Technical Specification of the proposed unified X-ray file format for Non-intrusive Inspection devices – UFF 2.0
SUMMARY DOCUMENT

The Technical Experts Group on Non-Intrusive Inspection (TEG-NII) of the World Customs Organization (WCO) was established in 2016. One of the main tasks of the TEG-NII is to promote synergies between the Customs and the Industry to develop a standard NII data format with the purpose to facilitate interoperability of different NII equipment provided by different manufacturers/vendors; to facilitate exchange of images in a unified file format within and between Customs administrations for Customs purposes; to facilitate the development of databases or libraries of images; and to assist in training of image analysts.

The development of a standard NII data format, codenamed the Unified File Format (UFF), is being implemented in three phases. Following the successful proof of concept during Phase 1, Phase 2 started in October 2016. The objectives of Phase 2 are to launch and execute the architecture of the UFF; test and evaluate the Phase 2 architecture and launch the preliminary discussions on the approach for Phase 3.

The UFF development work is carried out primarily by a UFF Development Team comprising experts from the NII vendors involved in the UFF Development - L3, Nuctech, Rapiscan Systems AS&E and Smiths Detection, supported by the Customs Co-Chairperson of the TEG-NII and the WCO Secretariat.

The UFF Development Team works towards continuously evolving the UFF standard to support and improve the data content of the UFF standard and provide a common output from scanning hardware, detection activities and potentially provide maintenance information.

As part of the UFF Phase 2 implementation, the UFF Development Team developed a Technical Specification of the proposed unified X-ray file format for Non-intrusive Inspection devices – UFF 2.0 (appended hereto). The proposed UFF 2.0 image architecture was tested, from 11 September – 19 October 2018, through a pre-pilot with Dutch Customs. In the first quarter of 2019, it will be further tested through a Pilot with volunteering WCO Member Customs administrations.

The purpose of this Summary Document is to summarize the complete architecture document contents into a concise, non-technical summary.

The UFF 2.0 Technical Specification states the technical details required for successful delivery for a unified X-ray file format, as jointly prepared for the various types of high energy cargo inspection systems used throughout most Customs administrations inspection operations. This architecture document (from inception), details the technical approach for the required architecture for the UFF file structure delivery. The document contains information regarding the UFF 2.0 related dataset flow, file structure, UFF versioning, file artifacts, data artifacts (XML Data), as well as explanations regarding data exchange and storage, digital signatures, archiving, UFF 1.0 compatibility and UFF 2 dataset merging. Further details on the content of the various sections are outlined below.

1.1 Dataset Flow

The dataset flow is the technical process flow required to extract out the individual core data elements from the respective NII systems' native data file formats, applying the proper data flow routing from native to UFF file structure.

1.2 File Structure

The file structure is described as the core architecture, which the data takes as a result of the actions executed during the dataset flow process.

1.3 Versioning

As the UFF 2.0 file architecture is a “living” software architecture, which means that a proper governance and file version structure is required to maintain the planned routine cadence of evolutionary development of the UFF 2.0 file architecture.

1.4 File Artifacts

The section titled File Artifacts articulates the construct elements, which make up the core building blocks required to build out the UFF file output. These artifacts, when coded together, make up the fundamental structure of the base UFF product. The core elements included as part of this file structure framework include:

- **Naming Convention** - describes the fundamental naming approach for the different sub-events that take place when the UFF 2.0 translation process is in execution.
- **Manifest** - the manifest copy (if existing with native file) associated with that particular inspection event data, which remains in PDF form and travels with its appropriate named file during the translation process.
- **Optical Images** - images associated with any pictures taken with optical capture devices (e.g. cameras) for the purpose of including any identifier artifact for the scanned object, such as container identifier, plate numbers or wagon identifiers.
- **X-Ray File** - the X-ray file image as a result of the X-ray event. The X-Ray file can include one or more X-ray images, depending on what is included as part of the native file output.
- **Material Discrimination (HEMD)** - the image file artifact associated with one or more images data sets as a result of any material discrimination results, should the native system be equipped to provide those as part of the inspection event.
- **Radiation Detection** - the UFF 2.0 file architecture has the foundational capability to include captured data as part of a radiation detection function, should the inspection system have this capability. However, this feature is not included in the first release. It will be added in later releases.
- **Metadata** - the UFF 2.0 architecture also accommodates a starting package of metadata inputs via one XML file. Metadata fields are divided into 2 event categories:
    1. Target definition event: definition of the scanned object.
        a. This element is used to describe the target type details (cargo, who transports it, how it is transported, owner, consignor, consignee and declarations, etc.).
    2. Inspection event(s): definition of types of inspection events (as listed in the following):
        a. Prognosis event: assessment of the risk associated to the target.
        b. Scan event: X-ray scan (could also be a radiation detection, to be considered for future versions).
        c. Analysis event: automatic analysis of the case, image analysis performed by an operator or physical inspection.
        d. Comment event: additional information provided by an operator.
    3. **Integrity Validation** - allows for an optional “DigiSign” attribute, which can contain the digital signature of a section (i.e. the event element and its contents), allowing the file architecture to provide a level of file

integration validation as a result of the translation process for the UFF 2.0 file format.

4. **Maintenance Event** - this capability has been acknowledged as a future need and as such shall be part of the future development during Phase 3 of the UFF Development Programme.
5. **Limitations** - as part of the due diligence effort during the development process of UFF 2.0, some limitations have been identified that are expected to be addressed in the effort to expand the UFF 2.0 capability through future development stages.

1.5 Miscellaneous Elements

Elements to be considered in future versions have been identified, reviewed and listed for future developments as part of the growth of the UFF 2.0 file architecture:

- **Data Exchange and Storage** - this is a primary capability which the NII Industry aspires to provide, as a formal means of exchanging the UFF standard format between systems and interfaces, which has been determined as a desirable attribute by the WCO, as well as the TEG-NII members.
- **Digital Signatures** - the need has been identified to validate the integrity of the data items within the structure. Digital signatures will serve to both provide tamper validation and data integrity check when parts of the structure will be transmitted over (sometimes remote and unreliable) network.
- **Archiving** - due to the nature of the file size potential, the Root directory that stores individual UFF 2.0 files will eventually grow to an unmanageable size. It is the responsibility of the NII Supplier, along with the customer, to provide an archiving and deletion strategy.
- **UFF 1.0 Compatibility** - backward compatibility of UFF 2.0 to UFF 1.0 is not supported since the previous format supports only a single scan event per dataset. However, UFF 2.0 scan events can be exported as standalone UFF 1.0 datasets, if necessary.
- **UFF 2.0 Dataset Merging** - a dataset merging process has been created within the architecture to mitigate the potential scenario as per which, if a cargo is inspected in different disconnected locations, several UFF 2.0 datasets will be generated for the same cargo.

*

* *

Unified File Format (UFF) 2.0 Standard

A technical description of the unified X-ray file format for Non-Intrusive Inspection (NII) devices

Version: **1.0.3**
Document dated: **23 November 2018**

Prepared by UFF Development Team for the World Customs Organization's Technical Experts Group on Non-Intrusive Inspection

Document Management

About	
About	**This document describes the UFF 2.0 Standard data exchange format**
Author	Alexis Labonne (original inception)

Revision History

Version	Reason for change	Updated by	Date
0.1	First Draft	Alexis Labonne	02/04/2017
0.2	Reviewed by Sebastien for XML schema changes and file examples	Sebastien Rogerat	12/04/2017
0.3	Document management added, responses to Sebastien	Alexis Labonne	16/04/2017
0.4	Release for approval by UFF development team	Tim Norton	29/04/2017
0.5	Modifications to support Backscatter	Christopher Hogg	05/06/2017
0.6	Modifications based on TEG-NII members feedback (meeting in Brussels, 3-5 May, 2017)	Ivan Gudak	29/08/2017
0.6.1	Digital Signature in UFF2	Ivan Gudak	31/08/2017
0.7	Overall review	Sebastien Rogerat	10/08/2017
0.8	Integration of CIS industry comments	Ivan Gudak Sebastien Rogerat	14/09/2017
0.8.2	Integration of Nuctech comments	Denys Spektor	18/09/2017
0.9	Integration of CIS industry comments	Ivan Gudak Denys Spektor Sebastien Rogerat	24/10/2017
0.9.1	Updated schema to support target updates + small improvements	Christopher Hogg Sebastien Rogerat Ivan Gudak	01/11/2017
0.9.2	Make Goods as list in Declaration. Move manifest to Event directory	Christopher Hogg Sebastien Rogerat Ivan Gudak	9/11/2017
0.9.3	Fixed manifest file identification. Descriptive file referencing. UFF2 Merge procedure	Ivan Gudak	29/11/2017
0.9.5	Global review for the changes. HEMD chapter updated	Sebastien Rogerat	30/11/2017
0.9.6	Changed HEMD format to PNG; minor fixes to the schema	Ivan Gudak	27/02/2018
0.9.8	Incorporated HEMD and Dataset Preview decisions made in Uganda during the 3rd Meeting of the TEG-NII	Ivan Gudak Sebastien Rogerat	10/04/2018
0.9.9	Updated Data artifact section as agreed during the NII vendors' meeting in Brussels	Sebastien Rogerat	26/06/2018
1.0.0	Glossary updates; update HEMD format; various corrections in document and XML schema (double <Target> removed)	Ivan Gudak Sébastien Rogerat	11/09/2018
1.0.1	Updated formatting and corrected spellings. Added definition of *UFF 2.0 Basic* subset. Changed EventId data type from integer to UUID, to simplify procedures for digital signing and dataset merging. Updated UFF XSD for further technical simplification.	Guy Levell Sébastien Rogerat	31/10/2018
1.0.2	Changed wording of HEMD specification to make it clearer	S Jessup C Hogg	01/11/2018
1.0.3	Improve x-ray and HEMD format descriptions	Sébastien Rogerat	23/11/2018

