



nationaal archief

***Technical Description
of the Universal Virtual
Computer (UVC)***

***Data preservation
process for
spreadsheets***

The Digital Preservation Testbed was founded bij the Ministry of Education, Culture and Science and the Ministry of the Interior and Kingdom Relations. It was established in October 2000 to research different strategies to preserve digital documents over the long-term. Since July 2003 Testbed Digital Preservation is adopted by and continued within the Nationaal Archief of the Netherlands.

Nationaal Archief
Prins Willem-Alexanderhof 20
2595 BE Den Haag

Tel. +31 70 331 54 00

Fax: +31 70 331 55 40

Email testbed@nationaalarchief.nl
www.digitaleduurzaamheid.nl

Digital Preservation Testbed *Technical Description of the Universal Virtual Computer*
(version 1.0)

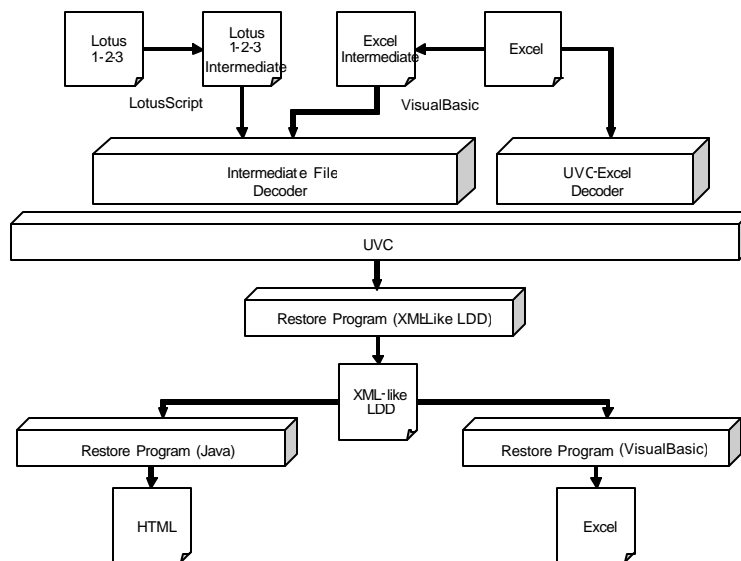
The Hague, January 2005

© Digital Preservation Testbed, The Hague 2005
All rights reserved. No part of this publication may be published or reproduced by printing, photocopying, microfilm or any other means without the prior permission of the programme office. The use of all or part of this publication to explain or support articles, books and theses and suchlike is permitted, provided that the source is clearly identified.

UVC technical specifications

Overview of UVC data preservation

The UVC data preservation process for spreadsheets is shown in the following diagram. An intermediate format is generated from Excel and Lotus spreadsheets that contains relevant information extracted from spreadsheet's original programming environment (via LotusScript or Visual Basic). The UVC data preservation approach is still in an experimental phase, and the development tools for the implementation of the data format decoder programs are still very basic: at present, the programs are still developed in assembly language! Ultimately it will be possible to implement all the logic for a data format decoder program that is written entirely in UVC code. A first step in this direction is the UVC Excel-format decoder program that directly implements a subset of the Logical Data Description (LDD) for Excel, and consequently without the use of the aforementioned intermediate file.



The Intermediate File Decoder and the UVC -Excel decoder generate the LDD for Lotus 1-2-3 and Excel spreadsheets. The LDD is comprised of a hierarchy of labelled elements in which the labels and their position in the hierarchy specify all the relevant information in the original spreadsheet. The following description specifies the top layers of the LDD spreadsheet hierarchy. The initial element is a spreadsheet (label 10), which indicates that the spreadsheet has a name (label 15) and is comprised of one or more worksheets (label 20), a set of labelled cells, and the metadata associated with the spreadsheet. The entire listing of the LDD used for spreadsheets is enclosed in Appendix C.

ELEMENT 10 [Spreadsheet] (15, 20+, 30+, 40?)

ELEMENT 15 [SpreadsheetName] (CHAR)

ELEMENT 20 [Worksheet] (50, 80+)

ELEMENT 30 [LabelledCell] (90, 50, 170)

ELEMENT 40 [Metadata] (350?, 360?)

The interface of the UVC passes the various labelled elements, in a standard format, on to the Restore program via the memory interface. Each element is comprised of three sections: the label code (tag), the length of the value string (in number of characters), and the value string.

TAG ELEMENT (16 bits == 2 bytes)	NUMBER OF CHARACTERS (32 bits == 4 bytes)	CHARACTERS (16 bits per char == 1 byte)
---	--	--

The tag and length of the string are represented by numbers in 16 and 32 bits. The value string is coded in Unicode UTF-16 format. The 'little endian' sequence is employed for all bytes; i.e. the most significant (or highest) byte of a number or UTF-16 character code is transmitted first, followed by the remaining bytes, in sequence.

