Figure 2‑3: CCSDS-standard Top-Level XFDU Structure in XMLspy Graphical Representation**



**Figure 2‑4: Components of the Top-Level XFDU Structure Used by TGFT, in XMLspy Graphical Representation**

NOTE - The mechanism by which the file recipient accesses such out-of-band metadata is outside the scope of the TGFT specification. For some TGFT-using services the metadata access mechanism could simply be defined bilaterally. Other TGFT-using services may define standard mechanisms by which metadata is made available to the file recipient, e.g., by always pushing out new versions of the metadata whenever they become available. However, such service-specific mechanisms would be defined by the specification of those services and not part of TGFT.

### XFDU Manifest Attributes

The XFDU Manifest conforms to the XFDUType defined in [1]. XFDUType has four attributes:

1. **ID** (of type xsd:ID): This optional attribute is not defined in [1]. This attribute might be intended for use in cross referencing across different XFDU Manifests, which is outside the scope of TGFT. Therefore, this attribute has no defined purpose for TGFT.
2. **objID** (of type xsd:string): This optional attribute is defined in [1] as “a primary identifier assigned to this XFDU instance by the producer of the XFDU.” Therefore, this attribute has no defined purpose for TGFT.
3. **textInfo** (of type xsd:string): This optional attribute is defined in [1] as “a title/text string identifying the document for users” For TGFT, whether the optional **textInfo** attribute is used and the value(s) that it carries are defined by the specification of the service that uses TGFT.
4. **version** (of type xsd:string): This optional attribute is defined in [1] as “version of the XFDU XML Schema this XFDU should be validated against. Currently this is a string but when formal CCSDS XML Schema Naming and Versioning rules are defined it is expected that this type will be specialized to conform to those rules.” Unfortunately, there is no indication in [1] what the current version is (presumably version 1, but is that coded as “1”, “1.0”, “one”, etc.?). Furthermore, as noted in the definition itself, there are not yet formal XML Schema Naming and Versioning Rules, so the conditions that control the content of this attribute are currently unknown. For TGFT, absence of the version attribute implies conformance to the version as specified in Issue 1 (September 2008) of [1]. If and when additional version of the XML schema become available in the future, and assuming that by then Schema Naming and Versioning rules will be in place to unambiguously define the corresponding contents of the version attribute, the version attribute will be required in TGFT XFDUs to specify which version is being used.

### Package Header

According to reference [1]:

“The ***Package Header*** (packageHeader element of packageHeaderType) is an XML Complex Type that contains metadata that apply to the whole XFDU Package. These metadata may include data to inform the XFDU parsing software about volume metadata (e.g., logical volume information and specification version), administrative metadata (e.g., author and creation data) and technical data (e.g., hardware and operating system).

“The package header type has two elements:

* environmentInfo—contains application [service] specific information defined either by an extension of the XFDU Schema or by freeform XML;
* volumeInfo—contains XFDU volume-related metadata such as XFDU specification version and logical volume sequence information.”

Figure 2‑5 is the XMLspy graphical representation of the CCSDS-standard InformationPackageMapType schema type.



**Figure 2‑5: CCSDS-standard PackageHeaderType in XMLspy Graphical Representation**

The packageHeader element is optional, according to [1]. The packageHeader element should be used in TGFT XFDUs to carry information about user and provider identification, etc., in the environmentInfo element (see XXX).

The packageHeader has one attribute, **ID** (of type xsd:ID): This is a required attribute so it must be present in the packageHeader, but its usefulness is minimal since there is only one Package Header in the XFDU Manifest. The service using TGFT may specify semantics for this attribute, but the default value is “PkgHdr”.

#### volumeInfo element

The volumeInfo element has two component elements: specificationVersion and sequenceInformation.

The required specificationVersion element “specifies the version of the XFDU specification to which this manifest complies.” It is defined as of type xsd:string in [1]. Technically, CCSDS “specifications” (e.g., Recommended Standards) are updated by “Issue”, not “Version”, and presumably this refers to the specific Issue of the *XML Formatted Data Unit (XFDU) Structure and Construction Rules* (reference [1]). As of this writing, the current Issue of [1] is “1.0”.

EDITOR’S NOTE - The example XML snippet in [1] that includes the specificationVersion element has a value of “1.0”. However, full Recommended Standards do not carry decimal places in the Issue numbers. Presumable the existence of the decimal place is there to accommodate with intermediate draft versions of Recommended Standards.

The optional sequenceInformation element is not used for TGFT.

#### environmentInfo element

The environmentInfo element has two component elements: xmlData and extension.

The xmlData element is optional in the CCSDS-standard XFDU, but it is required in the TGFT XFDU if the XFDU contains a packageHeader element. The xmlData element contains service-specific information the is needed or helpful in identifying the context of the data (e.g., provider and user identification). The specific content of this attribute is defined in the specification of the TGFT-using service.

The extension element is optional in the CCSDS-standard XFDU. The extension element is not used in TGFT XFDUs.

Figure 2‑6 shows the components of the PackageHeadertType schema type that are used by TGFT.



**Figure 2‑6: PackageHeaderType Components Used by TGFT,in XMLspy Graphical Representation**

### Information Package Map

According to 4.1 (b) of reference [1] the mandatory **“Information Package Map** (informationPackageMap element of informationPackageMapType): provides a hierarchical view of the content of the XFDU using a series of nested **contentUnit** elements. Content Units contain pointers to data objects and to the metadata associated with those data objects.” I.e., the Information Package Map is the “table of contents” for XFDU.

Figure 2‑7 is the XMLspy graphical representation of the CCSDS-standard InformationPackageMapType schema type. Figure 2‑8 shows the components of that schema type that are used by TGFT.



**Figure 2‑7: CCSDS-standard InformationPackageMapType in XMLspy Graphical Representation**



**Figure 2‑8: InformationPackageMapType Components Used by TGFT,in XMLspy Graphical Representation**

**informationPackageeMap attributes**. The informationPackageMap element contains four attributes:

1. **ID** (of type xsd:ID): This optional attribute allows an ID to be assigned to the informationPackageMap. Because each XFDU Manifest contains only one informationPackageMap, this ID provides no additional information within the XFDU Manifest and its purpose is not specified in [1]. Therefore, there is no defined use for this ID in TGFT XFDUs.
2. **packageType** (of type xsd:string); According to [1], “a type for the object.” The “typical values” and only examples provided in the documentation for this optional attribute in in [1] are ones that are to pertinent to data archives that are defined in the context of the OAIS Reference Model (reference [3]). For TGFT, the **packageType** attribute is required. The values that are used in packageType are registered in a TBD SANA registry and indicate services that uses TGFT (e.g., “ForwardFile” for the Forward File service).
3. **textInfo** (of type xsd:string); According to [1], “a string to describe the informationPackageMap to users.” There is no defined TGFT use for this attribute, because there is a one-to-one relationship between the XFDU Manifest as a whole and the informationPackageMap element, and therefore whatever textual description is needed can be included in the textInfo attribute of the XFDU Manifest.
4. **anyAttribute**: According to [1], “wild-carded attribute extension point.” There is no defined use for this extension point for TGFT XFDU Manifests, but individual TGFT-using services may specify such extensions and their associated semantics.