Contents

1.1 Dataset Flow ... 1
1.2 File Structure ... 2
1.3 Versioning ... 2
1.4 File Artifacts ... 2
1.5 Miscellaneous Elements ... 3
Document Management ... 2
Revision History ... 2
Contents ... 3
1 Terms ... 4
2 Dataset flow ... 6
3 File structure ... 8
4 Versioning ... 10
5 File Artifacts ... 11
5.1 Naming convention ... 11
5.2 Manifest ... 11
5.3 Optical images ... 11
5.4 X-ray ... 12
5.5 High energy material discrimination (HEMD) ... 12
5.6 Radiation detection ... 16
5.7 Metadata ... 16
6 Data Artifact (XML Data) ... 17
6.1 Overview ... 17
6.2 File references ... 17
6.3 Target ... 18
6.4 Inspection events ... 20
6.5 Integrity validation (optional) ... 23
6.6 Maintenance event ... 24
6.7 Limitation ... 24
7 Data Exchange and Storage ... 26
8 Digital Signatures ... 27
9 Archiving ... 28
10 UFF 1.0 Compatibility ... 29
11 UFF 2 Dataset Merging ... 30

1 Terms

Term	Explanation for the purposes of the current document
API	Application Program Interface – the interface for program-to-program communication.
Cargo Declaration	Information submitted prior to or on arrival or departure of a means of transport for commercial use that provides the particulars required by the Customs relating to cargo brought to or removed from the Customs territory.
CCR	Container Code Recognition – OCR that reads and recognizes container numbers.
CIS	Cargo Inspection System
Dataset	A collection of sets of tracking and inspection information (data and files) related to an inspection case, placed into a ZIP container according to this UFF specification.
FTP	File Transfer Protocol. Internet protocol designed for file transferring.
HFS	HTTP File Server, or Hyper Text Transfer Protocol File Server - free web server specifically designed for publishing and sharing files.
HSCode	The Harmonized Commodity Description and Coding System, also known as the Harmonized System (HS) of tariff nomenclature is an internationally standardized system of names and numbers to classify traded products.
HTTP	Hyper Text Transfer Protocol – Internet protocol designed to transfer webpages and REST requests/responses.
Inspection Case	A cargo unit (truck, container etc) which may pass inspection events at the borders UFF2 is created to manage one Inspection Case.
Inspection Event	One-time event the cargo unit is being inspected. Can be scan, manual inspection. UFF2 may include multiple inspection events.
IoT	Internet Of Things – machine to machine communication over the Internet.
LPR	License Plate Recognition – OCR that reads and recognizes vehicle plate numbers.
Manifest	Digital Document containing the Cargo Declaration. Presented as PDF file format.
Metadata	XML database that describes the cargo, shipping and inspection events.
NFTS	New Technology File System – file system developed by Microsoft Corporation, used in Windows NT and successors as the primary file system.
NII	Non-Intrusive Inspection
OCR	Optical Character Recognition
Scanning	Capturing information (which may include images or radiation signatures) relating to goods and means of transport by utilizing nonintrusive detection equipment.
Screening	The evaluation of information and intelligence relating to goods and means of transport in a risk assessment process (manual, automated or otherwise).
SMB (Samba)	Server Message Block, network protocol used to share resources (including files) over a network involving mostly computers running Microsoft Windows.

TEG	Technical Experts Group
TEG-NII	Technical Experts Group on Non-Intrusive Inspection
UFF	Unified File Format
UFF Development Team	Organizations and individuals who contributed to the UFF format. NII Vendors, Dutch Customs (Joris Groeneveld) and WCO Secretariat.
URI	Unified Resource Identifier.
UUID	Universally Unique Identifier v4 that is generated using a secure random number generator.
WCO	World Customs Organization
WNR	Wagon Number Recognition – OCR that reads and recognizes wagon numbers.

2 Dataset flow

The UFF2 dataset supports multiple inspection events of different types such as X-Ray scanning, manual cargo inspection, image analysis etc. Refer to Section 6.4 (Inspection events) for more details about inspection event types.

Inspection events can be added at different points on the cargo's way from departure to the final destination.

A UFF2 dataset shall be created at the point when the cargo item is formed (container loaded and preliminary assessment is done). Initially the data file (in .xml format) shall contain the information about the target. The data file will be updated with new inspection information during the further inspection events.

An initial manifest file shall be created at the creation step, and will be named "manifest1.pdf".

On its way the cargo can be inspected several times, and each inspection event shall be added to the same UFF2 dataset as a new inspection event:

- New directories contacting inspection files (X-rays, optical images, documents etc.)
- New sections in the metadata file
- Additional manifest files (optional)
- Previous inspection events should not be changed (neither files in the directories nor sections in the XML data file)
- Target Sections shall not be changed. If any field in an existing Target section needs an update, then a new target section should be created by copying the existing target section and correcting the necessary fields. The new section will contain the latest Target data. All previous target sections remain in the metadata for history purposes.

The files in the inspection events' directories shall not be duplicated in further events even if they are used there. For instance, if an X-ray file was created during Scan Event, it shall not be copied to the directory of the Analysis Event, but can be referenced in both sections (see Section 6.4.3, Analysis event for more details).

Refer to Figure 1 below for more details.

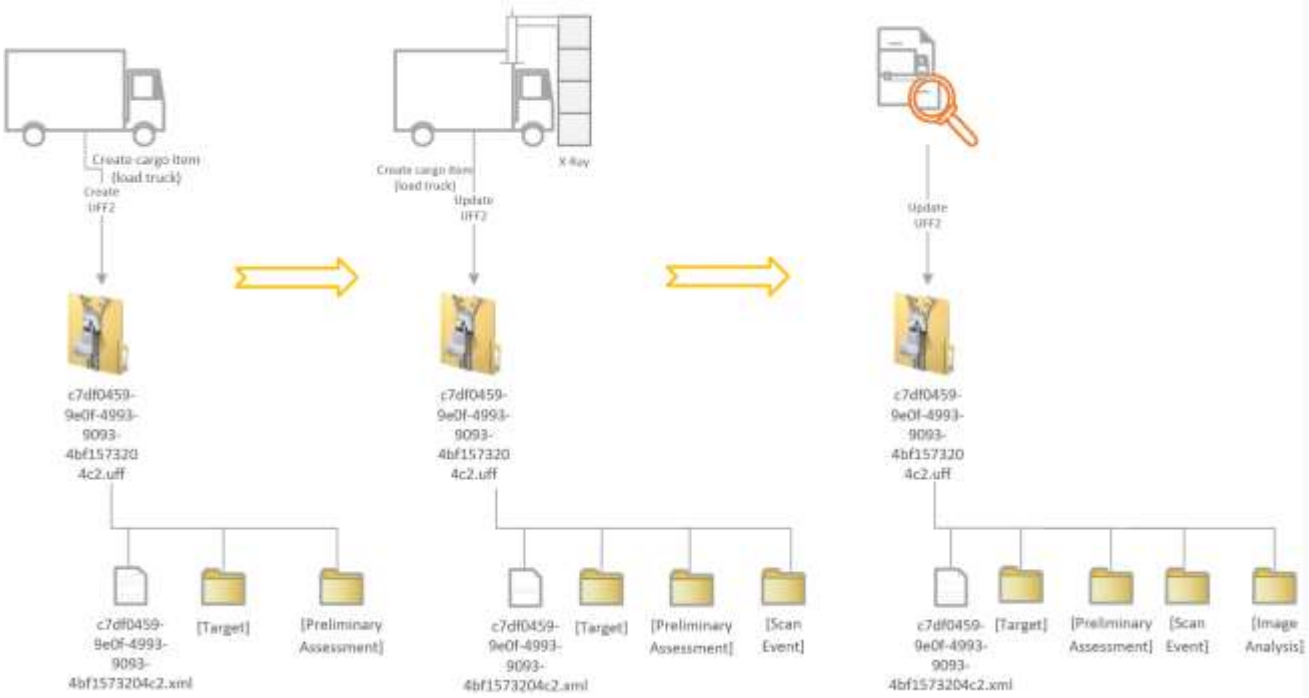


Figure 1. UFF2.0 life cycle

At any point when the cargo is checked, inspectors may have the access to all previous inspection information, so they can compare their own findings with the foregoing check results.

