Newton Toolkit 1.6.x File Formats

This document describes the file formats used by NTK 1.6.x, both for Windows and Macintosh. This information is being provided to developers of utilities that need to access the information in these files, such as conversion utilities. Apple Computer, Inc., reserves the right to change the information in this document in the future without notice. While every effort has been made to make this document accurate, its accuracy is not guaranteed.

Please report any errors in this document to <mailto:ntkbugs@newton.apple.com>.

This document describes the formats of the following files:

- Project files
- Layout files
- Native code module files.

Preliminaries

All WinNTK files are stored in Newton Stream Object Format (NSOF). Such files can be read, for example, by FD_Unflatten of the 2.0 FDIL, or FDHydrateObject of the 1.0 FDIL.

MacNTK project files store project item information in the project file data fork, and preference information in the resource fork. Layout files contain layout information in NSOF in the data fork, and preference information in the resource fork. Native code module files are the same as that under WinNTK: all information is stored in NSOF in the data fork.

Binary objects containing embedded integers store them in Big Endian format. This is the same format as that used by 680x0-based Macintosh computers, PowerPC (as used in Power Macintosh computers), and ARM (as used in current eMate and MessagePad devices) processors. It is backwards from format the used by the Intel family of processors.

The following types are used in the definitions of MacNTK file formats.

```
typedef unsigned char Str255[256], Str63[64], Str32[33];

typedef unsigned char Boolean;

typedef struct VPoint {
      long  y, x;
} VPoint;

typedef struct VRect {
      long  top, left, bottom, right;
} VRect;

typedef struct GridInfo {
      long        scope;      // 4 bytes  Unused
      long        snap;       // 4 bytes  True if gridding active
      Boolean     show;       // 1 byte   True if gridding shown
      char        padding;    // 1 byte
      long        spacing;    // 4 bytes  Grid spacing
```


Copyright © 1997, Apple Computer, Inc.
All rights reserved
Subject to change

```
} GridInfo;
```

File Reference Frames

Project and layout files contain references to other files. Project files contain references to project items. Layout files contain references to linked subview files, user protos, and picture/resource files. These references are stored as frames, described here.

Windows

File references are stored in a frame containing information helpful in finding the file. In NTK 1.6 and 1.6.1, the frame has the following slots:

`class`	A symbol object identifying this frame as a `'fileReference`.
`deltaFromProject`	A string object containing the relative path from the project directory to the directory containing the file. This slot can also contain a string object holding a full path to the file.
`projectPath`	An optional string object containing the path of the project to which the file belongs. If present, this path would be used to find the file if there were no open project.

There are problems with the above approach. For instance, `'deltaFromProject` assumes that the path was relative to a project, which wasn't appropriate when one layout file contained a reference to another (in other words, there was a layout file to layout file relationship, not a project file to layout file relationship).

In NTK 1.6.2, the `'deltaFromProject` and `'projectPath` slots are still supported, but are deprecated in favor of the following additional slots added to the frame:

`relativePath`	An optional string object containing the relative path from the file containing the reference to the referenced file. This slot is not present if there is no relative path (for instance, the two files are on different volumes).
`fullPath`	A string object containing the full path to the referenced file. This string can be in UNC (Universal Naming Convention) or DOS (Disk Operating System) format.

Macintosh

Files references are merely binary objects with the class `'fsspec` and containing the contents of an alias handle. If a project is open at the time the file reference is created, the alias is created relative to the project file. Otherwise, it is not created relative to any file.

Older versions of NTK stored actual FSSpecs in the binary object instead of aliases. To distinguish between the two, NTK checks the size of the binary object before examining its contents. If the size of the object is equal to the size of an FSSpec, then the contents are treated as an FSSpec. Otherwise, they're treated as an alias.

If the alias cannot be resolved into an FSSpec, and a project is open, NTK looks in the project for an item with the same name as that stored in the alias. If one is found, that project item is assumed to be the one referred to by the

Copyright © 1997, Apple Computer, Inc.
All rights reserved

alias. If no project item is found with the same name, NTK creates an FSSpec for a file with the same name as the one referred to in the alias, and that exists in the same directory as the project.

Picture Reference Frames

Windows

In NTK 1.6, pictures (bitmap files) are referenced via a frame with the following slots:

`__ntExternalFile`	A file reference to a file containing an image to be converted into a 1-bit bitmap.
`__ntMaskFile`	An optional file reference to a mask image. Unused.
`__ntCreateMask`	An integer object indicating whether or not the generated Newton bitmap frame contains a `'mask` slot. If this slot exists, and it contains a non-zero value, then a `'mask` slot will be generated, either from the “complementary image” (a file with the same name as that referenced by `'__ntExternalFile`, suffixed with a “!”) or by calculating a mask from the `'__ntExternalFile` image. If the slot does not exist, or it doesn’t contain an integer object, or the integer object is zero, then no mask is generated for the Newton bitmap frame.

With the addition of “picture families” (a collection of related bitmap files), the above slots are deprecated (but are maintained for backward compatibility) in favor of the following slots:

`imageInfo1`	An optional file reference to a file containing an image to be converted into a 1-bit bitmap.
`imageInfo2`	An optional file reference to a file containing an image to be converted into a 2-bit bitmap.
`imageInfo4`	An optional file reference to a file containing an image to be converted into a 4-bit bitmap.
`imageInfo8`	An optional file reference to a file containing an image to be converted into an 8-bit bitmap.
`maskInfo`	An optional file reference to a file containing an image to be converted into a 1-bit masking bitmap. Must be present if `'maskOption` is 1 or 2.
`maskOption`	An integer object with one of the following values: 0 no mask 1 use specified image as mask 2 XOR specified image to produce mask 3 calculate a mask

Macintosh

Picture references created by NTK 1.6.3 and earlier are frames containing the following slots:

Copyright © 1997, Apple Computer, Inc.
All rights reserved

`__ntExternalFile` A file reference to a file containing a `'PICT'` resource to be converted into a 1-bit bitmap.

