UCam Configuration File Reference Manual

Version 0.3

Stewart McLay

5th March 2010

Acknowledgements

Professor Vikram Dhillon who was P.I. for the UltraCam instrument the project from which UCam was born.

The UCam development team has involved several members of UK ATC staff. Valuable contributions were made by: David Atkinson, Steven Beard, Derek Ives, Stewart McLay, Chris Tierney and Andy Vick.

Table of Contents

1 Introduction

The UCam software is a camera controller and data acquisition application. It is developed for use on camera systems as part of astronomical research instrumentation and is presently in use at some of the leading telescopes in the world today. UCam was originally developed for the UltraCam instrument which is a high temporal resolution triple-beam CCD camera and has subsequently been used on the WFCAM instrument which is a wide field infra-red survey camera. These are just two examples of the wide diversity of astronomical instrumentation that UCam can support but there are others.

The UCam software is primarily designed to make full use of the ARC controller as developed by Astronomical Research Cameras. Although it is a highly configurable system and may be adapted in the future for use with other camera controllers. The UCam design model is application centric where different applications tailored for specific detectors and readout modes are downloaded and executed on the camera controller hardware. This highly configurable application centric design is what has enabled UCam to be an adaptable and reusable solution for several astronomical instruments.

UCam runs on a real-time Linux operating system so it is able to provide fast imaging capabilities for instrumentation where it is needed. It is also designed to be a server application so it can be installed and used remotely across a network running on a standard Linux operating system. At present UCam supports the HTTP protocol and uses the XML format for sending data packets. The interface is simple enough to be accessible through a web browser and there are several client GUI applications and software libraries available for interfacing with the UCam server.

This document is a reference manual for the UCam configuration files. These configurations files are used extensively by UCam at run time. The intended audience for this document are developers who are programming new applications and the technical staff who are responsible for maintaining UCam. The document gives detailed description of the XML file format and elements that make up the configuration files.

1.1 About UltraCam

The UCam software system was originally designed and developed for the UltraCam instrument. UltraCam is an ultra-fast, triple-beam CCD camera which has been designed to study one of the few remaining unexplored regions of observational parameter space - high temporal resolution. The camera, funded by PPARC, saw first light during 2001 and has been used on 2m, 4m and 8m class telescopes in Australia, the Canary Islands, Chile, Greece, South Africa and Spain to study astrophysics on the fastest time scales. More information about UltraCam can be found at http://www.vikdhillon.staff.shef.ac.uk/ultracam.

1.2 About ARC Controller

The UCam software system was originally designed to interface with a ARC controller. The ARC controller is developed by Astronomical Research Cameras, Inc. They design, develop and manufacture controllers for operating CCD and infra-red imaging arrays for astronomical and related applications. They can control a wide variety of imaging arrays in a medley of exposure and readout modes at medium to low speeds with detector limited noise levels. The controllers include electronic circuits, mechanical assemblies, power supplies and supporting software. They are variously known as “Leach controllers”, "ARC controllers", as well as "SDSU controllers" for their origins at San Diego State University. Additionally, we can supply turn-key customized systems

incorporating almost any type of scientific CCD. More information about ARC controllers can be found at http://www.astro-cam.com.

1.3 About UK Astronomy Technology Centre

The UK Astronomy Technology Centre is the national centre for astronomical technology. We design and build instruments for many of the world's major telescopes. We also project-manage UK and international collaborations. Our scientists carry out observational and theoretical research into fundamental questions such as the origins of planets and of galaxies. More information is available at http://www.roe.ac.uk/atc.

1.4 Acronyms And Abbreviations

ARC	Astronomical Research Cameras
CDS	Correlated Double Sampling
Debian	Debian Linux distribution
FITS	Flexible Image Transfer System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
LAN	Local Area Network
NDR	Non Destructive Readout
OS	Operating System
RTAI	Real Time Application Interface
UltraCam	Ultra-fast, triple-beam CCD camera
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
UK ATC	United Kingdom Astronomy Technology Centre
XML	Extensible Mark-up Language

1.5 Stylistic Conventions

Terminal shell commands will be displayed using `Courier` font with the `$` representing a regular user the shell prompt. Everything after the `$` prompt will be a command entered at the terminal shell by a user.

```
$ pwd
```

A `#` shell prompt is used to represent the root user shell.

```
# lsmod
```

Examples of XML data will be displayed using `Courier` font.

```
<Logging>
      <Name>UCam</Name>
```

```
        <Filename>/home/ucam/logfiles/ucam.log</Filename>
        <Level>INFO</Level>
    </Logging>
```

Python shell input and output will also be displayed using `Courier` font with >>> representing the python shell prompt. Again everything after the >>> prompt will be a command entered at the python shell by a user.

```
>>> import sys
>>> print sys.version
2.5.2 (r252:60911, Oct  5 2008, 19:24:49)
[GCC 4.3.2]
```

Examples of Python source code files will be framed with each line prefixed with a line number. The line numbers are included to aid the reader but are not included in the actual source code files. The python code be displayed using `Courier` font.

```
001 #!/usr/bin/env python
002
003 from ucam import UCam
004
005 ucam = UCam()
006 ucam.setAddress('ucamdev.roe.ac.uk')
007 ucam.setPort(9980, 9981, 9982)
008 ucam.connect()
```

Examples of configuration files will be framed with each line prefixed with a line number. The line numbers are included to aid the reader but are not included in the actual configuration text files. The text file data be displayed using `Courier` font.

```
001 logging.loggers.root.channel.class = ConsoleChannel
002 logging.loggers.app.name = Application
003 logging.loggers.app.channel = c1
004 logging.formatters.f1.class = PatternFormatter
005 logging.formatters.f1.pattern = [%p] %t
```

2 Configuration File Overview

The UCam software is designed to be highly configurable to allow it to be reusable for a number of camera applications. This is achieved by using several configuration files which use the XML file format. Below is a short summary of the different configuration files which are described in more detail in the following sections.