Disconnected locations, without the access to the datasets from other locations, will create new UFF2 datasets containing only the inspection events made locally. Later several UFF2 datasets created for the same cargo item can be merged into a single one containing all the inspection events performed for the cargo on its way.

Some real world cases may be only partially handled by UFF 2.0 such as one container having multiple declarations or one declaration being linked to several containers. Later versions of the UFF will be able to manage such cases.

3 File structure

The file structure provides a differentiation between inspection events and the respective artifacts. Whereas currently scanning data is stored and consumed from a filesystem, a different storage mode is proposed in Figure 2 below of this document. Currently, most scanners and their integrated software provide a “file share” access to the data (SMB or FTP).

The UFF2 files and all the data inside are not intended to be manipulated by operators directly, but via special software (UFF2 viewers), so the directory structure and the naming convention are optimized for software and machines.

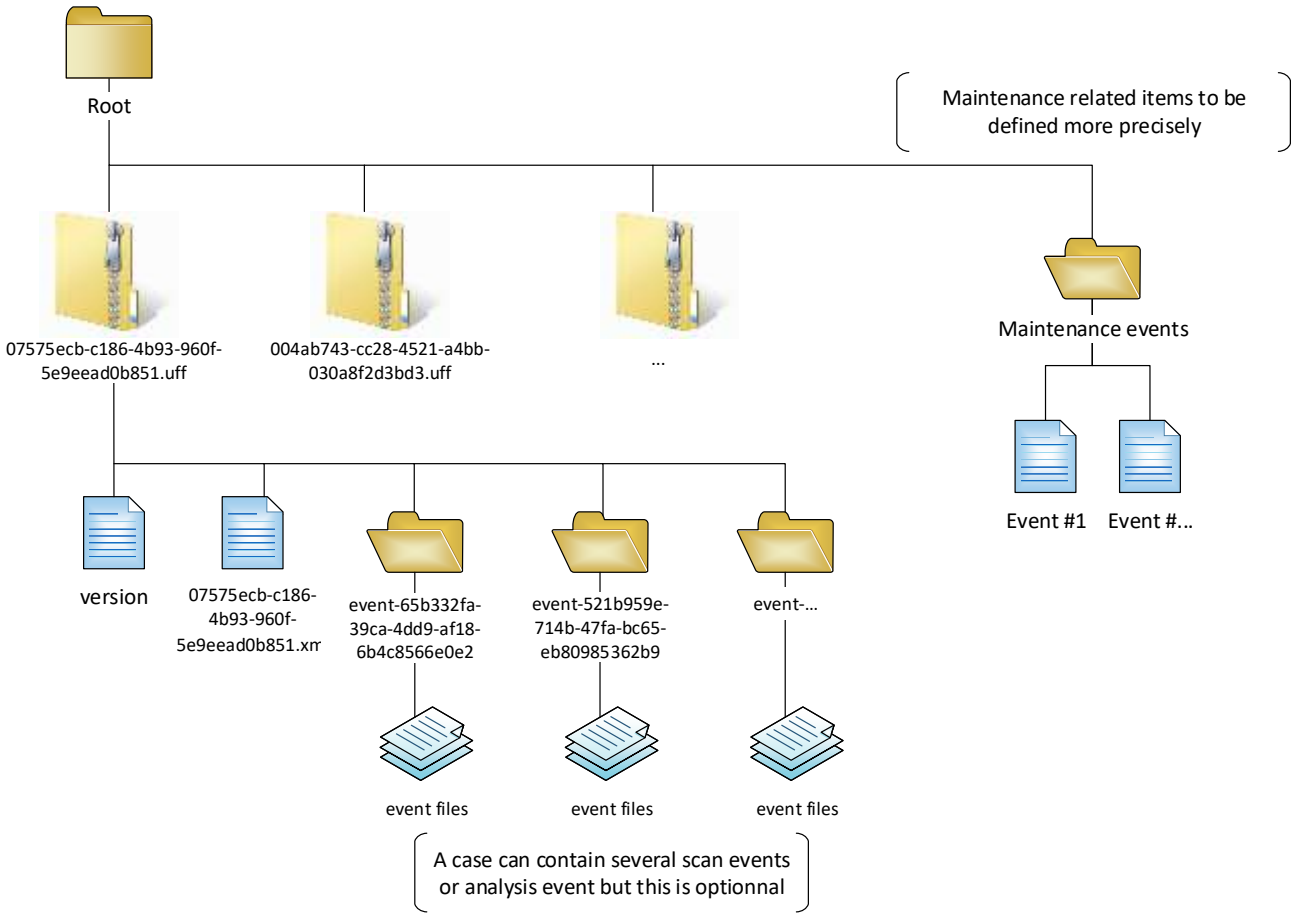


Figure 2. UFF2 structure

Root

A root endpoint or directory exists for each piece of hardware (e.g. Scanner). It can be accessed through FTP or mounted filesystem (e.g. FTP, SMB, HFS etc.…)

Note: Previous iterations of UFF 1.0 contained heartbeat information. Part 3 of the UFF 2.0 Standard is planning to provide room for maintenance information (undefined at this time). It is likely that future scanning apparatus will adopt IoT standards (either Microsoft or Amazon). UFF will evolve along with the technology evolution for future IoT standardization.

The root directory contains UFF2 files that are a collection of inspection cases (target definition and associated events). For each inspection case there will be a separate "`.uff`" container which would encapsulate all related files. In a manner similar to a .JAR Java Archive, the UFF file format is built on ZIP standard version 6.3.4. The zipped dataset will have extension `.uff`. The same extension shall be used also for future major versions of UFF (UFF3, UFF4, etc).

A single UFF2 dataset contains multiple inspection events related to the same cargo item (container, truck etc.). Each inspection event is a directory containing all files related to the inspection (such as optical images made by visual inspection, snapshots of marks made during X-Ray analysis etc.).

A UFF2 dataset shall be created when a container is checked for the first time. All further inspection events shall be added into the same UFF2 dataset until the cargo's destination.

In addition to the UFF2 files, the root directory shall also include collection of Maintenance Events (note that this portion of UFF Part 3 is not defined at this stage).

4 Versioning

UFF will evolve to include more features in the future. To ensure correctness of file opening and to avoid corruption, each UFF dataset, starting from version 2.0, shall include a version file in its root. This file shall be always named "`version`", without an extension, and must include a single line of text (encoded using UTF-8) as follows:

```
<major version>.<minor version>
```

Where:

`<major version>` is the major version of the UFF, usually included into the format name. For the UFF, described in this document, the major version is 2, and the dataset format is UFF2. Datasets with different major versions may be fully incompatible.

`<minor version>` is the minor version of the UFF. For the format described in this document it is 0 (zero). Datasets with the same major version but different minor versions should be backward compatible, i.e. if a software is certified to support UFF 2.2, it should also work with UFF 2.0 and UFF 2.1.

The use of a plain text format allows software to read the version information and to fail gracefully, if required, without any need to open the XML file (which may be incompatible).

Datasets without version information shall be considered as UFF1 format.

Examples:

```
2.0
2.1
3.2
```

5 File Artifacts

5.1 Naming convention

The following naming conventions shall apply to the UFF 2.0 file format:

- UFF2 datasets shall be named as `<uuid>.uff` to ensure worldwide uniqueness. For instance, `0c211f28-6e66-4edf-a1e6-3d3473216e9f.uff`.
- In addition to the UUID of the overall UFF2 dataset, each inspection event within the UFF2 dataset (including initial identification of the target) shall also be allocated a UUID.
- One directory per event shall be created within the UFF2 dataset. The naming convention shall be `event-{uuid}` (for example, `event-996d0022-c8be-4f2e-921d-f1bf0b982f7f`).
    - The '`event-`' prefix is used to avoid any confusion between the UUID used for the overall UFF2 dataset, and the UUIDs used for events within the UFF2 dataset.
- Any files associated with an event (including manifest files for target definition events) shall be located in the event's directory. For internationalization purposes the filenames shall meet the following general requirements:
    - Filenames shall be unique within each directory.
    - Filenames shall not be case sensitive. ("`Name`", "`NAME`" and "`name`" will be considered as the same filename).
    - File extensions shall correspond to the content (`.jpg` for JPEG files, `.tif` for TIFF files and so on).
    - Only English letters, numbers, dash (-) and underscore (_) characters are permitted for internationalization purposes.
- A Data file (XML) shall be placed in the UFF2's root and shall have the same name as the overall UFF2 dataset, with an XML extension (e.g. `0c211f28-6e66-4edf-a1e6-3d3473216e9f.xml`).
- A Version file shall be placed in the root of UFF2. The filename should be "`version`" without extension.
- A Dataset Preview File may be optionally placed in the root of UFF2. The preview should be stored in JPEG format and is to provide an indication to the user as to what cargo/object information is stored in the dataset. In most cases, the preview file will be a JPEG version of the first scan event's X-ray image so the user can quickly view it without opening the dataset in the analyzer. The filename for the preview file should be "`preview.jpg`".