**Content Units**. In general, the CCSDS-standard XFDU Manifest Information Package Map contains one of more Content Units. As defined in 6.1 of reference [1]:

“A ***Content Unit*** (contentUnit element of contentUnitType) is the basic structural unit of the XFDU. Content Unit elements may include other Content Units, may be internal pointers to elements in the Data Object section or may be external pointers to other XFDUs. Therefore a Content Unit can be used to associate a Data Object with one or more Metadata Objects, and multiple Content Units can present a hierarchical view of these data/metadata associations.”

However, for TGFT the composition of the Information Package Map and Content Unit are significantly simplified:

* The Information Package Map contains only one Content Unit, which contains one Data Object Pointer representing a single payload file Data Object.
* There is no recursive containment of Content Units.
* Data Objects (which Content Units point to) are not used to contain or refer to metadata.

The CCSDS-standard XFDU Manifest allows recurrent containment of Content Units and containment of metadata in Data Objects in order to support arbitrarily-complex relationships among various aggregations of payload data and metadata. The simplifications supported by the TGFT XFDU composition rules allow the single payload file to be accompanied (physically or be reference) by up to four kinds of metadata files – three of which are pre-defined by the XFDU Blue Book and the fourth essentially representing any other form of metadata that doesn’t fall into one of the three predefined categories (these categories are explained later in this Tech Note).

**Content Unit attributes.** The CCSDS-standard Content Unit has nine (9) attributes:

1. **ID** (of type xsd:ID). Because the CCSDS-standard XFDU Manifest accommodates multiple Content Units, this ID is used to differentiate among them. However, since the TGFT XFDU Information Package Map has only one Content Unit, this attribute is not required in TGFT XFDUs.

NOTE - The Behavior Section (see 2.4.6) uses the Content Unit IDs to associated behaviors with individual Content Units, but since the Behavior Section is not used in TGFT XFDUs, the absence of the Content Unit ID attribute is not a problem.

1. **order** (of type xsd:string). The attribute is available to represent the order (sequence) among the Content Units in the Information Package Map. Because the TGFT XFDU Information Package Map has only one Content Unit, the ‘order’ attribute is NOT used in TGFT XFDUs.
2. **unitTyp**e (of type xsd:string). As defined in 6.2 of reference [1], “a type of content unit (e.g., Application Data Unit, Data Description Unit, Software Installation Unit, etc.)”. There do not appear to be any pre-defined values for the content of this optional attribute. Therefore, if used, the content of this attribute must be defined by the TGFT-using service.
3. **textInfo** (of type xsd:string). As defined in 6.2 of reference [1], “a string label to describe this contentUnit to an end user viewing the document, as per a table of contents entry.” If used, the content of this optional attribute must be defined by the TGFT-using service.
4. **repID** (of type xsd:IDREFS). If there are any associated metadata files that are categorized as Representation Information (see 2.4.5.5), this attribute contains the IDREF values of the corresponding metadataObjects in the Metadata Section (see 2.4.5) component of the XFDU Manifest.
5. **dmdID** (of type xsd:IDREFS). If there are any associated metadata files that are categorized as Descriptive Information (see 2.4.5.5), this attribute contains the IDREF values of the corresponding metadataObjects in the Metadata Section (see 2.4.5) component of the XFDU Manifest.
6. **pdiID** (of type xsd:IDREFS). If there are any associated metadata files that are categorized as Preservation Information (see 2.4.5.5), this attribute contains the IDREF values of the corresponding metadataObjects in the Metadata Section (see 2.4.5) component of the XFDU Manifest.
7. **anyMdID** (of type xsd:IDREFS). If there are any associated metadata files that cannot be categorized as Representation, Descriptive, or Preservation Information (see 2.4.5.5), this attribute contains the IDREF values of the corresponding metadataObjects in the Metadata Section (see 2.4.5) component of the XFDU Manifest.
8. **behaviorID** (of type xsd:IDREF). The CCSDS-standard XFDU Manifest allows a single behaviorObject to be associated with each Content Unit, and if such an associated behaviorObject exists, the bahaviorID contains the ID of that behaviorObject. However, since the Behavior Section is not used in TGFT XFDUs (see 2.4.62.4.6, this attribute does not appear in the TGFT XFDU Manifest.

NOTE - The type xsd:IDREFS is defined as a list of IDREFs, formatted as a string of whitespace-separated IDREFs. Thus a single atrribute of type xsd:IDREFS can point to any number of components, each of which has an ID.

**Data Object Pointer.** The Data Object Pointer (dataObjectPointer element) points to a Data Object (i.e., a payload data file) within the same XFDU Package.

The CCSDS-standard XFDU Manifest allows one Content Unit to point to zero or more Data Objects (zero in the case where the Data Objects reside in different XFDU Packages – see 2.4.3.2.3).

For TGFT, the Content Unit points to one and only one Data Object.

The dataObjectPointer element has two attributes:

1. **ID** (of type xsd:ID): an XML ID for the element. There is no defined use for this optional attribute for TGFT XFDUs.
2. **dataObjectID** (of type xsd:IDREF): the value of the **ID** attribute of the dataObject element being pointed to. This attribute is required.

The dataObjectPointer element itself is empty – its information content is carried in the dataObjectID attribute.

**XFDU Pointer**. The CCSDS-standard XFDU Manifest allows one Content Unit to point to Data Objects that reside in XFDUs outside the XFDU Package that contains the XFDU Manifest. However, for TGFT XFDU, all Data Objects (payload data) must reside within the same XFDU Package as the XFDU Manifest, so the XFDU Pointer is NOT used for TGFT.

**Content Unit extension element.** The CCSDS-standard XFDU Manifest allows for the addition of extensions of the XFDU Content Unit from a separately-controlled namespace. There are no Content Unit XFDU extensions defined for TGFT XFDUs, although individual TGFT-using service may add them if required by those services.

### data Object Section

Payload data is contained in, or referenced by, the dataObjectSection component of the XFDU Manifest.

Figure 2‑9 is the XMLspy graphical representation of the CCSDS-standard DataObjectSectionType schema type.



**Figure 2‑9: CCSDS-standard DataObjectSectionType in XMLspy Graphical Representation**

**CCSDS-standard XFDU Data Object Section**. In the CCSDS-standard XFDU Manifest, the dataObjectSection of the XFDU Manifest can contain one or more dataObjects, each of which can contain one or more byteStreams. The byteStreams are a mechanism for breaking a large(r) data entity (the Data Object) into small(er) physical files (the byteStreams). The current XFDU specification defines only one combination relationship among the byteStreams that comprise a dataObject – concatenation (combinationName attribute = ‘concat’). Each dataObject element may also contain an (optional) checksum over the contents of the Data Object, and one or more (optional) transformObjects (described below).