- Telescope Configuration: Contains information about the telescope site where the instrument is installed.
- Instrument Configuration: Contains information about the instrument using the UCam software system. Includes information about camera detectors.
- DSP Hardware Configuration: Contains information about the camera controller hardware devices for running UCam software applications. Includes information about processor types and memory configuration.
- Executable Application: An application is defined by a hierarchy of three configuration files. Each file contains particular information relating to application and that is used in a particular way by the UCam software system. A hard reference XML element is specified in the parent configuration file in order to maintain the link to its child configuration file.
  - Configuration File: This file contains variable data about the application that can be altered at run time before and application is executed. For example, the application parameter values that determine the exposure time and the number of exposures to be taken when the application is executed. This XML file can be fetched by a client application that is talking to the UCam server. The client application can then alter the values of the XML elements in the file before posting it back to the UCam server and passing the 'GO' command to execute the application.
  - Application File: This file contains static data about the application which should remain unchanged during run time. The information contained in this file is configuration data that is pertinent to the application and its function. For example, parameters describing how to demultiplexing the raw data and construct an image. This XML file is used internally by the UCam server.
  - DSP Binary File: This file contains the DSP binary executable that is downloaded onto the DSP processor and executed. In essence, it is the application executable that reads the image data from the detector or performs whatever task the application is designed for.
- Configuration File List: Contains a list of all the configuration files available on the UCam servers. Different lists can be compiled into separate files for alternative instruments or instrument configurations. The path and name of the configuration file list is specified in the UCam server properties file which is read on start-up by the UCam servers.

3 Telescope Configuration File Reference

The telescope configuration file contains information about the telescope where the instrument is installed. The diagram shown in figure 1 shows the XML tree of elements that form the telescope XML file. Each of the elements are described in more detail in the following subsections.

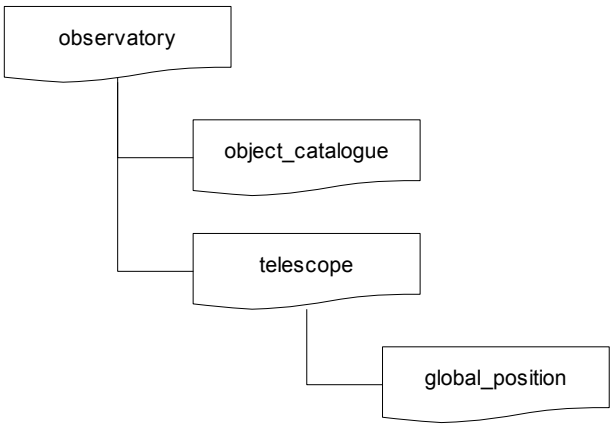

Figure 1: Telescope file XML tree

3.1 Observatory Element

Element	observatory
Multiplicity	One
Description	Root node element for telescope configuration file document. Contains the observatory name and time zone offset.
Value	Unused
Attributes	
Name	**Value**
name	The name of the observatory.
timezone	The UTC offset (e.g. “+1” hours).

3.2 Object Catalogue Element

Element	object_catalogue
Multiplicity	Zero or one
Description	The observatory's object catalogue data file.
Value	The path for the observatory's object catalogue data file.

3.3 Telescope Element

Element	telescope
Multiplicity	One
Description	Details of the telescope the camera is operating on.

Value	Unused
Attributes	
Name	**Value**
id	Telescope identifier.
name	The name of the telescope the camera is installed on.

3.4 Global Position Element

Element	global_position
Multiplicity	One
Description	Details of the telescope's location.
Value	Unused
Attributes	
Name	**Value**
height	Height in metres above sea level.
latitude	Latitude in degrees North.
longitude	Longitude in degrees West.

4 Instrument Configuration File Reference

The instrument configuration file contains information about the instrument using the UCam software system. The diagram shown in figure 2 shows the XML tree of elements that form the instrument XML file. Each of the elements are described in more detail in the following subsections.

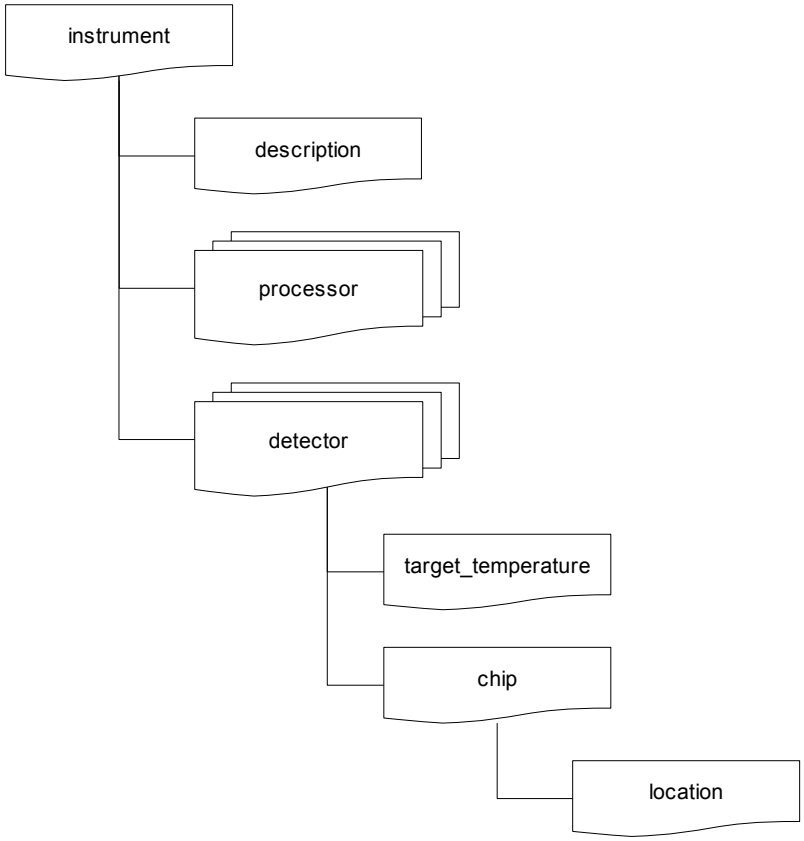

Figure 2: Instrument file XML tree

4.1 Instrument Element

Element	instrument
Multiplicity	One
Description	Root node element for instrument configuration file document. Contains the instrument name and type.
Value	Unused
Attributes	
Name	**Value**
name	The name of the instrument.
type	Short description of the type of instrument (e.g. “camera”).

4.2 Description Element

Element	description
Multiplicity	Zero or one
Description	Instrument description.
Value	Description of the instrument.

4.3 Processor Element

Element	processor
Multiplicity	One or more
Description	Details for a processor for downloading and executing applications.
Value	Unused
Attributes	
Name	**Value**
ref	Processor reference.
name	Processor name.
xlink:href	Hard reference link URI to the processor configuration XML file. The value should should contain the full file name including the file extension.

4.4 Detector Element

Element	detector
Multiplicity	One or more
Description	Details for a detector.
Value	Unused
Attributes	
Name	**Value**
id	Detector identifier.
type	The type of detector (e.g. “CCD” or “IR”).

4.5 Target Temperature Element

Element	target_temperature
Multiplicity	One
Description	Target detector chip temperature.
Value	The target temperature for the detector chip in the focal plane.