Each file associated to a case is anyway referenced in the XML metadata file. Using this method, more information can be stored, including a checksum of the file. Additionally, for the XML metafile being signed (see below in this document), any alteration of a file after its generation will be known.

5.2 Manifest

Zero or more files containing manifest information associated with the cargo.

Location: directory of the Target Change Event.

File format: `PDF`

File name: `manifest<index>.pdf`

- `<index>` incremental index of the file, to avoid duplicated names. The first manifest for the event shall have index 1 in the filename (the name will be `manifest1.pdf`).

Examples:

```
manifest1.pdf
manifest5.pdf
```

5.3 Optical images

One or more optical image data files of the scanned object. The intended uses are:

- To provide an image of the identifier of the scanned object: container identifier, plate numbers or wagon identifier.

- To provide an image of the content of the scanned object (e.g. photo of the seizure).

Location: directory of the associated event.

File format: `PNG` or alternatively `JPG, BMP, TIF`.

File name: `image<index>.<ext>`

- `<index>` incremental digital index to ensure uniqueness of the filename in its directory.
- `<ext>` file extension that should correspond to the content (`.jpg` for JPEG files, `.tif` for TIFF files and so on).

Examples:

```
image1.png
image185.png
```

5.4 X-ray

One or more x-ray image data files including multiple technics (transmission or backscatter), energies and views. The x-ray image must have been corrected in speed. X-ray values must be linear from 0 to 65535 or in other words, the image mustn't be corrected in log.

Location: directory of the associated event.

File format:

- TIFF file (single image file)
- Grayscale 16 bits depth (0 meaning no signal)
- Lossless compressed or uncompressed

File name: `xray<index>.tif`

- `<index>` incremental digital index to ensure uniqueness of the filename in its directory.

Examples:

```
xray1.tif
xray985.tif
```

5.5 High energy material discrimination (HEMD)

One or more image data files representing the result of the HEMD. There should be one or zero HEMD file per x-ray image.

Location: directory of the associated event.

File format:

- PNG file
- Palletized – 8 bits per pixel referring to a color in the palette.

  One single file is used to store HEMD result. There is one or zero HEMD file for each x-ray intensity image. No X-ray intensity data is stored in this file. HEMD files shall be the same dimension and orientation as the X-ray intensity data.

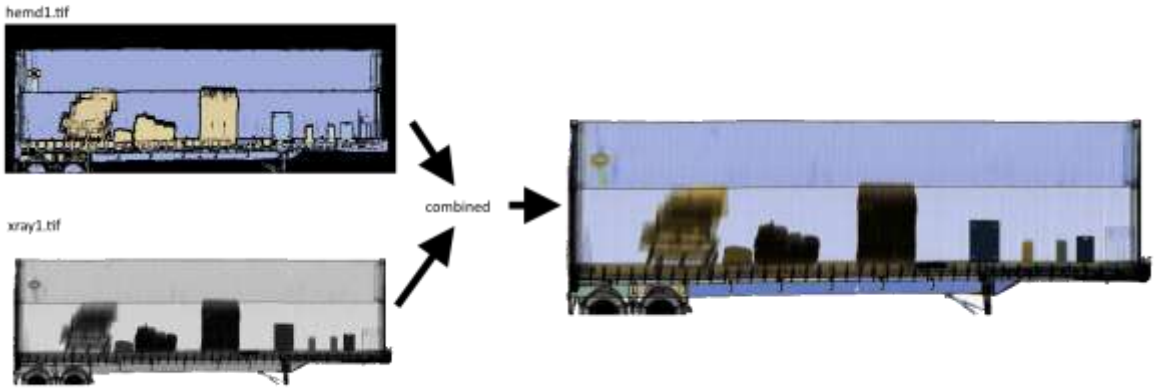
  The values in the HEMD file represent the Z value of the material. For material with a Z inferior to 30, the palette index can be considered as the Z value but this is not true for material with higher Z.
  - 0 is used for a pixel not having HEMD information. Associated (background) color in the palette should be black or white. This is used when Z value cannot be determined for example with very dense material.
  - 1 to 10 are used for organic Z values – for example plastic has the Z value of 5.
  - 11 to 20 are used for intermediate Z values – for example aluminium has the Z value of 13.
  - 21 to 30 are used for mineral Z values – for example steel has the Z value of 26.

- 31 to 40 may be used for high Z materials on systems that support them - e.g. Lead has the Z-value of 82 but associated color index is between 31 and 40.

A viewer can use this fixed Z range information to display only a sub part of the Z values (material stripping). This can be easily achieved my manipulating the HEMD palette to replace ranges that are not to be colored with grey values (128,128,128) values.

The colors in the palette are stored as RGB (8 bits per component). The alpha channel can be present but not used.

The HEMD image is used as a transparent overlay of the X-ray image, the recommended transform is to convert the data into HSL space and combine with the X-ray intensity data to produce a colorized image.



The recommended method of combining color with intensity is to convert to HSL (Hue Saturation Lightness) color-space and then replace the lightness value with the intensity value from the X-ray image.

Per Pixel (assuming that h, s and l are within the range [0.0, 1.0]):

```
h,s,l = rgbtohsl( original_z_colour )     // Extract hue and saturation from HEMD image
l = intensity_from_xray / 65535.0         // Replace lightness value by the X-ray intensity value linearly
                                          // brought back to [0.0, 1.0]
output_colour = hsltorgb( h, s, l )       // Combine HEMD H, S with Intensity L from x-ray Intensity
```

A viewer should use the color in the HEMD palette to display the HEMD data to produce an accurate rendition of the original vendors image. Optionally the vendor can substitute their own palette to produce a uniform experience across multiple vendors (see figures 3 and 4 below for an example of palette substitution).