The XFDU specification (reference [1]) allows each byteStream to contain (a) (optionally) zero or more fileLocation elements (which point to payload data files that are not contained within the Manifest), (b) (optionally) one fileContent element, which contains either base64 binaryData or xsd:any xmlData, and (c) (optionally) a checksum over the contents of the byteStream.

Thus the CCSDS-standard XFDU specification allows multiple byteStream files to comprise a single Data Object, and the contents of those byteStreams can be (a) embedded within the XFDU Manifest itself, (b) external to the XFDU Manifest but contained within the same XFDU Package as the XFDU Manifest, or (c) external to the XFDU Package. The flexibility to include (by reference) byteStream files that are external to the XFDU Package accommodates Data Objects that are too large to fit into a single XFDU Package, e.g., a multi-year archive.

The semantics specified in 8.4 of reference [1] put constraints on the permitted combinations of fileContent and fileLocation elements:

1. A single fileLocation element may be contained by a byteStream to reference the payload data, either in the containing XFDU Package or some external source.
2. A (single) fileContent element may be contained by a byteStream to contain the payload data (in base64 binary or XML format) within the body of the XFDU Manifest itself.
3. A single fileLocation element **and** a (single) fileContent element may be contained by a byteStream, where the fileContent data is to be a *backup* if the file referenced by fileLocation is not available – i.e., fileContent and fileLocation contain/reference the same (redundant) information, but in different locations. The apparent intention here is that the data referenced by the fileLocation element is external to the XFDU Package (since having the referenced file contained within the same XFDU Package would be needlessly redundant).
4. Two fileLocation elements may be contained in a byteStream, where one of the fileLocation elements references payload data *within the XFDU Package*, and the second fileLocation element references payload data that is *external to the XFDU Package*. In this combination, the referenced file within the XFDU Package is to be considered backup to the referenced file that is external to the XFDU Package, i.e. the two sets of referenced data are the same.
5. Other forms of multiple fileLocation elements, and the presence of one fileContent element and more than one fileLocation element, are explicitly undefined and their use is discouraged by the XFDU specification (reference [1]).

The CCSDS-standard XFDU Manifest also contains optional transformObject elements that “contain required information (e.g., algorithms and parameters) to reverse any transformations to the digital content and restore them to the original binary data object” (reference [8.1, [1]). The transformObject are used to (a) specify the relationship between a Data Object and the byteStreams into which it has been decomposed, and/or (b) to identify encryption and/or compression information associated with the Data Object.

**TGFT XFDU Data Object Section.** For purposes of TGFT usage, the content of the Data Object Section is significantly simplified. Figure 2‑10 shows the components of the DataObjectSectionType schema type that are used by TGFT..



**Figure 2‑10: DataObjectSectionType Components Used by TGFT, in XMLspy Graphical Representation**

**Data Object**. For TGFT, the only Data Objects that are recognized are those that can be conveyed in a single XFDU Package. Furthermore, for the purposes of TGFT, each TGFT XFDU Data Object Section contains a single Data Object.

NOTE - A TGFT-using service may still be able to use TGFT to transfer XFDU Packages that are segments of a larger XFDU, e.g., ones in which a Data Object consists of a concatenation of multiple byteStream files, some of which are contained within the XFDU Packages themselves and others that are external to those XFDU Packages. However, any such usage must be defined as part of the specification of such a TGFT-using service.

Each Data Object contains seven attributes:

1. **ID** (of type xsd:ID): This mandatory attribute contains the ID of the Data Object. It is this ID value that is contained in the **dataObjectID** attribute of the dataObjectPointer element of the contentUnit element of the informationPackageMap.
2. r**epID** (of type xsd:IDREFS): This optional attribute contains the “list of representation metadata IDREFs” applicable to the Data Object, and is present only if there are applicable Representation Information metadata files.

NOTE - The presence and contents of the repID attribute of the DataObject are the same as those for the informationPackageMap contentUnit that corresponds (and points) to the Data Object.

1. **mimeType** (of type xfdu:mimeTypeType): This optional attribute contains the “MIME type for the dataObject.” The presence and content of this attribute are controlled by the specification of the TGFT-using service.
2. **size** (of type xsd:long): This attribute contains the “size of the dataObject in bytes.” This attribute is optional in the CCSDS-standard XFDU Manifest but it is required for TGFT XFDUs.
3. **combinationName**: This optional attribute is not used in TGFT XFDUs because each Data Object consists of a single byteStream.
4. **registrationAuthority** (of type xsd:string): This optional attribute contains the name of “the authority that issued the registration.” This attribute is not used in TGFT XFDUs.
5. **registeredID** (of type xsd:string): This optional attribute contains the name of “the ID for the registration.” This attribute is not used in TGFT XFDUs.

For TGFT, the Data Object contains one byteStream element.

For TGFT, the Data Object may contain a checksum element.

For TGFT, the Data Object does not contain any transformObjects. transforObjects are unnecessary for TGFT because (a) there is a one-to-one relationship between Data Object and byteStream, and so there are no transformations related to the decomposition of Data Objects into multiple byteStreams; and (b) any information related to encryption and/or compression algorithms are assumed to be defined as part of the specification of the TGFT-using service.

**byteStream element**. The byteStream element has three attributes:

1. **ID** (of type xsd:ID): This optional attribute, if used, contains the ID of the byteStream. The byteStream **ID** attribute is not required for TGFT since there is a one-to-one relationship between the Data Object and the byteStream.
2. **mimeType** (of type xfdu:mimeTypeType): This optional attribute contains the MIME type for the byteStream. The byteStream **mimeType** attribute is not required for TGFT since there is a one-to-one relationship between the Data Object and the byteStream.
3. **size** (of type xsd:long): This attribute contains the “size of the dataObject in bytes.” The byteStream **size** attribute is not required for TGFT since there is a one-to-one relationship between the Data Object and the byteStream.

As noted above, [1] allows each byteStream to contain (a) (optionally) zero or more fileLocation elements (which point to payload data files that are not contained within the Manifest), (b) (optionally) one fileContent element, which contains either base64 binaryData or xsd:any xmlData, and (c) (optionally) a checksum over the contents of the byteStream.

The CCSDS-standard XFDU schema technically allows a byteStream element to be empty except for the content of the attributes. Whether or not this is intentional/and or represents a legitimate configuration for XFDUs in general, for purposes of TGFT either a single fileContent element or a single fileLocation element must be present in each byteStream.

The checksum element of the byteStream is not used for TGFT because it is redundant with the dataObject checksum.