4.6 Chip Element

Element	chip

Multiplicity	One
Description	Details for a detector chip.
Value	Unused
Attributes	
Name	**Value**
id	Detector chip identifier.
name	Detector chip name.
type	The type of detector chip.
columns	Number of columns of pixels on the detector chip.
rows	Number of rows of pixels on the detector chip.

4.7 Location Element

Element	location
Multiplicity	One
Description	Detector chip's location in the instrument's focal plane.
Value	Unused
Attributes	
Name	**Value**
xshift	The difference in *fpunits* between the centre of the chip and the instrument optical axis, projected onto the instrument X axis.
yshift	The difference in *fpunits* between the centre of the chip and the instrument optical axis, projected onto the instrument Y axis.
xscale	The number of *fpunits* per pixel along the detector's column axis.
yscale	The number of *fpunits* per pixel along the detector's row axis.
rotation	Any rotation in degrees clockwise between the chip's column axis and the instrument's X axis. (Should be zero if they are perfectly aligned).
fpunits	The name of the focal plane units (e.g. “microns” or “pixels”)

5 Processor Configuration File Reference

The processor configuration file contains information about the camera controller hardware devices for running UCam software applications. The diagram shown in figure 3 shows the XML tree of elements that form the DSP hardware XML file. Each of the elements are described in more detail in the following subsections.

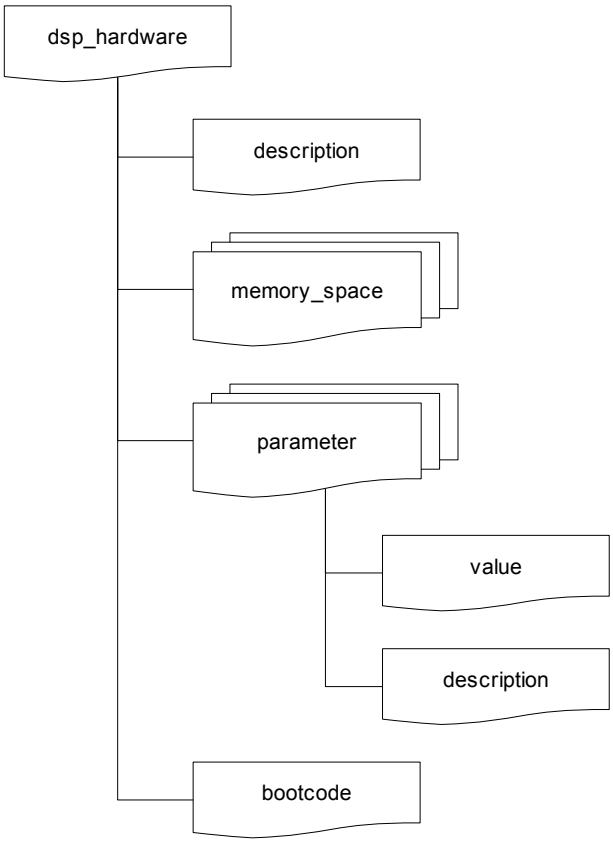

Figure 3: DSP hardware file XML tree

5.1 DSP Hardware Element

Element	dsp_hardware
Multiplicity	One
Description	Root node element for DSP hardware configuration file document. Contains the DSP device identifier, name and type.
Value	Unused
Attributes	
Name	**Value**
id	DSP hardware identifier.
name	The name of the DSP hardware device.
type	DSP processor type (e.g. “DSP56300”).

5.2 Description Element

Element	description
Multiplicity	One
Description	Description of the DSP hardware device.
Value	Comment describing the function of the DSP hardware device.

5.3 Memory Space Element

Element	memory_space
Multiplicity	One
Description	Memory space available on the DSP hardware device.
Value	Unused
Attributes	
Name	**Value**
id	Memory space identifier (e.g. P, X or Y).
mask	Memory type mask • 0x100000 P memory space • 0x200000 X memory space • 0x300000 Y memory space
start	Starting memory address (e.g. 0x000000).
finish	Finishing memory address (e.g. 0x001fff).

5.4 Parameter Element

Element	parameter
Multiplicity	Zero or more
Description	DSP device memory space parameter.
Value	Unused
Attributes	
Name	**Value**
id	Parameter identifier.
type	Data type (e.g. int, uint, float).
access	Memory access mode (e.g. read, write and readwrite).
fulladdress	Parameter memory address (e.g. 0x0000ff).

5.5 Value Element

Element	value
Multiplicity	One
Description	Parameter default value.

Value	Unused
Attributes	
Name	**Value**
default	Parameter default value.

5.6 Description Element

Element	description
Multiplicity	Zero or one
Description	Parameter description.
Value	Comment describing the parameter.

5.7 Boot Code Element

Element	bootcode
Multiplicity	One
Description	Describes the boot code installed on the DSP hardware device.
Value	Unused
Attributes	
Name	**Value**
xlink:href	Hard reference link URI to the boot code XML file. The value should should contain the full file name including the file extension.

6 Application File Hierarchy

UCam was designed to be a highly configurable software system. The application configuration files are core to the UCam software design philosophy. An application is defined by a hierarchy of three configuration files as shown in figure 4. Each file contains particular information relating to application and that is used in a particular way by the UCam software system. A hard reference XML element is specified in the parent configuration file in order to maintain the link to its child configuration file.

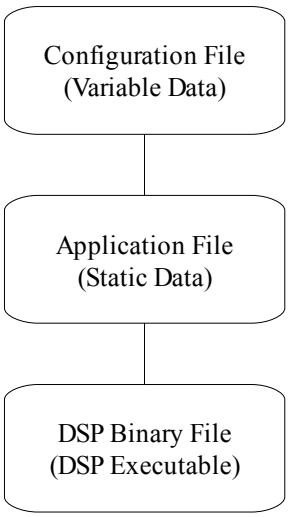

Figure 4: Application XML configuration file hierarchy

- Configuration File: This file contains variable data about the application that can be altered at run time before and application is executed. For example, the application parameter values that determine the exposure time and the number of exposures to be taken when the application is executed. This XML file can be fetched by a client application that is talking to the UCam server. The client application can then alter the values of the XML elements in the file before posting it back to the UCam server and passing the 'GO' command to execute the application.
- Application File: This file contains static data about the application which should remain unchanged during run time. The information contained in this file is configuration data that is pertinent to the application and its function. For example, parameters describing how to demultiplexing the raw data and construct an image. This XML file is used internally by the UCam server.
- DSP Binary File: This file contains the DSP binary executable that is downloaded onto the DSP processor and executed. In essence, it is the application executable that reads the image data from the detector or performs whatever task the application is designed for.

The following sections describe each of the application XML files in more detail. Listing and describing all of their XML elements.

6.1 Configuration File Reference

The diagram shown in figure 5 shows the XML tree of elements that form the configuration XML file. Each of the elements are described in more detail in the following subsections.