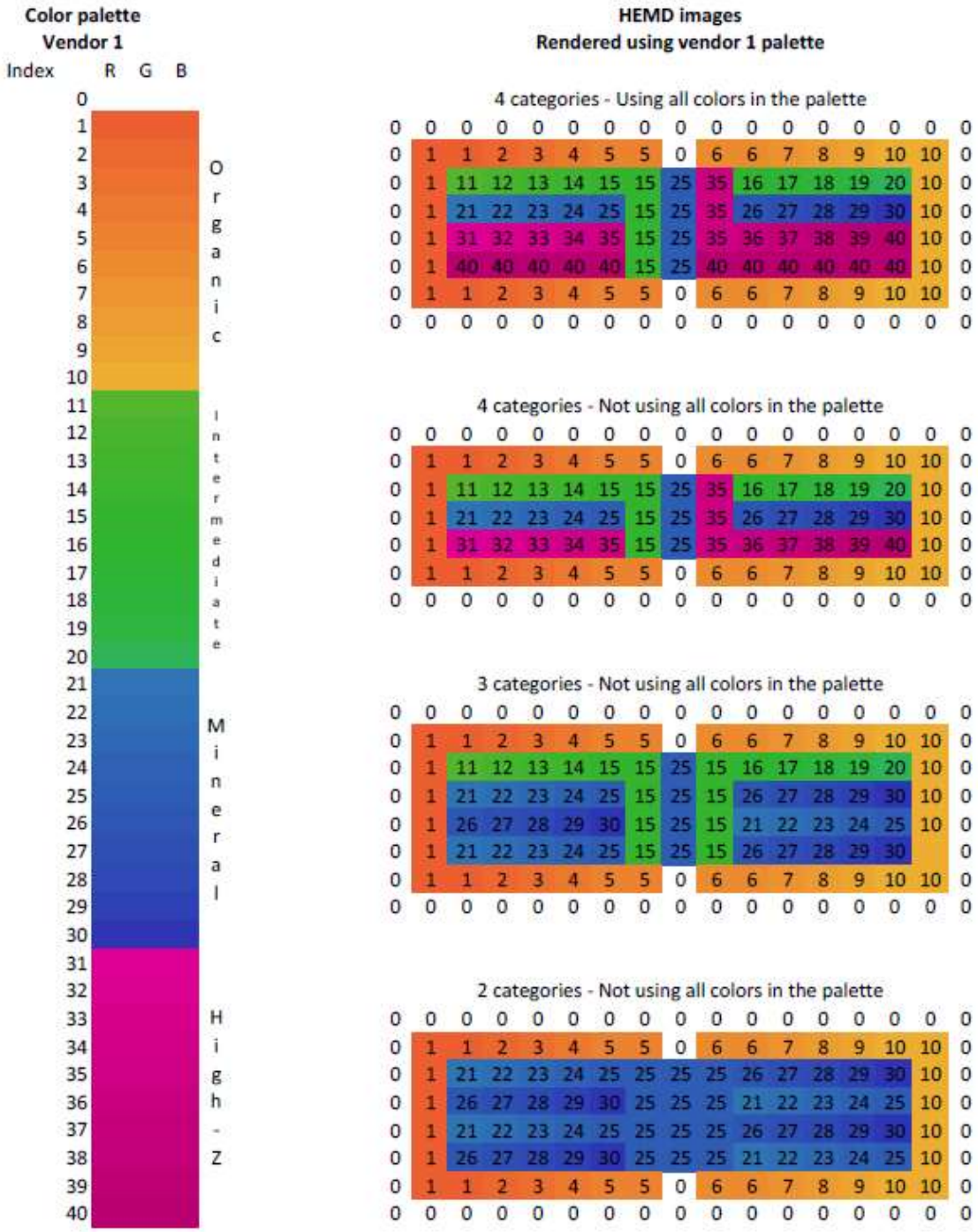


Figure 3 - Example of HEMD Image rendered with vendor 1 palette

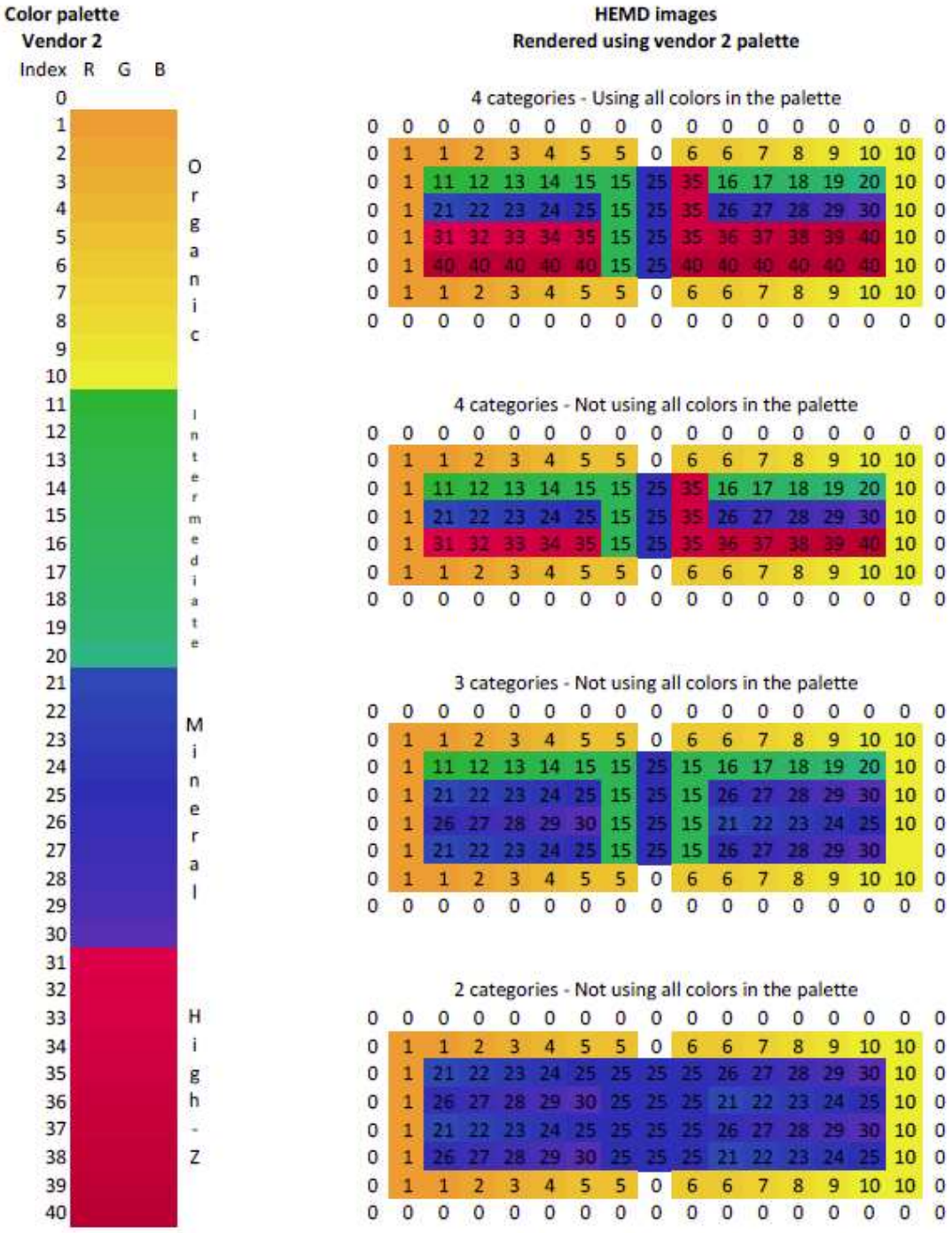


Figure 4 - Example of HEMD Image rendered with vendor 2 palette

- Lossless compressed or uncompressed

File name: `hemd<index>.png`

- `<index>` is incremental digital index to ensure filename uniqueness.

Examples:

```
hemd1.png
hemd17.png
```

5.6 Radiation detection

Radiation detection is not supported by UFF 2.0. Support will be added in a future version.

5.7 Metadata

One XML file containing the metadata associated to an inspection case (see next section for a complete description).

Location: The root directory of the UFF2 dataset.

File format: `XML` compliant with UFF 2.0 schema definition.

File name: `<inspection case name>.xml`

- `<inspection case name>` is the same as the name of the UFF2 file.

Example:

```
0c211f28-6e66-4edf-a1e6-3d3473216e9f.xml
```

6 Data Artifact (XML Data)

6.1 Overview

The attached XSD file describes the structure of the XML file containing either the metadata associated to an UFF 2.0 inspection case or the data associated to maintenance event.

UFF2.0.xsd

Metadata associated to an inspection case are split into events. There are several types of event:

- Target definition event: definition of the scanned object.
- Inspection events:
  - Prognosis event: assessment of the risk associated to the target.
  - Scan event: X-ray scan (could also be a radiation detection, to be considered for future versions).
  - Analysis event: automatic analysis of the case, image analysis performed by an operator or physical inspection.
  - Comment event: additional information provided by an operator.

The number of events in an XML file is not limited. For example, the inspection process can start with the definition of the target, a risk assessment is done, an X-ray scan is performed, the result is analyzed, the target definition is updated, another scan is done and so on.

Each event has a mandatory field "`EventId`" (supporting a UUID – Universally Unique Identifier). The EventId is used to identify the subdirectory containing additional files (such as scanned documents, X-ray images, optical images, reports etc) related to the event. See Section 3 (File structure) for more details about the file naming convention.

The file below contains the documentation extract from the XSD file:

UFF2.0 XSD documentation.html

6.2 File references

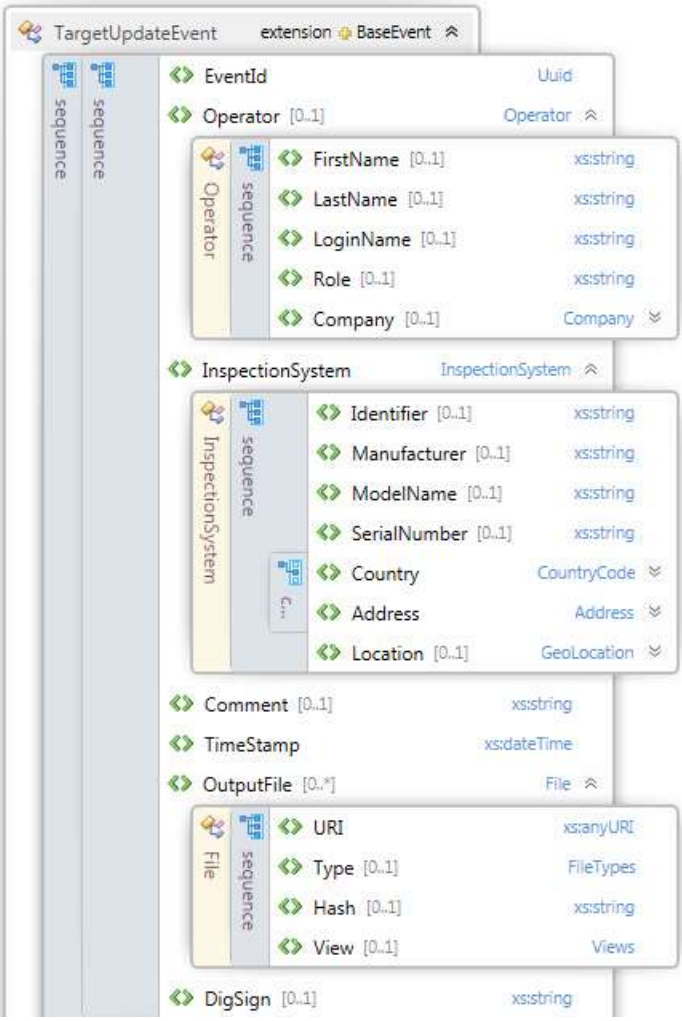
Sections in metadata may reference other files in the UFF2, such as X-ray files, HEMD files or any other output files.