The semantics specified in 8.4 of reference [1] that specify the permitted combinations of fileContent and fileLocation elements are further constrained for TGFT as follows:

1. The single fileLocation element can only be used as a reference to a payload data file within the same XFDU Package.
2. A (single) fileContent element may be contained by a byteStream to contain the payload data (in base64 binary or XML format) within the body of the XFDU Manifest itself. This is also valid for TGFT.
3. A single fileLocation element **and** a (single) fileContent element may be contained by a byteStream, where the fileContent data is to be a *backup* if the file referenced by fileLocation is not available – i.e., fileContent and fileLocation contain/reference the same (redundant) information, but in different locations. The apparent intention here is that the data referenced by the fileLocation element is external to the XFDU Package (since having the referenced file contained within the same XFDU Package would be needlessly redundant). In the context of TGFT, this is not a valid combination.
4. Two fileLocation elements may be contained in a byteStream, where one of the fileLocation elements references payload data *within the XFDU Package*, and the second fileLocation element references payload data that is *external to the XFDU Package*. In this combination, the referenced file within the XFDU Package is to be considered backup to the referenced file that is external to the XFDU Package, i.e. the two sets of referenced data are the same. In the context of TGFT, this is also not a valid combination because references to payload data external to the XFDU Package are not allowed.
5. Other forms of multiple fileLocation elements, and the presence of one fileContent element and more than one fileLocation element, are explicitly undefined and their use is discouraged by the XFDU specification (reference [1]), and for TGFT.

**fileLocation element**.

The fileLocation element has six attributes:

1. **ID** (of type xsd:ID): This optional attribute is the identifier of the fileLocation element. This attribute is not required for TGFT because TGFT constrains each byteStream to one fileLocation element, and so a separate identifier is redundant.
2. **textInfo** (of type xsd:string): This optional attribute accommodates a human-readable description of the file location. This attribute is not required for TGFT; any use of this attribute will be as specified for the TGFT-using service.
3. **locatorType** (of type xsd:string, with enumerated values “URL” and “OTHER”): For TGFT, this attribute is required and the value is always “URL”.
4. **otherLocatorType** (of type xsd:string): For TGFT, this optional attribute is not used because the value of locatorType in never “OTHER”.
5. **href** (of type xsd:string): For TGFT, this attribute contains the URL of the file. The URL always begins with “file:” because the payload file always exists within the same XFDU Package as the XFDU Mainfest.
6. **locator** (of type xsd:string): For TGFT, this optional attribute is not used because the value of locatorType in never “OTHER”.

The fileLocation element itself is empty: all information is carried in the attributes.

**fileContent element**.

The fileContent element has one attribute (**ID**) which is not required for TGFT because it is redundant with the ID of the containing dataObject.

The fileContent element contains a choice between a binaryData element and an xmlData element. The binaryData element is used to directly contain a Base64-encoded binary data file. The xmlData element is used to directly contain an XML file (document) from any namespace.

**Summary of TGFT XFDU simplified Data Object construction rules**. The net effect of the constraints on the use of fileContent and fileLocation elements that are specified in [1], combined with the TGFT requirement that the payload data be contained within the same XFDU Package, simplifies the allowed use of dataObject, fileContent and fileLocation elements as follows:

* Each XFDU Package shall contain a single dataObjectSection.
* Each dataObjectSection shall contain a single dataObject element.
* Each dataObject element shall contain one and only one byteStream element.
* Each byteStream element shall contain either a single fileContent element or a single fileLocation element.
* If the byteStream element contains a fileLocation element, that fileLocation element must point to a file within the same XFDU Package.
* If the bytestream element contains a fileContent element, that file Content element must contain either a Base64-encoded binary block of data or an XML document.

### Metadata Section

Metadata is contained in, or referenced by, the (optional) metadataSection component of the XFDU Manifest.

Figure 2‑11 is the XMLspy graphical representation of the CCSDS-standard MetadataSectionType schema type. Figure 2‑12 shows the components of that schema type that are used by TGFT.



**Figure 2‑11: CCSDS-standard MetadataSectionType in XMLspy Graphical Representation**



**Figure 2‑12: MetadataSectionType Components Used by TGFT, in XMLspy Graphical Representation**

The XFDU specification [1] allows the metadataSection to contain zero or more metadataObject elements. However, for use in TGFT, if the optional metadataSection component is present, it is required to contain at least one metadataObject element. I.e., if there are no metadataObjects, their absence in a TGFT XFDU Package is always represented by the absence of the metadataSection component.

According to [1], each metadataObject has the following attributes:

1. **ID** (of type xsd:ID): This required attribute contains the ID that is pointed to by the **repID**, **dmdID**, **pdiID**, or **anyMdID** attributes of the contentUnits associated with payload files that use those metadataObjects (see 2.4.3.2.1).
2. **classification** (of type xsd:string with enumerated values): see 2.4.5.5 for more details.
3. **category** (of type xsd:string with enumerated values): see 2.4.5.5 for more details.
4. **otherClass** (of type xsd:string): see 2.4.5.5 for more details.
5. **otherCategory** (of type xsd:string): see 2.4.5.5 for more details.

Each metadataObject contains (a) an optional metadataReference element, (b) an optional metadataWrap element, and (c) an optional dataObjectPointer element.

**metadataReference element.** The metadataReference element is used to point to a metadata file that is outside of the XFDU Manifest itself. This could be either in the XFDU Package or external to the XFDU Package (e.g., on a server somewhere). Both options are supported by TGFT.

The metadataReference element has eight attributes:

1. **ID** (of type xsd:ID): This optional attribute is not used for TGFT, because there is a one-to-one relationship between the metadataReference and metadataObject. Therefore, the required metadataObject **ID** attribute is also the ID for the metadataReference.
2. **textInfo** (of type xsd:string): This optional attribute is not required for TGFT, but it may be useful for some TGFT-using services. If so, the semantics and allowed values must be defined as part of the service specification.
3. **locatorType** (of type xsd:string, with enumerated values “URL” and “OTHER”): required
4. **otherLocatorType** (of type xsd:string): optional – used when locatorType = “OTHER”
5. **href** (of type xsd:string): optional – used when locatorType = “URL”.
6. **locator** (of type xsd:string): optional – used when locatorType = “OTHER”.
7. **vocabularyName** (of type xsd:string): According to [1], the optional attribute contains. “the type of metadata contained (e.g., MARC, EAD, etc.)”. This attribute is not required for TGFT, but it may be useful for some TGFT-using services. If so, the semantics and allowed values must be defined as part of the service specification.
8. **mimeType** (of type xsd:string): This optional attribute identifies the MIME type of the medata file. This attribute is not required for TGFT, but it may be useful for some TGFT-using services. If so, the semantics and allowed values must be defined as part of the service specification.

The metadataReference element itself is empty; all information is contained in the attributes.

**metadataWrap element.** The metadataWrap element is used to include metadata within the XFDU Manifest itself, in either base64 binary or XML format. This is supported by TGFT.

The metadataWrap element has four attributes:

1. **ID** (of type xsd:ID): This optional attribute is not used for TGFT, because there is a one-to-one relationship between the metadataWrap and metadataObject. Therefore, the required metadataObject **ID** attribute is also the ID for the metadataWrap.
2. **textInfo** (of type xsd:string): This optional attribute is not required for TGFT, but it may be useful for some TGFT-using services. If so, the semantics and allowed values must be defined as part of the service specification.
3. **vocabularyName** (of type xsd:string): According to [1], the optional attribute contains. “the type of metadata contained (e.g., MARC, EAD, etc.)”. This attribute is not required for TGFT, but it may be useful for some TGFT-using services. If so, the semantics and allowed values must be defined as part of the service specification.
4. **mimeType** (of type xsd:string): This optional attribute identifies the MIME type of the medata file. This attribute is not required for TGFT, but it may be useful for some TGFT-using services. If so, the semantics and allowed values must be defined as part of the service specification.