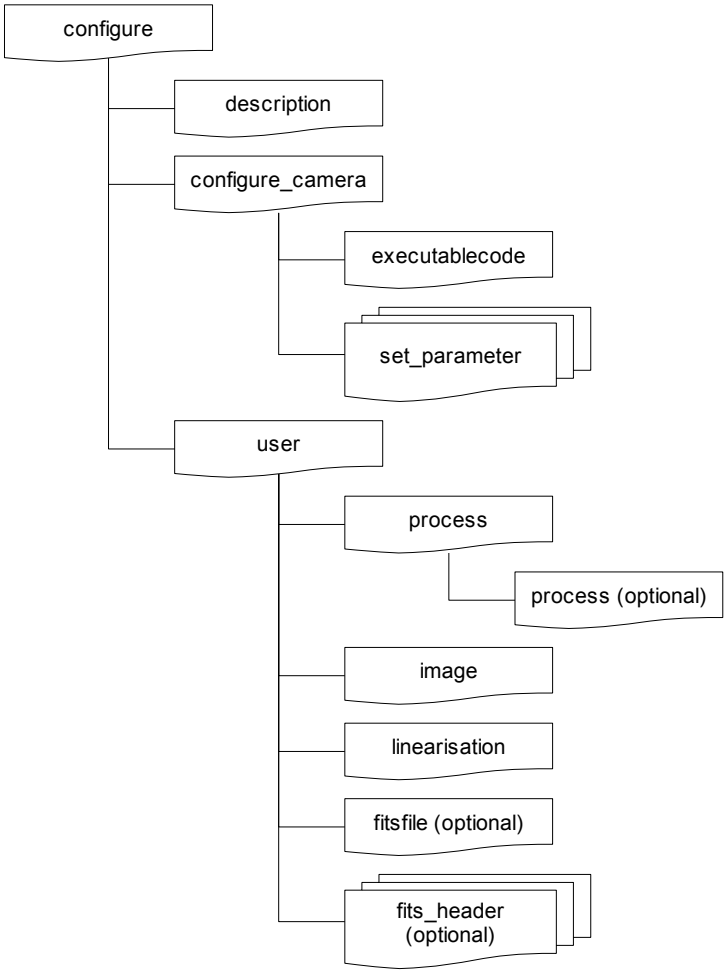

Figure 5: Configuration file XML tree

6.1.1 Configure Element

Element	configure
Multiplicity	One
Description	Root node element for configuration file document. Contains basic identification details such as author, date, etc.
Value	Unused
Attributes	
Name	**Value**
id	A string identifier (or name) describing the configuration.
user	The name of the file's author who created the file.
datetime	The date and time the configuration file was created. The format used for this attribute is usually “day-month-year” (e.g. “09-Oct-2006”).

6.1.2 Description Element

Element	description
Multiplicity	One
Description	Provides a short description of what the configuration file is used for.
Value	Short description of the purpose of the configuration file.

6.1.3 Configure Camera Element

Element	configure_camera
Multiplicity	One
Description	Parent element node for the camera controller configuration elements.
Value	Unused

6.1.4 Executable Code Element

Element	executablecode
Multiplicity	One
Description	Contains a hard reference link URI to the child application XML file. It can also be configured to forcibly download applications before they are execution.
Value	Unused
Attributes	
Name	**Value**
force_download	If the value is “yes” then the binary code will be forcibly downloaded on to the camera controller hardware even if it is already loaded. If the value is “no” the binary code will only be downloaded if it is not already loaded.
xlink:href	Hard reference link URI to the application XML file. The value should should contain the full file name including the file extension.

6.1.5 Set Parameter Element

Element	set_parameter
Multiplicity	Zero or many
Description	Specifies an application a parameter value. All parameters are automatically written as FITS headers to the FITS image data file.
Value	Unused

Attributes	
Name	**Value**
ref	The parameter reference or the parameter name. This value should match the name of the parameter as used by the DSP executable. The reference will be used for the FITS header keyword.
value	The default parameter value. The value will be used as the FITS header value.
min (optional)	The minimum parameter value is an optional attribute. This attribute is only applicable to parameters that have integer values.
max (optional)	The maximum parameter value is an optional attribute. This attribute is only applicable to parameters that have integer values.
visibility (optional)	Controls the visibility of the parameter for different application modes. These modes are have an incremental ranking for visibility where higher ranking modes are able to view lower ranking mode elements. The different modes and their rankings are listed: 1. user: normal user mode 2. engineering: engineering mode for technical support staff 3. admin: administration mode for technical support staff 4. none: permanently hidden from all users

The *set_parameter* is a very important element for applications. A number of parameter values have been reserved for special use on all DSP applications. These parameters are used by the demultiplexing process for sampling the raw image data. Table ??? shows the list of reserved application parameter names. Application parameters are automatically written to FITS files as FITS file headers. The parameter name is used as the FITS header keyword and therefore should follow the FITS convention of a maximum of 8 characters using upper case letters.

Table 1: Reserved application parameters

Parameter Name	Description
RD_TIME	Time to read out the detector (milliseconds)
RS_TIME	Time to reset the detector (milliseconds)
DWELL	The dwell time for the exposure (milliseconds)
NUM_READ	Number of read outs per exposure
NUM_NULL	Number of NULL reads per exposure
NSAMPLES	Number of sample pixel read outs
NUM_EXPS	Number of exposures
X[N]_START	X axis start position (Starting column number)
Y[N]_START	Y axis start position (Starting row number)
X[N]_SIZE	X axis size (columns)
Y[N]_SIZE	Y axis size (rows)
X_BIN	X axis binning size (pixels)
Y_BIN	Y axis binning size (pixels)

Please not that [N] denotes a single numerical character between 1-9.

6.1.6 User Element

Element	user
Multiplicity	One
Description	Parent element node for user definable elements. The user's client application can include any XML elements of their own as child elements of the user node. These elements will be passed write through to the resulting XML file which accompanies the raw data file that is generated by the DSP application. They are useful for writing additional attributes for use in post data sampling. However, there are some elements already reserved for use in UCam.
Value	Unused

6.1.7 Process Element

Element	process
Multiplicity	One or two (nested)
Description	The data sampling mode to be used on the raw image data. Two process elements can be nested for certain combination of sampling modes.
Value	Unused
Attributes	

Name	Value
type	Sampling modes supported include: • SRR – Single Raw Readout (basic imaging) • CDS – Correlated Double Sampling • Fowler – Fowler sampling • NDR-ABS – Non Destructive Readout (absolute) • NDR-SLOPE – Non Destructive Readout (slope) Parent process sampling modes include: • COADD – Add exposures • MEAN – Mean average exposure
threshold	Only applicable for the NDR sampling modes. Specifies the threshold pixel sample value to be used for threshold limited NDR sampling.

