Table of Contents

1 Primitives

Type	*Size (byte)*	*Format*
int	4	big endian, signed
long	8	big endian, signed
double	8	IEE 754 encoding
string	variable	UTF8 encoding, starts with an int giving the string length in bytes
pointer	8	absolute position in file in bytes

2 File format

Type	*Description*
int	application specific magic number
int	archive version
pointer {Node}	position of the root node (always with list entries)
double	reserved
double	reserved
double	reserved
double	reserved

3 Structures

3.1 Structure {Node}

Type	*Description*
int	magic number of the node (0x00015555)
int	flags
pointer {Metadata}	position of node metadata
double	reserved
{NodeEntries}	entries of the node

3.2 Structure {NodeEntries}

Type	*Description*
int	magic number of the entries (0x00025555 or 0x00045555)
[{NodeEntriesAsList}]$^{0..1}$	list entries, if magic number is 0x00025555
[{NodeEntriesAsTable}]$^{0..1}$	table entries, if magic number is 0x00045555

3.3 Structure {NodeEntriesAsList}

Type	*Description*
int	total number of entries
int	number of child node entries

int	number of data entries
pointer {NodeEntriesAsList}	position of additional entries (0 for no additional entries)
[{NodeEntry}^{n}]	child entries (n = number of child node entries)
[{NodeEntry}^{n}]	empty space (n = total number of entries - number of child node entries - number of data entries)
[{NodeEntry}^{n}]	data entries (n = number of data entries)

3.4 Structure {NodeEntry}

Type	*Description*
pointer {Node}/{Data}	position of the child node or data
int	uid of the child node or data

3.5 Structure {NodeEntriesAsTable}

Type	*Description*
int	number of child node entries
int	number of data entries
[{NodeEntry}^{n}]	child entries (n = number of child node entries)
[{NodeEntry}^{n}]	data entries (n = number of data entries)

3.6 Structure {Data}

Type	*Description*
int	magic number of the data block 0x00105555
int	flags
long	total size of data in bytes
long	size of the main data block in bytes
pointer {DataAdditionalBlock}	position of the next additional data block
*	byte data

3.7 Structure {DataAdditionalBlock}

Type	*Description*
int	magic number of the additional data block 0x00205555
long	size of the additional data block in bytes
pointer {DataAdditionalBlock}	position of the next additional data block
*	byte data

3.8 Structure {Metadata}

Type	*Description*
{MetadataContent}	main metadata content

int	number of extended metadata contents (-1 for none)
[{MetadataContentExt}^{n}]	extended metadata contents

3.9 Structure {MetadataContent}

Type	*Description*
int	size of entries
[{MetadataContentEntry}]	

3.10 Structure {MetadataContentExt}

Type	*Description*
string	name of extention
{MetadataContent}	

3.11 Structure {MetadataContentEntry}

Type	*Description*
string	entry name
int	type of entry
[int]	size of data (optional, only if type of entry is -4)
[*]	data (size depends on the type of entry)

3.11.1 Types of data

Type of entry	*Type of data*	*Size of data (byte)*
-1	boolean	1
-2	long	8
-3	double	8
-4	raw data	[see size of data]
>=0	string	[see type of entry]