The metadataWrap element contains a choice between a binaryData element and an xmlData element. The binaryData element is used to directly contain a Base64-encoded binary metadata file. The xmlData element is used to directly contain an XML file (document) from any namespace.

**dataObjectPointer element**. The optional dataObjectPointer element points “to a Data Object in the Data Object section. This allows a Metadata Object to also be described as Data Object in the Data Objects section. Since the dataObject includes an attribute that is an internal pointer to Representation Information, a Metadata Object can be associated with its own Representation Information. Note that this mechanism allows the construction of OAIS-defined ‘Representation Nets’ when the associated Representation Metadata Objects are also held as Data Objects.” [1], section 9.1).

The dataPointerObject type appears to exist to serve uses of XFDUs that are more esoteric than those present in simple file transfer, Therefore, the dataObjectPointer element is not used in TGFT XFDUs.

NOTE - If a particular TGFT-using service requires the use of dataObjectPointers as either mandatory or optional choices for the content of metadataObjects, the specification of that service must specify the inclusion of the dataObectPointer choice and define the semantics for its use in the context of that service.

#### Refined Construction Rules for TGFT metadataObject Elements

By the construction of the CCSDS-standard XFDU schema, it is possible for a single metadataObject to have a metadataReference element, a metadataWrap element, and a dataObjectPointer element concurrently. Unlike for the case of the fileLocation and fileContent elements in the Data Object byteStream element (see 2.4.4.2.2), reference [1] does not define any constraints or relationships among the metadataReference, metadataWrap, and dataObjectPointer elements. For TGFT XFDUs, the metadataReference and metadataWrap elements shall be used as follows:

1. Each XFDU shall contain a zero or one metadataSection.
2. If present, the metadataSection may contain one or more metadataObject elements.
3. Each metadataObject element may contain a metadataReference element, a metadataWrap element, or both.
4. Each metadataObject element shall contain at least a metadataReference element or a metadataWrap element.
5. If the metadataObject element contains a metadataReference element but not a metadataWrap element, the metadataReference shall point either to a file contained within the same XFDU Package as the XFDU Manifest or a file that is external to the XFDU Package. In either case, the referenced file contains the normative metadata.
6. If the metadataObject element contains a metadataWrap element but not a metadataReference element, the metadataWrap shall contain the normative metadata content of the metadataObject.
7. If the metadataObject element contains both a metadataReference element and a metadataWrap element, the metadataReference shall point to a file that is external to the XFDU Package, and the content of the metadataWrap element is to be a backup copy of the metadata. The referenced file contains the normative metadata, and the embedded backup copy is to be used only if the normative file is unavailable.

NOTES

1 One example use for having both a metadataWrap element and metadataReference elements used is the case of an XML schema, where the normal schema location is a URL associated with the namespace of that schema. Having the backup copy of the schema file embedded in the XFDU Manifest allows the data content (an XML document) to be validated even if the normative source is unavailable.

2 The combination of a metadataWrap element and a metadataReference that points to a file within the same XFDU Package is redundant and therefore disallowed.

3 As described in 2.4.5.3, the dataObjectPointer element is not used in TGFT XFDUs.

#### Metadata Categorization and Classification

The metadataObject has four optional attributes that are closely related to each other: **category**, **classification**, **otherClass**, and **otherCategory**.

The **category** attribute can have one of five enumerated values: “REP”, “PDI”, “DMD”, “OTHER”, or “ANY”.

The **classification** attribute can have one of eight enumerated values: “DED”, “SYNTAX”, “FIXITY”, “PROVENANCE”, “CONTEXT”, “REFERENCE”, “REFERENCE”, “DESCRIPTION”, and “OTHER”.

The **otherClass** attribute is defined in [1] as the “type of metadata in case classification contains value of "OTHER."

The **otherCategory** attribute is defined in [1] as the “type of metadata class in case category contains value of "OTHER."

Three of the categories – REP, PDI, and DMD – refer to metadata categories defined by the OAIS Reference Model (reference [3]). As described in [1]:

“The XFDU Manifest Schema provides predefined metadata categories and classes via enumerated attributes that follow the OAIS information model as follows:

* Descriptive Information, intended for the use of Finding Aids such as Catalogs or Search Engines, may be categorized as ‘DMD’ and further classified as ‘DESCRIPTION’ or ‘OTHER’.
* Representation Information may be categorized as ‘REP’ and then further classified as ‘SYNTAX’, ‘DED’ (data entity dictionary), or ‘OTHER’.
* Preservation Description Information may be categorized as ‘PDI’ and then further classified as ‘REFERENCE’, ‘CONTEXT’, ‘PROVENANCE’, ‘FIXITY’, or ‘OTHER’.”

A metadataObject of category “DMD” is pointed to by a contentUnit through the **dmdID** attribute (see 2.4.3.2.1).

A metadataObject of category “REP” is pointed to by a contentUnit through the **repID** attribute (see 2.4.3.2.1).

A metadataObject of category “PDI” is pointed to by a contentUnit through the **pdiID** attribute (see 2.4.3.2.1).

A metadataObject of category “ANY” or “OTHER” is pointed to by a contentUnit through the **anyMdID** attribute (see 2.4.3.2.1).

A metadataObject that does not have a **category** attribute is pointed to by a contentUnit through the **anyMdID** attribute (see 2.4.3.2.1).

For TGFT, there are no requirements for the **category**, **classificatio**n, **otherClas**s, and **otherCategory** attributes. Use of these attributes should be considered in the context of each service that uses TGFT and specified for that service.

Unless use of one or more of the OAIS pre-defined categories (REP, PDI, DMD) is appropriate to a service, the **category** attribute of a metadataObject will contain “ANY” or “OTHER”.

If use of one or more of the OAIS pre-defined catagories is appropriate to a service, that service should conform to the semantics of those categories as specified in [3].

### Behavior Section

The Behavior Section is not part of the standard TGFT XFDU Manifest.

## XFDU PAckage File formats

Each invocation of TGFT transfers one XFDU Package.

NOTE - A payload data file contained within an XFDU Package may itself be an XFDU Package. For example, the proposed Forward Last Hop service could use the XFDU Package as its packaging mechanism for containing the payload data to be delivered to the target along with the instructions for such delivery). The Forward Last Hop XFDU Package would be the payload data file within the Forward File XFDU Package that is transferred via TGFT from the MOC to the Earth Space Link Terminal (ESLT).

Each XFDU Package is a file that is constructed using either TAR or ZIP. The specification of the TGFT-using service specifies whether TAR or ZIP is to be used for that service, or defines the mechanism by which that selection is made (e.g., by service agreement).

The XFDU Manifest within the (zipped/tarred) XFDU Package is an XML formatted document with the extension “.xfdu” that uniquely identifies it as the XFDU Manifest and differentiates it from any other XML-formatted file that might be in XFDU Package.