Process nodes can be nested as shown in the example below. In this example the raw image data would be sampled using a Threshold Limited Non-Destructive Readout (TLNDR) mode with slope. The threshold sample limit would be 32000 ADU. The exposures for each TLNDR would then be averaged to provide a single mean image. Please note that the maximum depth supported for nested process nodes is two. When process nodes are nested the parent node can only specify *COADD* or *MEAN* sampling modes.

```
...
<user>
      <process type="MEAN">
            <process type="NDR-SLOPE" threshold="32000"/>
      </process>
</user>
...
```

6.1.8 Image Element

Element	image
Multiplicity	Zero or one
Description	Contains additional attributes for image processing.
Value	Unused
Attributes	
Name	**Value**
fullframe	If "yes" then the image data read out is mapped onto a full detector size image where the missing pixels have a zero pixel value. If "no" then the image will only consist of the pixel data read out.

6.1.9 Linearisation Element

Element	linearisation
Multiplicity	Zero or one
Description	Used for applying linearisation to raw image data.
Value	Unused

Attributes	
Name	**Value**
filename	The name of the FITS file used for linearisation. The linearisation FITS file should contain a single array of linearisation values.

6.1.10 Fits File Element

Element	fitsfile
Multiplicity	Zero or one
Description	Specifies FITS file data storage attributes.
Value	Unused
Attributes	
Name	**Value**
prefix	The prefix part of the FITS file name.
zerofill	The number of zero fill characters to be used for the sequence number part of the FITS file name.
format	Specify “cube” for a 3D data cube or “extended” for an extended FITS file format.
storage	This is only really applicable to windowing applications. Specify “single” for a single FITS file where windowed images are combined into a single stacked image. Or specify “multiple” for separate FITS files for each window.
compression	Specify the compression algorithm to be used when writing the FITS image file. Supported compression algorithms include: ● none – No compression used ● gzip – GNU zip compression ● rice – RICE compression ● hcompress - HCompression

6.1.11 Fits Header Element

Element	fits_header
Multiplicity	Zero or many
Description	Specifies FITS file header.
Value	FITS header comment.
Attributes	
Name	**Value**
name	FITS header keyword value.
value	FITS header value.
type	FITS header value type where the list of supported types include

	“logical”, “string”, “uint”, “int”, “ulong”, “long”, “float” and “double”.

6.2 Application File Reference

The diagram shown in figure 6 shows the XML tree of elements that form the application XML file. Each of the elements are described in more detail in the following subsections.

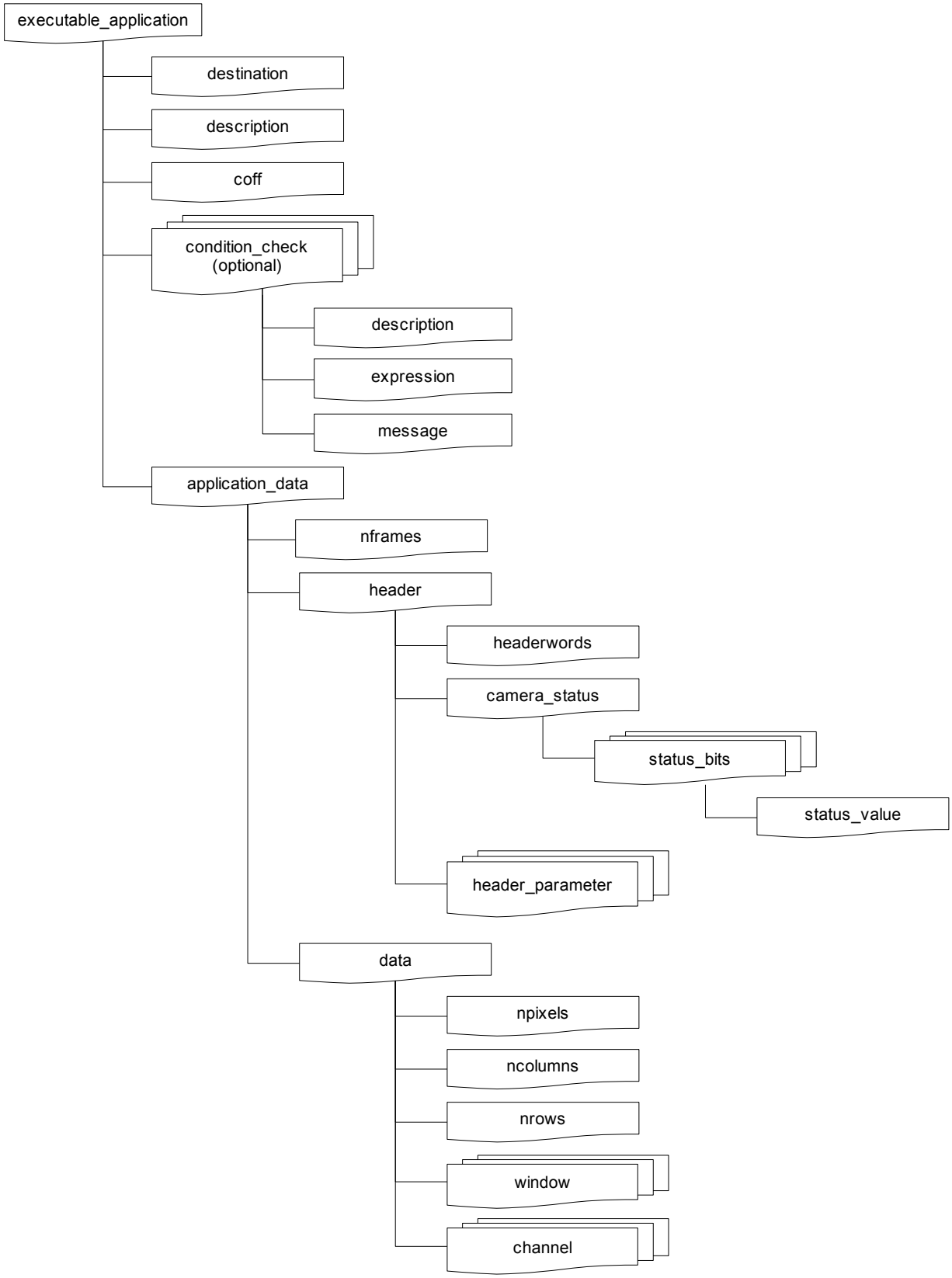

Figure 6: Application file XML tree

6.2.1 Executable Application Element

Element	executable_application
Multiplicity	One
Description	Root node element for application file.
Value	Unused
Attributes	
Name	**Value**
id	Application identifier.
boot	???
name	Application name.
version	Application version number.
author	Application author's name.