A basic restore program then translates this data into an XML-like structure, whereby each label is assigned a beginning tag and a closing tag comparable to XML. The entire process is shown in the following example:

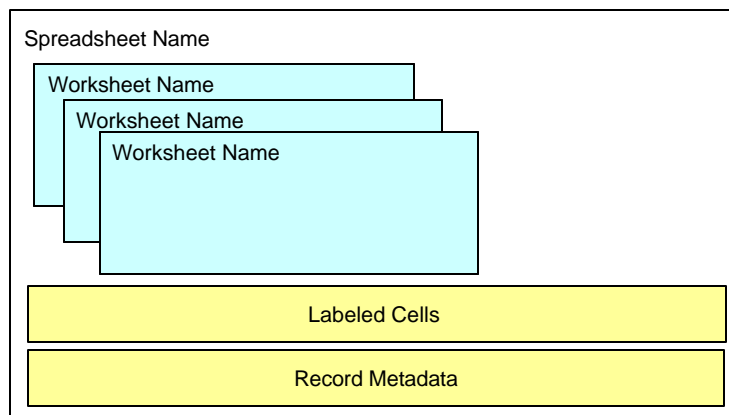
```
10 3 ABC
20 4 DEFG
```

which the basic restore program transforms into

```
<TAG1>ABC
    <TAG2>DEFG</TAG2>
</TAG1>
```

using the following LDD definitions:
ELEMENT 10 [TAG1] (20+)
ELEMENT 20 [TAG2]

The XML-like translation of the LDD contains, in schematic form, the element categories as shown in the following diagram. This XML-like representation of the spreadsheet is then used either for the conversion into Excel or for the representation of the spreadsheet data in an HTML browser.



The UVC data preservation approach is based on the principle that the LDD that is generated will be independent of future prevailing *de facto* technology standards. The only technology-dependent component is the UVC itself, which will need to be implemented on each platform. The UVC is the invariant that ensures that it will

always be possible to use the data format decoder program. On its own, the LDD is a sufficient interpretation of the relevant information from original format, independent of specific technology. Consequently the last steps (translation to XML-like Excel and HTML formats) are not essential for the durable preservation of digital records; self-evidently it will also be possible to convert to the *de facto* standard formats required for future prevailing platforms.

Interpretation of the Logical Data Description

It is essential that conversions to logical descriptions result in descriptions that can readily be interpreted in a unique manner. This is of vital importance to both UVC data preservation and XML-driven solutions employed in the long-term preservation of digital records. Often insufficient attention is devoted to the specific preconditions that need to be satisfied if an interpretation is to be successful.

The preconditions to be satisfied for the interpretation of a LDD for spreadsheets are identified below, on the basis of the above description of the entire UVC data preservation process.

Natural language

The relevant information of the format is displayed in natural language, independent of the technology, such that the comprehension of the natural language is as such a basic requirement. However, this is applicable to the provision of all information. The majority of LDDs are written in English.

Logical Data Schema

The Logical Data Schema (LDS) defines the correct construction of LDDs for spreadsheets. A number of options are available for the preservation of the LDS:

1. Make a printout of the LDS, and store the printout;
2. Use a generic LDD for all LDS specifications, as a result of which it will be necessary to preserve one unique LDS for all the other LDS specifications.

Without the LDS it will be impossible to interpret the LDD generated by the data format decoder program, since this returns solely numbers for the label codes/tags.

Unicode UTF-16

The format decoder returns all values associated with a label in Unicode (UTF-16) format. UTF-16 decoding is currently the world-wide standard for the electronic specification of characters. This standard has global support, and the specification is widely accessible. However from a long-term perspective nothing is eternal; consequently it will be necessary to ensure, should another character format become the *de facto* standard, that the UTF-16 specification is preserved and remains accessible.

Big versus Little Endian

The storage of a data element in a byte-oriented computer is always accompanied by the problem of the sequence to be employed for elements longer than one byte. For example, the decimal number 369 is represented in the form of 0000000101110001 when stated as a 16-bit binary number (2 bytes). Two options are available for the sequence of the bytes:

1. Big Endian
The bytes are transmitted from left to right, i.e. beginning with byte (00000001) and followed by (01110001).
2. Little Endian
The bytes are transmitted from right to left, i.e. beginning with byte (01110001) and followed by (00000001).

It will be self-evident that the incorrect interpretation of the sequence will yield a completely different result: what should have represented as 369 will, in the event that the bytes are exchanged, be interpreted as 28929 (0111000100000001). The UVC spreadsheet decoder uses little endian.

Layout References

References will also need to be incorporated in the LDS for any layout information that is included in the data to be preserved. In the instance of spreadsheets, the LDS also incorporates layout elements such as the font, cell borders, background colours, and patterns. Consequently these are present as specific elements in the LDD, such as Times-Roman font. However, these references are always specific to the application program, such as the lines round spreadsheet cells, or to the operating system, such as fonts. It is virtually certain that it will be impossible to use the current Excel program on the prevailing infrastructure in 2025, and consequently it will then be impossible to interpret layout references of this nature incorporated in the LDD. One possible way to avoid this problem would be to preserve images (bitmaps) of the current layout characteristics indexed by the possible values of the corresponding items of the LDS. This will ensure for the continued availability of an example of the appearance of a specific line pattern used in Excel 2003.

Functions in Formulae

Spreadsheets use formulae to calculate cell values, as well as standard computational functions enclosed with the application, such as accounting, mathematical and statistical functions. The LDD for spreadsheets also includes specifications of the formulae and the call procedure used for the functions incorporated in the program.

From the above summary it will be evident that the durable preservation of a digital object involves more than merely defining an LDS and implementing a specific UVC data format decoder program. It is also necessary to make a thorough review of the additional preconditions governing the interpretation of the LDD. The extent to which this will be necessary will depend on the estimation of the risks and the importance of the preserved objects. It is possible that many organisations will be of the opinion that, for example, their duties do not extend to the preservation of UTF-16 format definitions, and that it is unlikely that ICT specialists will be unable to interpret them at some point further in the future – a standpoint based on the wide support for Unicode. Although this can be a totally legitimate standpoint any decision of this nature must be taken after careful consideration of the situation – and not as a result of a lack of an insight into the relevant preconditions.