NOTE - Use of the ”.xfdu” extension instead of “.xml” will require XFDU Package parsers to recognize the .xfdu-extensioned files as XML file and perhaps to convert the extension before submitting the file to a standard XML parser.

EDITOR’S NOTE - Section 3.2.2 of [2] specifies that “The physical XFDU entity shall be constructed using either TAR or ZIP, the service agreement shall specify whether TAR and/or ZIP is supported.” (1) The use of the term “physical XFDU entity” is confusing. My interpretation (see above) is that what TGFT transfers is an XFDU Package. (2) The mechanism for determining whether ZIP or TAR is used should be deferred to the TGFT-using service.

XFDU Package name syntax – see draft TGFT specification (reference [2]).

EDITOR’S NOTE - Section 3.2.1 of [2] specifies that “To minimize the risk of overwriting files of the same name that are still present in the target directory, the file source shall post-fix the name of the destination file with the Zulu time at which the upload of the file was started.” The implication is that this applies to not only the “wrapper file” (i.e., the zipped/tarred folder), but also to the files that are contained in that folder – the XFDU Manifest and any payload and metadata files within that XFDU Package. But since the XFDU Manifest must point to the payload/metadata files, to do so requires that those file names be established by the time that the XFDU Manifest is created. In effect, it requires that all of the files be named, the XFDU Manifest be created, and they all be zipped/tarred instantaneously at the moment that they begin transmission. Is this reasonable? I can certainly understand requiring the “outer wrapper” (the zipped/tarred XFDU Package folder) to follow this rule. And what about the Return File example, where the payload file was created aboard the spacecraft and therefore the name would contain the timestamp of the time of file creation onboard. Also, concerning metadata – a poorly-defined cross support service could retransmit the same metadata file, but besides being redundant and bandwidth-wasteful it doesn’t really do harm to overwrite the same file. Perhaps we should simply require the timestamp as part of the enclosed file names but leave the semantics to the individual services that use TGFT.

## Applicability of TGFT XFDU Composition rules to services that use TGFT

The composition rules for TGFT XFDUs define the subset of CCSDS-standard XFDU Package and XFDU Manifest elements and schema types that are standard for services that use the TGFT.

Nominally, TGFT-using services are expected to support these composition rules in generating and processing the XFDU Packages that are transferred by those services. For example, the TGFT XFDU composition rules allow the payload data to be carried as a Data Object of the XFDU Manifest or as a file within an XFDU Package that is pointed to by a fileLocation element of and XFDU Manifest in the same XFDU Package. A service that directly adopts the TGFT XFDU composition rules allows the File Source to use either method, and any File Recipient must be able to receive and process both options. But the use of the Behavior section is undefined for TGFT XFDUs in general, so nominally a TGFT-using service would not be expected to generate/process XFDUs with Behavior sections in the XFDU Manifest.

While the subset of XFU capabilities supported by the TGFT XFDU are intended to be able to support the cross support service use cases that have been identified so far (e.g., Forward File service, Delta-DOR File service, Validated Radiometric Data File service), there may arise future cross support services that could use TGFT as the underlying transfer mechanism but that have information content requirements that are not fully met by the standard TGFT XFDU. For example, performance requirements or deployment characteristics of a particular service may dictate that only one of the payload-data-packaging options (either embedded within in the XFDU Manifest **or** carried as a separate file in the XFDU Package and pointed to by the Manifest) be supported by that service. The specification for such a service would constrain the XFDUs to the selected option and exclude to other.

On the other hand, a TGFT-using service may generate/process data that is fundamentally categorized as data of a type that is supported by the CCSDS-standard XFDU but is undefined for standard TGFT XFDUs (e.g., Behavior data). The addition of such content to the XFDUs for such a service must also be defined in the specification for that service.

# Composition Rules for XFDU Packages for TGFT

## General

This section contains the specification of the composition rules for XFDU Packages for TGFT. These rules will migrate to the TGFT Specification Recommended Standard.

For clarity, the TGFT XFDU composition rules are expressed in terms of XML entities (elements, attributes, etc.).

NOTE - The TGFT XFDU composition rules ignore components of the CCSDS-standard XFDU that are not part of the TGFT XFDU (e.g., the Behavior Section). Developers of a TGFT-using service that may wish to extend the definition of the TGFT XFDU for the purposes of that service should do so in conformance with the provisions, capabilities, and composition rules specified in the CCSDS-standard XFDU Blue Book (reference [1]).

## XFDU Manifest

**XFDU Manifest File Names.** The rules for forming names of XFDU Manifest files are to be defined by the service using TGFT, within the following constraints:

The characters used to form the names shall be constrained as follows:

* [a-z], lower case alphabetic characters (with exception as specified in 3.2.1.2)
* [0-9], numeric characters
* -, the “dash”(or minus) character
* \_, the underscore character
* ., the dot character

If the file name contains a timestamp (e.g., a time of creation), the timestamp portion of the name should conform to the modified CCSDS ASCII Time Code B format:

<YYYY>-<DDD>T<hh>-<mm>-<ss>Z

where <<YYYY> is the 4-digit year subfield with values 0001-9999,  
<DDD> is the 3 digit day of year subfield with values 001-365 or -366,  
T is Calendar-Time separator,  
<hh> is the 2-digit hour subfield with values 00-23,  
<mm> is the 2-digit minute subfield with values 00-59,  
<ss> is the 2-digit second subfield with values 00-59 (-58 or -60 during leap seconds), and   
Z is the time code terminator.

NOTES

1 The CCSDS-standard ASCII Time Code B uses colons (“:”) as delimiters between the hour, minute, and second subfields of the timecode string. The modified CCSDS ASCII Time Code B substitutes dashes (“-“) for colons (“:”), which are not permitted in filenames in some file systems.

2 To be conformant with the CCSDS ASCII Time Code B the “T” and the “Z” must be in upper case. The use of “T” and “Z in this context are the only exceptions to the all-lower-case constraints on file names.

The XFDU Manifest shall conform to the XFDU XML Schema specified in section 11 of [1], with further restrictions and refinements as specified in sections/paragraphs 3.2.3 through 3.2.6.4.2 below.

NOTE - Optional elements/attributes of the CCSDS-standard XFDUType that have no anticipated for TGFT-using services are ignored.

### XFDU Manifest Attributes

1. The optional textInfo attribute of the XFDU element may be used to carry TGFT-using service specific information.

NOTE - As of this writing, there is only one version of the XFDU XML Schema, that which is defined in Issue 1 (September 2008) of [1]. If and when additional version of the XML schema become available in the future, and assuming that by then Schema Naming and Versioning rules will be in place to unambiguously define the corresponding contents of the version attribute, the version attribute will also be required in TGFT XFDUs to specify which version is being used.

The XFDU Manifest shall contain:

a. one instance of the informationPackageMap element.

b. zero or one instance of the metadataSection element.

c. one instance of the dataObjectSection element.

### informationPackageMap element

#### informationPackageMap attributes

1. The optional **packageType** attribute of the informationPackageMap element is required for TGFT. It carries the registered name of the TGFT-using service specific information (e.g., “ForwardFile”).