6.2.2 Destination Element

Element	destination
Multiplicity	One
Description	Destination hardware platform this application is designed to run on.
Value	Unused
Attributes	
Name	**Value**
ref	Hardware reference.
xlink:ref	Hard reference link URI to the hardware XML file. The value should should contain the full file name including the file extension.

6.2.3 Description Element

Element	description
Multiplicity	One
Description	Description of what the application does.
Value	Application description.

6.2.4 Coff Element

Element	coff
Multiplicity	One
Description	Hard reference link URI to DSP binary XML file.

Value	Unused
Attributes	
Name	**Value**
xlink:ref	Hard reference link URI to the DSP binary XML file. The value should should contain the full file name including the file extension.

6.2.5 Conditional Check Element

Element	condition_check
Multiplicity	One
Description	A conditional check to be carried out.
Value	Unused
Attributes	
Name	**Value**
when	When the conditional check should be tested. The when attribute can take one of the following values: ● start - check these conditions after downloading the application ● pre - check these conditions before executing the application ● post - check these conditions after executing the application ● reset - check these conditions after executing a reset
fatal	Treat a failure to satisfy the conditional check as a fatal exception (“Y” for yes and “N” for no).

6.2.6 Description Element

Element	description
Multiplicity	One
Description	Description of what the conditional check is testing.
Value	Conditional check description.

6.2.7 Expression Element

Element	expression
Multiplicity	One
Description	The conditional check expression to be tested.
Value	The conditional check expression. Expressions may contain parameter value names that will be substituted with the real value when the expression is evaluated. The list of supported operators and their precedence are: ● `( )` brackets ● `* / %` arithmetic operators ● `+ -` ● `< > = #` conditional operators (# for “not equal”) ● `& |`

Attributes	
Name	**Value**
expect	Whether or not the conditional check is expected to test true or false (“T” for true and “F” for false).

6.2.8 Message Element

Element	message
Multiplicity	One
Description	Message reported in the event the conditional check occurs.
Value	Conditional check event message.

6.2.9 Application Data Element

Element	application_data
Multiplicity	One
Description	Describes the quantity, type and format of the data which this application generates.
Value	Unused
Attributes	
Name	**Value**
id	Application data identifier.
type	Data word type can be “int”, “uint” and “float”.
wordsize	The number of bytes that make up the data word size.

6.2.10 Number Of Data Frames Element

Element	nframes
Multiplicity	One
Description	Number of image data frames read out.
Value	An expression that may contain parameter value names that will be substituted with the real value when the expression is evaluated. The list of supported operators and their precedence are: ● ( ) brackets ● * / % arithmetic operators ● + - ● < > = # conditional operators (# for “not equal”) ● & |

6.2.11 Header Element

Element	header

Multiplicity	One
Description	Each frame of data is assumed to consist of a header followed by data. The header element contains child element that describe the words contained in the data header at the beginning of each frame.
Value	Unused

6.2.12 Header Words Element

Element	headerwords
Multiplicity	One
Description	Number of data words in the header.
Value	An expression that may contain parameter value names that will be substituted with the real value when the expression is evaluated. The list of supported operators and their precedence are: ● ( ) brackets ● * / % arithmetic operators ● + – ● < > = # conditional operators (# for “not equal”) ● & |

6.2.13 Camera Status Element

Element	camera_status
Multiplicity	One
Description	The camera status field is the first word(s) in the header. It contains bit flags that report the status of the camera controller and data quality when the data is being read out.
Value	Unused
Attributes	
Name	**Value**
type	Data word type can be “int” or “uint”.
length_words	The number of data words that make up the camera status field.

6.2.14 Status Bits Element

Element	status_bits
Multiplicity	Zero or many
Description	Definition of a bit(s) flag that forms part of the camera status field.
Value	Unused
Attributes	
Name	**Value**
name	Status bits flag name.

mask	Hexadecimal value for masking the bit(s) in the camera status field.
expected	The value normally expected during operation.
id	A unique identifier.

6.2.15 Status Value Element

Element	status_value
Multiplicity	Zero or many
Description	Definition of a status bit value.
Value	A message reporting the status value.
Attributes	
Name	**Value**
value	The value tested against the masked of bits.
fatal	Whether or not if this value should be treated as a fatal failure and the application should be halted.

6.2.16 Header Parameter Element

Element	header_parameter
Multiplicity	Zero or many
Description	Header parameter value.
Value	Parameter description.
Attributes	
Name	**Value**
id	Parameter identifier.
type	Parameter data type. Supported types include “int”, “uint” and “float”.
start_word	Starting word offset from start of header.
length_words	Header parameter size in data words.

6.2.17 Data

Element	data
Multiplicity	One
Description	Describes the format and shape of the data and the instructions on how to demultiplex it.
Value	Unused
Attributes	
Name	**Value**
fullframe	Set to “yes” when data are full frame images and “no” when data images are divided into windows.

6.2.18 Number Of Pixel Element

Element	npixels
Multiplicity	One
Description	Number of pixels contained in the image data.
Value	An expression that may contain parameter value names that will be substituted with the real value when the expression is evaluated. The list of supported operators and their precedence are: ● ( ) brackets ● * / % arithmetic operators ● + – ● < > = # conditional operators (# for “not equal”) ● & |

6.2.19 Number Of Columns Element

Element	ncolumns
Multiplicity	One
Description	Number of columns of pixels contained in the image data.
Value	An expression that may contain parameter value names that will be substituted with the real value when the expression is evaluated. The list of supported operators and their precedence are: ● ( ) brackets ● * / % arithmetic operators ● + – ● < > = # conditional operators (# for “not equal”) ● & |

6.2.20 Number of Rows Element

Element	nrows
Multiplicity	One
Description	Number of rows of pixels contained in the image data.
Value	An expression that may contain parameter value names that will be substituted with the real value when the expression is evaluated. The list of supported operators and their precedence are: ● ( ) brackets ● * / % arithmetic operators ● + – ● < > = # conditional operators (# for “not equal”) ● & |

6.2.21 Window Element

Element	window
Multiplicity	One or many

Description	Provides the attributes for a window that is used for mapping image data pixels when the data is being demultiplexed.
Value	Unused
Attributes	
Name	**Value**
id	Unique window identifier.
name	Descriptive window name.
join	Join identifier used for matching with channel elements.
xleft	X axis left most pixel coordinate value.
ybottom	Y axis bottom most pixel coordinate value.
xsize	X axis size (pixels).
ysize	Y axis size (pixels).