If an event section refers to its own file, which is placed in the event's directory, then the file reference shall include only filename and extension (e.g. `hemd14.png`)

If a section refers to a file that belongs to a different event (for instance, if an analysis event refers to an X-ray file from a scan event), then the reference should include full path relative to the UFF2 root (e.g. `event-996d0022-c8be-4f2e-921d-f1bf0b982f7f/xray12.tif`).

6.3 Target

The Target element describes the cargo, who transports it, how it is transported, owner, consignor, consignee and declaration(s).



The cargo can be a `Vehicle`, `Wagon` or `Container`.

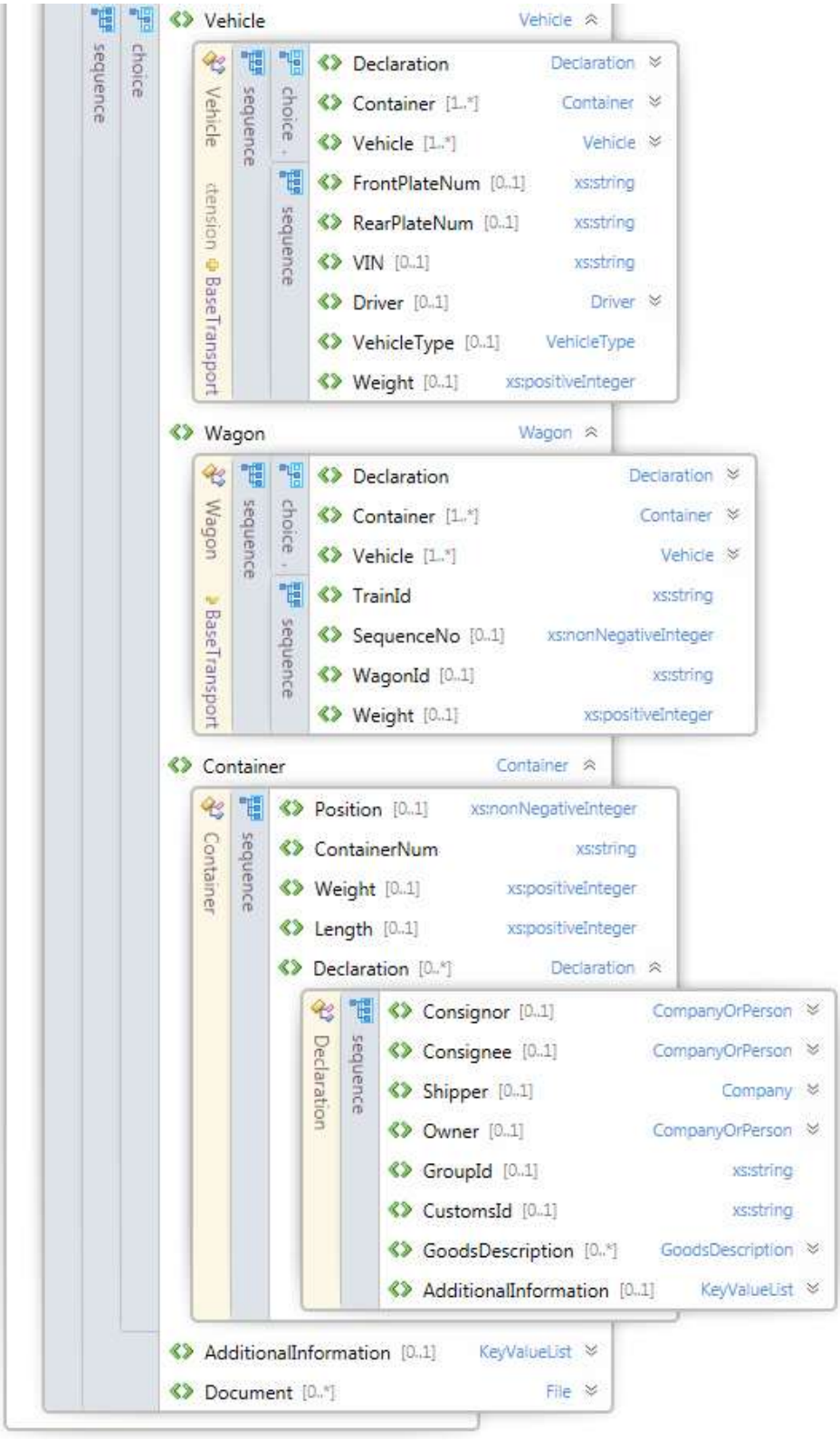
A Vehicle can be a `Truck`, `Car`, `Bus`, `Van` or `Motorcycle`.

`Wagons` and `Trucks` may have multiple `Containers`.

Each `Container` may have zero or more declarations.

A single declaration may also cover multiple containers. In such case the same declaration section shall repeat in all the corresponding containers.

If a `Target` element needs to be updated, a copy of it shall be created and the changes shall be applied to the copy. The latest `Target` element shall be considered as active and the previous `Target` elements will remain for historical purposes.



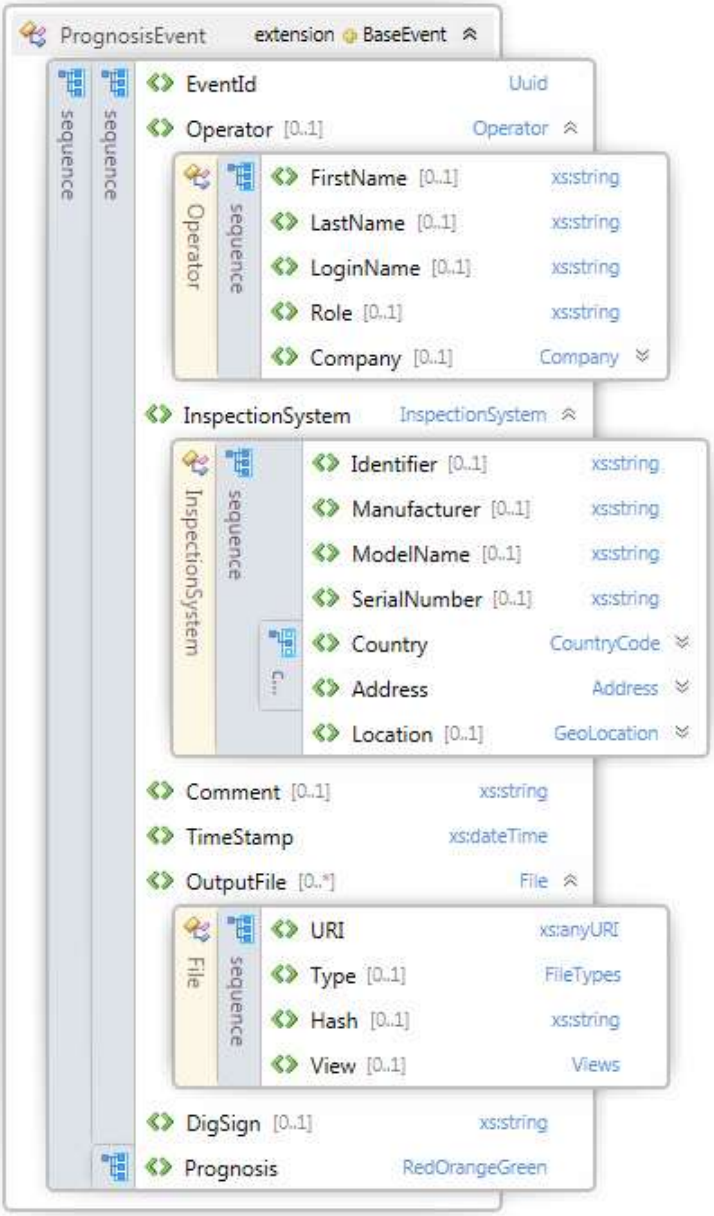
The other fields allow more information to be given about the `Target` (depending on the type of the `Target`). For example, for cargo: origin, destination, owner, shipper, length, weight, driver…). When the manifest is attached to the case, it can be referenced in the XML using the `Document` element.