#### The informationPackageMap element shall contain one contentUnit element.

#### contentUnit element

##### contentUnit attributes

1. The optional **unitType** attribute of the contentUnit element may be used to carry TGFT-using service specific information.
2. The optional **textInfo** attribute of the contentUnit element may be used to carry TGFT-using service specific information.
3. The optional **repID** attribute of the contentUnit element shall be present if there are any associated metadata files that are categorized as Representation Information (that is, that have a **category** attribute value of “REP” – see 2.4.5.5). If present, this attribute shall contain the IDREF values of the **ID** attributes of the corresponding metadataObject elements (see 3.2.6.2.1 (a)).
4. The optional dmdID attribute of the contentUnit element shall be present if there are any associated metadata files that are categorized as Descriptive Information (that is, that have a category attribute value of “DMD” – see 2.4.5.5). If present, this attribute shall contain the IDREF values of the **ID** attributes of the corresponding metadataObject elements (see 3.2.6.2.1 (a)).
5. The optional **pdiID** attribute of the contentUnit element shall be present if there are any associated metadata files that are categorized as Preservation Information (that is, that have a category attribute value of “PDI” – see 2.4.5.5). If present, this attribute shall contain the IDREF values of the **ID** attributes of the corresponding metadataObject elements (see 3.2.6.2.1 (a)).
6. The optional **anyMdID** attribute of the contentUnit element shall be present if there are any associated metadata files that cannot be categorized as Representation, Descriptive, or Preservation Information. If present, this attribute shall contain the IDREF values of the **ID** attributes of the corresponding metadataObject elements (see 3.2.6.2.1 (a)).

The contentUnit element shall contain one dataObjectPointer element.

#### dataPointerObject element

##### dataPointerObject Attributes

1. The **dataObjectID** attribute shall contain the ID of the dataObject element (3.2.7.2.1 (a)).

### metadataSection element

The metadataSection element shall contain one metadataObject element for each metadata file associated with the payload data being transferred in the XFDU Package.

#### metadataObject element

##### metadataObject Attributes

1. The **ID** attribute shall contain the unique ID of the metadataObject.
2. The **category** attribute may be used to specify the category of the metadata contained in the metadataObject, as suits the needs of the service using TGFT:
   * 1. If the metadata conforms to the OAIS information model [3] categories of Descriptive, Representation, or Preservation Description Information, the value of the **category** attribute may be set to ‘DMD’, ‘REP’, or ‘PDI’, respectively.
     2. The value of the **category** attribute may be set to ‘OTHER’ or ‘ANY’. The category ‘OTHER’ should be used if an alternative categorization is provided, and ‘ANY’ should be used if no further categorization is to be provided.
     3. The **category** attribute may be absent.
3. The **classification** attribute may be used to specify the classification of the metadataObject, as suits the needs of the service using TGFT:
4. If the value of the **category** attribute is ‘DMD’, the value of the **classification** attribute shall be set to either ‘DESCRIPTION’ or ‘OTHER’.
5. If the value of the **category** attribute is ‘REP’, the value of the **classification** attribute shall be set to ‘SYNTAX’, ‘DED’, or ‘OTHER’.
6. If the value of the **category** attribute is ‘PDI’, the value of the **classification** attribute shall be set to ‘REFERENCE’, ‘CONTEXT’, ‘PROVENANCE’, ‘FIXITY’, or ‘OTHER’.
7. If the value of the **category** attribute is ‘OTHER’, the value of the **classification** attribute shall be set to ‘OTHER’, or the **classification** attribute maybe absent altogether.
8. If the value of the **category** attribute is ‘ANY’, the **classification** attribute shall be absent.
9. If the **classification** attribute value = ‘OTHER’, the **otherClass** attribute shall contain the classification of the metadata. The definition of other classifications of metadata is deferred to the specification of the TGFT-using service that uses such metadata.
10. If the **category** attribute value = ‘OTHER’, the **otherCategory** attribute shall contain the category of the metadata. The definition of other categories of metadata is deferred to the specification of the TGFT-using service that uses such metadata.

If the metadata content represented by the metadataObject element is normatively specified outside of the XFDU Package:

* + - 1. the metadataObject shall contain a metadataReference element that identifies the normative source.
      2. The metatadataObject may contain a metadataWrap element. The content of the metadataWrap element is nominally the same as that of the normative external metadata file: it’s inclusion in the XFDU Package is for backup purposes in case the normative external source of the metadata is unavailable.

If the metadata content represented by the metadataObject element is to be normatively specified as part of the XFDU Package, that metadataObject element shall contain either:

1. a metadataWrap element the contains the normative metadata; or
2. a metadataReference element that points to the normative metadata file within the same XFDU Package.

#### metadataReference Element

##### metadataReference Attributes

1. The optional **textInfo** attribute of the metadataReference element may be used to carry TGFT-using service specific information.
2. The **locatorType** attribute shall be set to ‘URL’ if the locator is a URL, or to ‘OTHER’.
3. If the **locatorType** attribute value = ‘OTHER’, the **otherLocatorType** attribute shall be present when and shall identify the type of the locator.
4. The **href** attribute shall be present when the **locatorType** attribute value = ‘URL’, and shall contain the URL of the file.
5. If the metadata file is contained within the same XFDU Package as the XFDU Manifest, the URL of the file shall be of the form

“file:”+<XFDU Package file name service-specific part>

+ ”/” + <metadata file name (with extension>.

1. If the **locatorType** attribute value = ‘OTHER’, the **locator** attribute shall be present when and shall contain the locator itself.
2. The optional **vocabularyName** attribute may be used if appropriate to the metadata type(s) used by a TGFT-using service. Identification/definition of appropriate metadata vocabulary names is deferred to the specification of the TGFT-using service.
3. The optional **mimeType** attribute may be used if appropriate to the metadata type(s) used by a TGFT-using service. Identification/definition of appropriate metadata MIME types is deferred to the specification of the TGFT-using service.

#### metadataWrap Element

##### metadataWrap Attributes

1. The optional **textInfo** attribute of the metadataWrap element may be used to carry TGFT-using service specific information.
2. The optional **vocabularyName** attribute may be used if appropriate to the metadata type(s) used by a TGFT-using service. Identification/definition of appropriate metadata vocabulary names is deferred to the specification of the TGFT-using service.
3. The optional **mimeType** attribute may be used if appropriate to the metadata type(s) used by a TGFT-using service. Identification/definition of appropriate metadata MIME types is deferred to the specification of the TGFT-using service.

The metadataWrap element shall contain either a Base64-encoded binaryData element or a valid XML-formatted document.

### dataObjectSection Element

The dataObjectSection element shall contain one dataObject element.