6.2.22 Channel Element

Element	channel
Multiplicity	One or many
Description	Provides the attributes for a data channel of image pixel data read out from the detector.
Value	Unused
Attributes	
Name	**Value**
id	Unique window identifier.
name	Descriptive channel name.
chip	Detector chip identifier.
join	Join identifier used for matching with window elements.
index	The primary index for mapping pixels. If the detector channel reads out columns fast and rows slow then the index should be “col”. If the detector reads out rows fast and columns slow then the index should be “row”.
stepcol	The step value for mapping pixels on to a window column. A positive value (+1) should be used for incremental columns to the right. A negative value (-1) should be used for decremental columns to the left. The size of the value denotes the magnitude of the shift in column pixels although this value is normally 1.
steprow	The step value for mapping pixels on to a window row. A positive value (+1) should be used for incremental rows upwards. A negative value (-1) should be used for decremental rows downwards. The size of the value denotes the magnitude of the shift in row pixels although this value is normally 1.
offset	The channel pixel offset value. For example, an offset of 4 would imply

	that every fourth pixel in the image data belongs to this channel. Although the channel size can be used when a block of pixels are read out from a detector channel.
Size (optional)	Number of pixels read out in successively from a detector channel. Default value is 1.

6.2.23 How To Define A Conditional Check

This section describes how to write a conditional check. The example below shows the XML for defining a conditional check that tests the X_BIN parameter value is greater than 0 and less than 9. The test is carried just before the application is executed and it will be halted with a fatal error message given in response if the test fails.

```
<condition_check when="pre" fatal="Y">
      <description>Check X binning factor</description>
      <expression expect="T">(X_BIN&amp;gt;0)&amp;amp;(X_BIN&amp;lt;9)</expression>
      <message>Invalid X binning selection</message>
</condition_check>
```

The *condition_check* element's *when* attribute can have the following values that determine when the test is carried out.

- start - check these conditions after downloading the application.
- pre - check these conditions before executing the application.
- post - check these conditions after executing the application.
- reset - check these conditions after executing a reset.

The *condition_check* element's *fatal* attribute has a logical value which determines if the application should be halted in the event that that the test fails. The attribute has a boolean value (the parser supports T,Y,1 for true and F,N,0 for false). A true value will cause failure of this check to cause a halt in processing and return of an error, a false value will return an error but continue processing

The *expression* element's *expect* attribute is a logical value. It represents the value of the expression expected on success (which allows you to use either positive or negative logic in the expressions).

Table 2: Operators recognised by the expression parser

Operator	**XML**	**Meaning**
+	+	Addition
-	-	Subtract
*	*	Multiplication
/	/	Division
%	%	Modulo
<	&lt;	Less than

>	&gt;	Greater than
=	=	Equal to
#	#	Not equal to
|	|	Logical OR
&	&amp;	Logical AND

6.2.24 How To Define The Rules For Demultiplexing Image Data

The window and channel elements are used together to describe the rules for demultiplexing the raw image data read out from the detector. The window elements are used to define windows where pixels are mapped to create the final demultiplexed image. The channel elements describe the raw image pixel data that has been readout from the detector and how they should be mapped to particular windows when the image data is being demultiplexed.

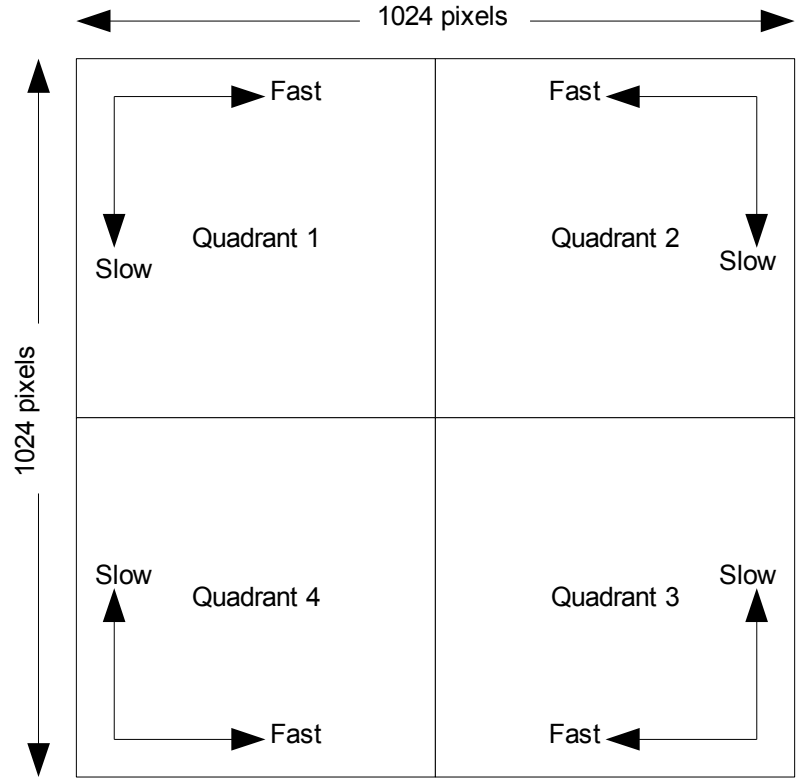

Figure 7: An example of an infra-red detector

Figure 7 shows a diagram that gives an example of how the pixels are read out from a infra-red detector. The detector is 1024 x 1024 pixels and has 4 quadrants. The detector has 4 read out channels one per quadrant. The arrows tell us where the first pixel read out from quadrant is to be found. The arrows are labelled fast and slow to show the order in which the pixels are read for each quadrant. For example, in quadrant 1 the first pixel read out is in the top left corner. The pixels are read out from the left most column to the right most column before moving down to the next row and reading the next row of pixels. The window and channel elements for demultiplexing this

quadrant are shown below.

```
<window id="w1" name="win1" join="win1" xleft="0" ybottom="512"
      xsize="512" ysize=”512”/>
<channel id="c1" name="chan1" chip="AladdinII" join="win1" index="col"
      stepcol="+1" steprow="-1" offset="0" size=”1”/>
```

The window element and its attributes describe the area of pixels readout from the first quadrant. The attributes *xleft*, *ybottom*, *xsize* and *ysize* describe the size and coordinates for the quadrant. The channel element and its attributes describe how to fill the window element. Notice that the window and channel elements both have an attribute called *join* which have matching values. This tells the demultiplexing software that the pixel data from channel 1 are to be mapped to window 1. The *index* element states that the primary index is columns and therefore rows must be the secondary index. The *stepcol* states that the pixels are written to the window incrementally in the X axis (from left to right) one pixel at a time. The *steprow* states that the pixels are written to the window decrementally in the Y axis (from top to bottom) one pixel at a time. The *offset* and *size* state that first pixel belongs to channel 1 and that only 1 pixel should be read at a time for that channel.