6.4 Inspection events

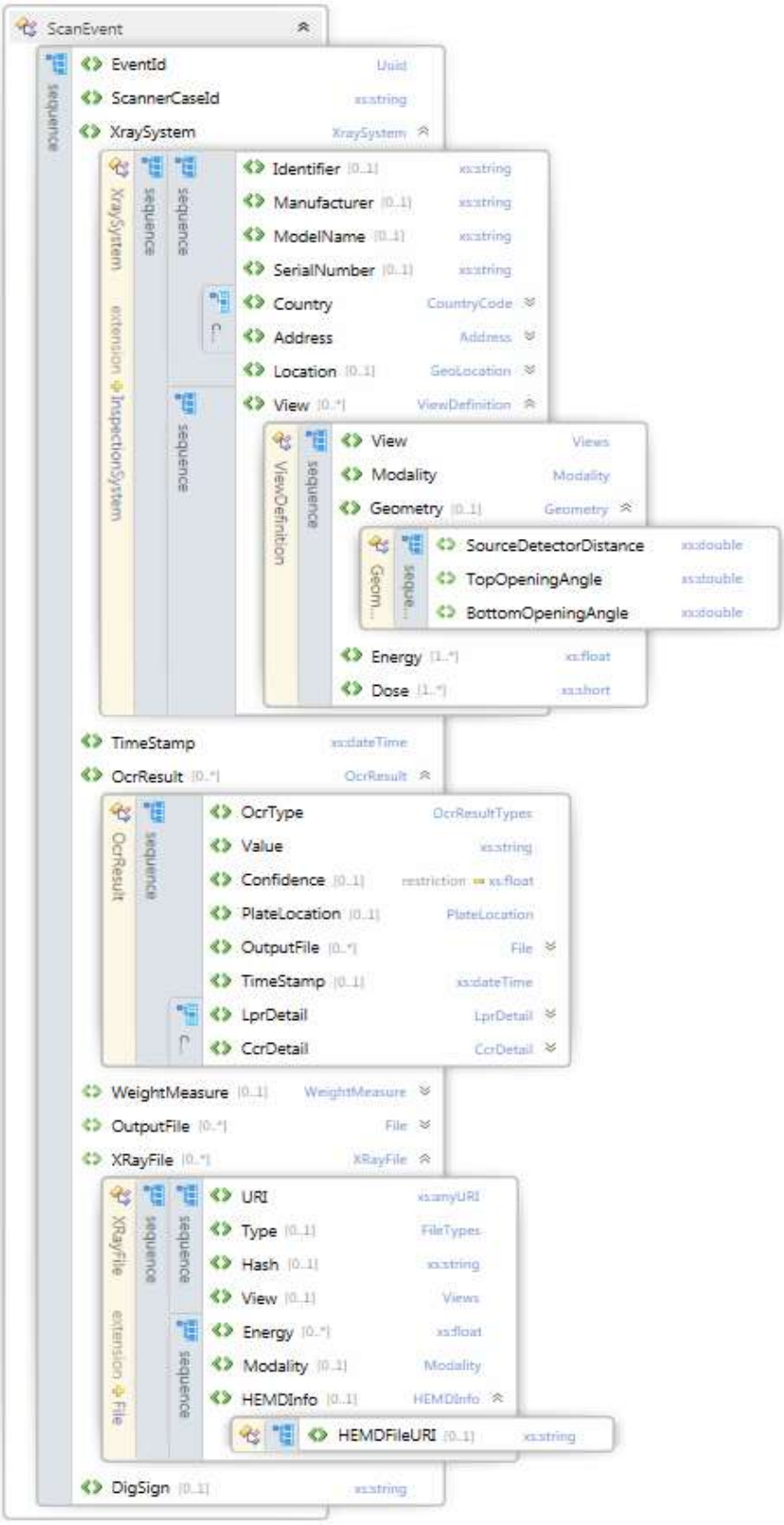
There are several types of inspection event.

6.4.1 Prognosis event

A prognosis event contains information about the system having performed the assessment, the result of the assessment and the date and time of the assessment.



6.4.2 Scan event



A scan event is identified by a `ScannerCaseId` which is provided by the scanner. It contains information about the scanner (ID, manufacturer, geometry, energy…), the date and time of scanning and the list of output files.

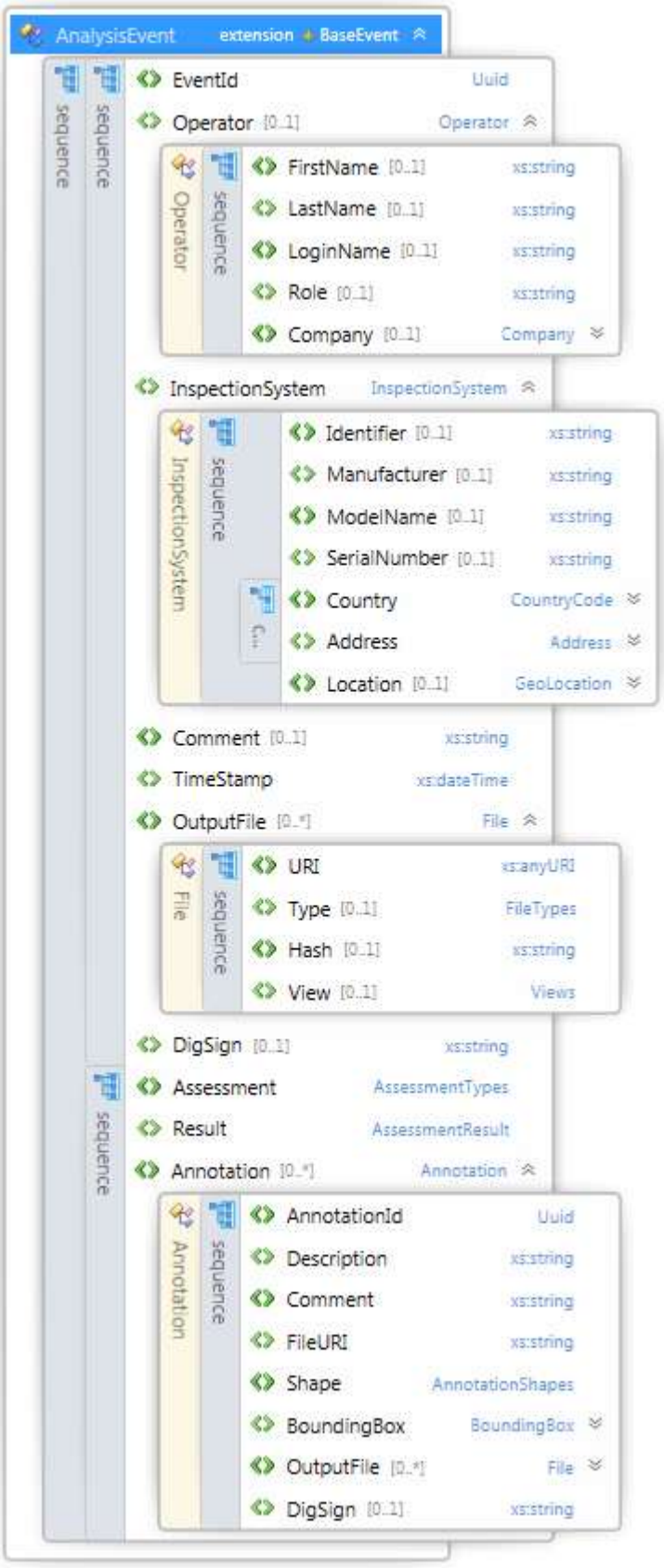
X-Ray and HEMD files received from the scanner must be referenced under `XRayFile` tags.

All other files received from the scanner (such as optical images, OCR files) must be referenced under OutputFile tags.

The file references shall be `<filename>.<ext>`, for instance, `xray12.tif`.

Scan events shall be also used for weight measuring and OCRs.

6.4.3 Analysis event



The analysis event contains information about the operator and/or system having performed the analysis. The type of analysis (automated, image based analysis or physical inspection), the result of the analysis, the date and time of the analysis, a comment and some annotations have to be included.

Each annotation is described by an Annotation element which refers to a file (image) and has a description and a shape. An annotation can be made on an X-ray, HEMD, optical or any other file added to previous events (such as scan events). The `FileURI` shall include the full path related to the UFF2 root.

If an annotation refers to a HEMD file, then viewers shall open also the X-ray file, which is the parent to the HEMD (see Section 6.4.2 Scan event) in order to display the X-ray with HEMD information properly.

All the files created when making the annotation (such as optical images or snapshots of the shapes with any filters applied) must be placed into the event's directory and referenced under the annotation's `OutputFile` tags (`AnalysisEvent:Annotation:OutputFile`). The file references here shall include filename and extension only since the files are placed into the event's directory.