#### dataObject Element

##### dataObject Attributes

1. The required **ID** attribute shall contain the ID of the Data Object.
2. The optional **repID** attribute shall be present only if there one of more representation metadata files applicable to the Data Object. The **repID** attribute shall contain the list of IDREFs for all metadataObjects in the metadataSection that have a **category** attribute value of ‘REP” (see 3.2.6.2.1 (b)).
3. The optional **mimeType** attribute may be present if appropriate to the Data Object that is transferred by the individual TGFT-using service, as specified for that service.
4. The **size** attribute shall be present and contain the size of the Data Object.

The dataObject element shall contain one byteStream element.

The dataObject element may contain a checksum element.

#### byteStream Element.

The byteStream element shall contain either (1) a fileLocation element or (2) a fileContent element.

#### fileLocation Element

##### fileLocation Attributes

1. The optional **textInfo** attribute of the fileLocation element may be used to carry TGFT-using service specific information.
2. The **locatorType** attribute shall be set to ‘URL’.
3. The **href** attribute shall contain the URL of the file, which shall be of the form

“file:”+<XFDU Package file name service-specific part>

+ ”/” + <payload file name (with extension>.

#### fileContent Element

The fileContent element shall contain either (1) binaryData element or (2) an xmlData element.

#### binaryData Element

The binaryData element of the fileContent element shall contain the Base64 encoding of the payload data.

#### xmlData content

The xmlData element of the fileContent element shall cantain a valid XML document.

## Payload Data File enclosed within the XFDU Package

The content of the payload data file shall be defined by the service using TGFT.

The rules for forming names of XFDU Package-enclosed payload data files are to be defined by the service using TGFT, within the constraints specified in 3.2.1.1 and 3.2.1.2.

## Metadata File enclosed within the XFDU Package

The content of the metadata data file(s) shall be defined by the service using TGFT.

The rules for forming names of XFDU Package-enclosed metadata files are to be defined by the service using TGFT, within the constraints specified in 3.2.1.1 and 3.2.1.2.

NOTE - The structure of the names of external metadata files is controlled by authority the controls the repositories that contain those metadata files.

## TGFT XFDU Package

Each TGFT XFDU Package shall contain one XFDU Manifest XML document, as defined in 3.2.

The TGFT XFDU Package shall contain an enclosed payload data file name if and only if the XFDU Manifest contained with the XFDU Package contains a fileLocation element (XFDU:dataObjectSection:dataObject:byteStream:fileLocation), where the name of the enclosed payload data file is contained in the **href** attribute of that fileLocation element.

The TGFT XFDU Package shall contain an enclosed metadata file for every metadataReference element in the XFDU Manifest (XFDU:metadataSection:metadataObject: metadataReference), for which the href attribute contains a URL that points to the XFDU Package itself (i.e., the URL is of the form

“file:”+<XFDU Package file name service-specific part>

+ ”/” + <metadata file name (with extension>).

The name of the XFDU Package file shall consist of two substrings: the service-specific part and the timestamp part.

The rules for forming the service-specific part of the XFDU Package file names are to be defined by the service using TGFT, within the following constraints specified in 3.2.1.1.

The timestamp part of the XFDU Package file shall conform to the modified CCSDS ASCII Time Code B, as defined in 3.2.1.2.

The XFDU Package file shall be packaged as a .zip or .tar file.

1. Example XFDU Manifest

In this example, a hypothetical Validated Radiometric Data cross support service generates XML-formatted Tracking Data Messages (TDMs) in conformance with the XML Specification for Navigation Data Messages Blue Book (CCSDS 505.0-B-1, December 2010). This hypothetical service collects all tracking data for a space link session, performs validation processing, and transfers the resulting TDM XML document as a file over TGFT. (Note that this hypothetical service is different from the Tracking Data Cross Support Transfer Service, which transfer samples of tracking data in near-real time.)

The Validated Radiometric Data service generates an XFDU Package ZIP file containing an XFDU Manifest file (XML document) and a separate XML document containing the TDM itself.

The metadata for this XML document consists of the XML schema file for the TDM, which is located in the SANA Registry at

<http://www.sanaregistry.org/r/ndmxml/ndmxml-1.0-tdm-1.0.xsd>.

The XFDU Package ZIP file is named

dss\_25\_validated\_tdm\_xfdu\_package-2017-058T23-15-46Z.zip,

where the timestamp substring of the file name indicates the time at which the XFDU Package begins being transferred.

The XML TDM file is named

dss\_25\_validated\_tdm-2017-058T19-35-24Z.xml,

where timestamp substring of the file name indicates the time at which the tracking data stopped being acquired.

Thus the **href** attribute of the dataObject fileLocation element is

file:dss\_25\_validated\_tdm\_xfdu\_package/dss\_25\_validated\_tdm-2017-058T19-35-24Z.xml,

which indicates that the file is located within the dss\_25\_validated\_tdm\_xfdu\_package zipped “folder”.

Table A-1 is the XFDU Manifest XML document for this example XFDU Package.

NOTE - The file naming convention used in this example has been contrived for the purposes of providing example file names.

Table A‑ : Example XFDU Manifest

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <xfdu:XFDU textInfo="Example of TGFT XFDU Manifest" version="1.0" xsi:schemaLocation="urn:ccsds:schema:xfdu:1 XFDUschema.xsd" xmlns:n1="http://www.altova.com/samplexml/other-namespace" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xfdu="urn:ccsds:schema:xfdu:1">  <packageHeader ID="PkgHdr">  <volumeInfo>  <specificationVersion>1.0</specificationVersion>  </volumeInfo>  <environmentInfo>  <xmlData>  <Provider\_ID> DSS-25 </Provider\_ID>  <Mission\_ID> XenoSat </Mission\_ID>  <Cross\_Support\_Service\_Type> Validated Radiometric Data</Cross\_Support\_Service\_Type>  </xmlData>  </environmentInfo>  </packageHeader>  <informationPackageMap textInfo="Validated Radiometric Data Info Pkg Map" packageType="ValidatedRadiometricData">  <xfdu:contentUnit textInfo="content unit for TDM XML document" anyMdID="TDM\_Schema" unitType="TDM">  <dataObjectPointer dataObjectID="TDM\_Payload"/>  </xfdu:contentUnit>  </informationPackageMap>  <metadataSection>  <metadataObject category="REP" ID="TDM\_Schema" otherClass="SCHEMA" classification="OTHER">  <metadataReference textInfo="Reference to repository of TDM schema" locatorType="URL" href="http://www.sanaregistry.org/r/ndmxml/ndmxml-1.0-tdm-1.0.xsd" mimeType="application/xml"/>  </metadataObject>  </metadataSection>  <dataObjectSection>  <dataObject ID="TDM\_Payload" size="214748" mimeType="application/xml">  <byteStream>  <fileLocation locatorType="URL" href="file:dss\_25\_validated\_tdm\_xfdu\_package/dss\_25\_validated\_tdm-2017-058T19-35-24Z.xml"/>  </byteStream>  <checksum checksumName="payload checksum">[payload checksum value]</checksum>  </dataObject>  </dataObjectSection>  </xfdu:XFDU> |

1. Reference Bookmarks (to be deleted)

[1] nRef\_661x0\_XFDU

[2] nRef\_927x1\_TGFT

[3] nRef\_650x0\_OAIS\_RM