The complete set of window and channel elements are shown below. As the detector has four channels (one per quadrant) the pixels are multiplexed as one pixel from each channel is read out in turn. Therefore each successive channel element has a higher *offset* attribute value.

```
<window id="w1" name="win1" join="win1" xleft="0" ybottom="512"
      xsize="512" ysize=”512”/>
<window id="w2" name="win2" join="win2" xleft="512" ybottom="512"
      xsize="512" ysize=”512”/>
<window id="w3" name="win3" join="win3" xleft="512" ybottom="0"
      xsize="512" ysize=”512”/>
<window id="w4" name="win4" join="win4" xleft="0" ybottom="0"
      xsize="512" ysize=”512”/>
<channel id="c1" name="chan1" chip="AladdinII" join="win1" index="col"
      stepcol="+1" steprow="-1" offset="0" size=”1”/>
<channel id="c2" name="chan2" chip="AladdinII" join="win2" index="col"
      stepcol="-1" steprow="-1" offset="1" size=”1”/>
<channel id="c3" name="chan3" chip="AladdinII" join="win3" index="col"
      stepcol="-1" steprow="+1" offset="2" size=”1”/>
<channel id="c4" name="chan1" chip="AladdinII" join="win1" index="col"
      stepcol="+1" steprow="+1" offset="3" size=”1”/>
```

6.3 DSP File Reference

The diagram shown in figure 8 shows the XML tree of elements that form the DSP binary XML file. Each of the elements are described in more detail in the following subsections.

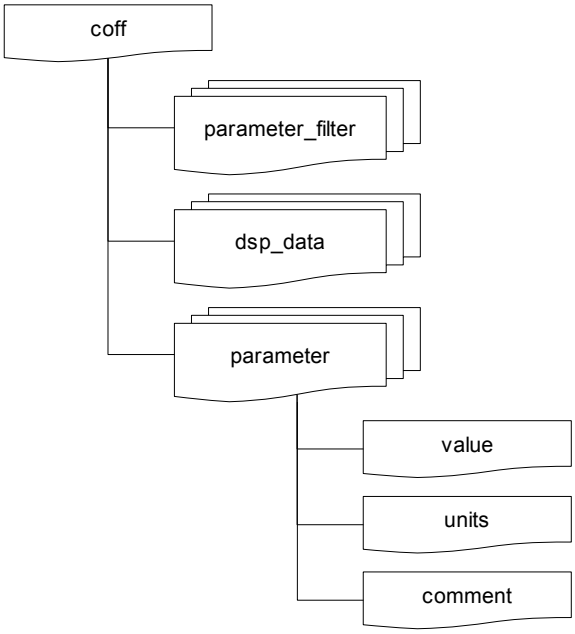

Figure 8: DSP binary XML tree

6.3.1 Coff Element

Element	coff
Multiplicity	One
Description	Root node element for the DSP binary data.
Value	Unused
Attributes	
Name	**Value**
id	Application identifier.
version	Application version number.
device	The device platform that supports this application. Supported devices include “DSP56300”.

6.3.2 Parameter Filter Element

Element	parameter_filter
Multiplicity	Zero or many
Description	Defines memory spaces where the application parameter values are allocated.
Value	Unused
Attributes	

Name	Value
memory_space_ref	Device memory type reference. On the DSP56300 platform the memory spaces are “P”, “X”, and “Y” memory.
readstart	Starting read memory address.
writestart	Starting write memory address.
readfinish	Finishing read memory address.
writefinish	Finishing write memory address.

6.3.3 DSP Data Element

Element	dsp_data
Multiplicity	Zero or many
Description	Defines the binary data for the executable application that is downloaded onto the camera controller device.
Value	The executable application binary data image. The binary data is written as a sequence of hexadecimal values. For example, “0x0A00A1 0x0004F5 0x0D00B5 ...” etc.
Attributes	
Name	**Value**
memory_space_ref	Device memory type reference. On the DSP56300 platform the memory spaces are “P”, “X”, and “Y” memory.
address	Starting memory address.
fulladdress	Full starting memory address including the memory space reference code.

6.3.4 Parameter Element

Element	parameter
Multiplicity	Zero or many
Description	Defines an application parameter value.
Value	Unused
Attributes	
Name	**Value**
id	Parameter identifier or name.
type	Parameter data type. Supported types include “int” and “float”.
access	Parameter access permissions. Supported access rules include “none”, “read”, “write” and “readwrite”.
memory_space_ref	Device memory type reference. On the DSP56300 platform the memory spaces are “P”, “X”, and “Y” memory.
address	Starting memory address.

fulladdress	Full starting memory address including the memory space reference code.

6.3.5 Value Element

Element	value
Multiplicity	Zero or one
Description	Defines the default application parameter value.
Value	Default parameter value.

6.3.6 Units Element

Element	units
Multiplicity	Zero or one
Description	Defines the application parameter units.
Value	Parameter units.

6.3.7 Comment Element

Element	comment
Multiplicity	Zero or one
Description	Descriptive comment about the parameter.
Value	Parameter description.

7 Configuration File List Reference

The configuration file list file is important because it is read by the UCam server at start-up. It lets the server know what the list of available configuration files are. The diagram shown in figure 9 shows the XML tree of elements that form the configuration file list XML file. Each of the elements are described in more detail in the following subsections.

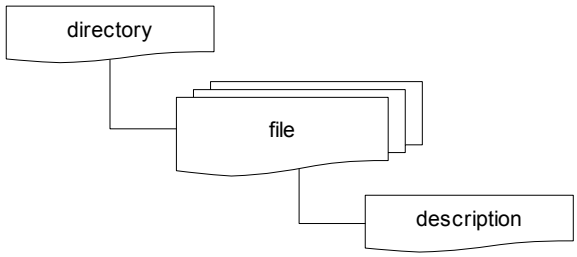

Figure 9: Configuration file list XML tree

7.1 Directory Element

Element	directory
Multiplicity	One
Description	Root node element for the directory and all the configuration file elements found in the directory.
Value	Unused
Attributes	
Name	**Value**
name	Directory name where configuration files are stored.

7.2 File Element

Element	file
Multiplicity	Zero or more
Description	Describes a configuration file
Value	Unused
Attributes	
Name	**Value**
name	The configuration file name. This includes the full file name including the XML extension.
type	The configuration file type. There are three types: • config - configuration file (e.g. telescope or instrument) • utility - utility application (e.g. power on/off biases) • data - data application (e.g. detector read out)

7.3 Description Element

Element	description

Multiplicity	One
Description	A description of the configuration.
Value	The description of what the configuration file does.