Any additional files created during the analysis and not belonging to any annotation (such as optical images) shall be referenced in the event's `OutputFiles` tag (`AnalysisEvent:OutputFiles`).

6.5 Integrity validation (optional)

Each event element within the XML data file (including target identification) has an optional `DigSign` attribute, which shall contain the digital signature of the section (i.e. the event element and its contents). The signature shall be added as the last step of event creation, when all files and data are defined and are not expected to change for the event being signed.

Example:

```
<DigSign>
------714A286D976BF3E58D9D671E37CBCF7C
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"

MIIB6gYJKoZIhvcNAQcCoIIB2zCCAdcCAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3
DQEHATGCAbYwggGyAgEBMIGcMIGOMQswCQYDVQQGEwJVUzEOMAwGA1UECBMFVGV4
YXMxFDASBgNVBAcTC1NhbiBBbnRvbmlvMQ0wCwYDVQQKEwRVVFNBMQswCQYDVQQL
EwJDUzEXMBUGA1UEAxMOYWkuY3MudXRzYS5lZHUxJDAiBgkqhkiG9w0BCQEWFWp1
bGlldEBhaS5jcy51dHNhLmVkdQIJAMvyApGmAWbKMAkGBSsOAwIaBQCggbEwGAYJ
KoZIhvcNAQkDMQsGCSqGSIb3DQEHATAcBgkqhkiG9w0BCQUxDxcNMDcwMjEyMjI1
ODU4WjAjBgkqhkiG9w0BCQQxFgQUdBfDe/KmnhmYA9DILxfq/zKlvwEwUgYJKoZI
hvcNAQkPMUUwQzAKBggqhkiG9w0DBzAOBggqhkiG9w0DAgICAIAwDQYIKoZIhvcN
AwICAUAwBwYFKw4DAgcwDQYIKoZIhvcNAwICASgwDQYJKoZIhvcNAQEBBQAEQFbJ
+8cZivvgvrjj8l1QbK2o7gWdWBM9yav6NJR2eBVj3hKGaKQ+7JNbygcqtVcMDIo1
jSpsZas33BvhocwGOqs=

------714A286D976BF3E58D9D671E37CBCF7C—

</DigSign>
```

The procedure for digital signing is as follows:

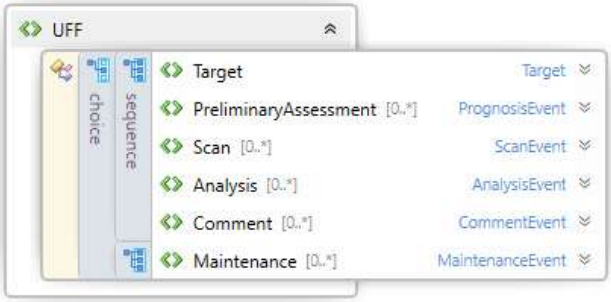
1. Create Event section in the XML and put the event's final metadata into it.
2. Place the event's output files into the event's directory.
3. Add references to the files into the `OutputFiles` tag of the event's metadata.
4. Calculate the checksums of each file using SHA3, 384 bits and place the checksums into the Hash tags of their `OutputFiles` sections.
5. Sign the event's metadata XML section (including all whitespace) with the NII supplier's private key.
6. Add `DigSign` tag as the last element of the event's metadata.

The verification procedure:

1. Remove `DigSign` element including the opening "`<DigSign>`" and closing "`</DigSign>`" tags from the event's metadata.
2. Check the signature of the remaining elements (including all whitespace) using the NII supplier's public key.
3. Recalculate the hashes of all output files in the event's directory.
4. Compare the hashes from the previous step with the ones from the corresponding `Hash` tags.

6.6 Maintenance event

The exchange of maintenance data will be developed in future versions of UFF and will not be described in detail in this document.



6.7 Limitation

In order to simplify UFF 2.0 development and adoption, the NII providers working on this format have agreed to support initially a subset of the UFF 2.0 XML data schema, referred to as the *UFF 2.0 Basic* subset. If a provider's hardware/software supports the UFF 2.0 Basic subset, then it is said to be 'compliant with the UFF 2.0 Basic standard' and/or 'supports the UFF 2.0 Basic standard'. Conversely, if a provider's hardware/software supports the full data schema as defined by this document, then it is said to be 'fully compliant with the UFF 2.0 standard' and/or 'fully supports the UFF 2.0 standard'.

6.7.1 Target

Here are the Target fields that must be supported for compliance with the *UFF 2.0 Basic* standard. Elements marked with an asterisk can appear several times. Some of these fields are optional (refer to the schema to get more detail about that):

- `EventId`
- `InspectionSystem (Identifier, Manufacturer, ModelName, SerialNumber)`
- `TimeStamp`
- `Target (can be either a Container, a Vehicle or a Wagon):`
  - `Container:`
    - `ContainerNum`
    - `Weight`
    - `Declaration (GoodsDescription*/HSCode)`
  - `Vehicle:`
    - `Container* (see above) or Declaration (GoodsDescription*/HSCode)`
    - `FrontPlateNum`
    - `RearPlateNum`
    - `Driver (FirstName, LastName)`
    - `Weight`
  - `Wagon`
    - `Container (see above) or Declaration (GoodsDescription/HSCode)`
    - `TrainId`
    - `WagonId`
    - `Weight`
- `Document*`
  - `URI`

Any other field not part of the above list (such as `DigSign`, as an example) is not required for compliance with the UFF 2.0 Basic standard.

Examples of target definition:


Target - Container.xml


Target - Vehicle.xml


Target - VehicleContainer.xml


Target - Wagon.xml


Target - WagonContainer.xml

6.7.2 Scan event

Amongst the inspection events, the only one which must be supported for compliance with the *UFF 2.0 Basic* standard is the scan event.

If the scan event includes an OCR, the OCR result (container number, license plate number and wagon number) should be stored in the target section of the XML. The images linked to the OCR should be stored in the scan event. The weight is also stored in the Target definition.

The following list of fields must be supported for compliance with the *UFF 2.0 Basic* standard. Elements marked with an asterisk can appear several times. Some of these fields are optional (refer to the schema to get more detail about that):

- `EventId`
- `ScannerCaseId`
- `InspectionSystem (Identifier, Manufacturer, ModelName, SerialNumber)`
- `TimeStamp`
- `OutputFile* (to store OCR images):`
  - `URI`
- `XRayFile*:`
  - `URI`
  - `Energy*`
  - `View`
  - `HEMDInfo:`
    - `HEMDFileURI`

Any other field not part of the above list (such as DigSign, as an example) is not required for compliance with the UFF 2.0 Basic standard.

Examples of scan event:


ScanEvent.xml


ScanEvent - DualView.xml


ScanEvent - HiLow.xml


ScanEvent - OCR.xml

7 Data Exchange and Storage

One of the objective of the UFF development effort is to provide a formal means of exchanging the UFF standard format between systems and interfaces. At this stage, the exchange format remains identical to the architecture used in UFF 1.0 (Exchange of data through file systems). There is, however, an endeavor within the Proof of Concept activities (AKA UFF2.0 demonstration), to create a new mechanism for UFF storage thereby exposing the management of UFF events through more modern means such as REST and HTTP. There is an intention to follow OpenAPI (https://www.openapis.org/) specification in API design.

The description of the REST API will be provided as a separate file.

8 Digital Signatures

There needs to be a way to validate integrity of the data items within the structure. Digital signatures will serve to both provide tamper validation and data integrity check when parts of the structure will be transmitted over (sometimes remote and unreliable) network. Still this can be an optional feature. Refer to Section 6.5 for further details.

9 Archiving

The Root directory that stores individual UFF 2.0 files will eventually grow to an unmanageable size. It is the responsibility of the NII Supplier, along with the customer's consort, to provide an archiving and deletion strategy.

10 UFF 1.0 Compatibility

The new UFF data standard is significant departure from the original UFF definition and the question of retro compatibility comes into question. It may be necessary to create a supplemental activity to the existing PoC project in order to not only provide new adaptors and visualization for NII suppliers, but also a means to transform UFF 1.0 data sets into UFF 2.0 (through XSL Transform).

Backward compatibility of UFF 2.0 to UFF 1.0 is not supported since the previous format supports only a single scan event per dataset. However, UFF 2.0 scan events can be exported as standalone UFF1 datasets if necessary.

11 UFF 2 Dataset Merging

If a cargo is inspected in different disconnected locations, several UFF2 datasets will be generated for the same cargo. The UFF2 datasets can be merged into one dataset by:

- Copying the event directories and their files from the second `.uff` file into the first `.uff` file.
- Appending the contents of the second dataset's XML file (except the `<uff>` and `</uff>` tags) to the content of the `<uff>` element of the first dataset's XML file.

Since the individual event Ids and associated directory names are based on UUIDs, then the risk of naming clashes during the merge process is negligible.