`__ntResID` An integer object with the resource ID of the `'PICT'` resource.

`__ntPictName` An optional string object containing the name of the `'PICT'` resource. This slot is present only if the `'PICT'` resource has a name. Yes, this information is redundant with `'__ntResID`.

`__ntCreateMask` An integer object indicating whether or not the generated Newton bitmap frame contains a `'mask` slot. If this slot exists, and it contains a non-zero value, then a `'mask` slot will be generated, either from the “complementary image” (a named resource with the same name as that referenced by `'__ntPictName`, suffixed with a “!”) or by calculating a mask directly from the `'PICT'`. If the slot does not exist, or it doesn’t contain an integer object, or the integer object is zero, then no mask is generated for the Newton bitmap frame.

With the addition of “picture families” (a collection of related bitmap files), the above slots are deprecated, but are maintained for backward compatibility. A new slot called `'vDesc` has been added. This slot contains a frame with the following slots:

`class` A symbol object containing `'picture`.

`images` An array object with each element corresponding to one of the possible picture bit depths. The first element corresponds to the the 1-bit image, the second element corresponds to the 2-bit image, the third element corresponds to the 4-bit image, and the fourth element corresponds to the 8-bit image. Each image specification is a frame with the following slots:

`resource`	An integer object containing the resource ID of the `'PICT'`.
`fileSpec`	A Macintosh file reference object of the resource file containing the `'PICT'`.

`mask` A frame object containing `'resource` and `'filespec` slots for the mask to be used for this picture family.

`maskStrategy` An integer object with one of the following values:

0	no mask
1	use specified image as mask
2	XOR specified image to produce mask
3	calculate a mask

Project File Format

Windows

Project files are stored in Newton Streamed Object Format (NSOF). Such files can be read, for example, by

Copyright © 1997, Apple Computer, Inc.
All rights reserved

FD_Unflatten of the 2.0 FDIL, or FDHydrateObject of the 1.0 FDIL.

The root level object is a frame containing the following slots:

`ntkPlatform` An integer object with one of the following values:

0	Macintosh platform
1	Windows platform

`fileVersion` An integer object indicating the file version number. The current version number is 2. This number is informational only; NTK currently doesn't do anything based on the version number.

`projectItems` A frame containing the project items. This frame is described in the section "Project Items Frame".

`projectSettings` A frame containing the following slots:

- `platform` A string object containing the platform file for this project. The string does not include the file extension or file path. It corresponds to the "Platform" drop-down menu.
- `language` A string object containing the specified language. It corresponds to the "Language" edit text item of the Package panel of the Settings dialog.
- `debugBuild` A boolean object indicating whether or not the package will be built with debugging features. It corresponds to the "Compile for Debugging" check box of the Package panel of the Settings dialog.
- `ignoreNative` A boolean object indicating how the NewtonScript "`native`" keyword should be handled. It corresponds to the "Ignore Native Keyword" check box of the Package panel of the Settings dialog.
- `checkGlobalFunctions` A boolean object specifying whether or not global functions should be checked against a list of known global functions during compile time. It corresponds to the "Check Global Function Calls" check box of the Package panel of the Settings dialog.
- `oldBuildRules` A boolean object specifying compatibility mode for projects created by Macintosh NTK 1.0. It corresponds to the "NTK 1.0 Build Rules" check box of the Package panel of the Settings dialog.
- `useStepChildren` A boolean object specifying how child views should be handled. It corresponds to the "Use stepChildren Slot" check box of the Package panel of the Settings dialog.
- `suppressByteCodes` A boolean object controlling code generation. It corresponds to the "Suppress Byte Code" check box of the Package panel of the Settings dialog.

Copyright © 1997, Apple Computer, Inc.
All rights reserved

`fasterFunctions` A boolean object controlling code generation. It corresponds to the “Faster Functions (2.0 Only)” check box of the Package panel of the Settings dialog.

`outputSettings` A frame containing the following slots:

`applicationName` A string object containing the name of the application in the built package. It corresponds to the “Name” edit text item of the Application panel of the Settings dialog.

`applicationSymbol` A string object containing the symbol of the application in the built package. It corresponds to the “Symbol” edit text item of the Application panel of the Settings dialog.

`partType` An integer object indicating the kind of package part this project is creating. Corresponding to the radio buttons on the Output panel of the Settings dialog, it can be one of the following values:

0 Application part
1 Book part
2 Auto part
3 Store part
4 Stream part
5 Custom part

`iconFile` A file reference containing the name of the project item holding the application icon. In NTK 1.6.1, this slot is deprecated in favor of `'iconProNormal` and `'iconProHighlighted`.

`iconProNormal` A picture reference frame containing the application icons to be used at different screen depths. This slot was added in NTK 1.6.1.

`iconProHighlighted` A picture reference frame containing the highlighted application icons to be used at different screen depths. This slot was added in NTK 1.6.1.

`topFrameExpression` A string object containing the expression typed into the “Result” edit text item of the Output panel of the Settings dialog.

`autoClose` A boolean object specifying whether or not the application should close when another “auto close” application is launched. It corresponds to the “Auto Close” check box of the Application panel of the Settings dialog.

`customPartType` A string object containing the text entered into the edit text item next to the “Custom Part” radio button on the Output panel of the Settings dialog. The string must be four characters long.

`fasterSoups` A boolean object controlling code generation. It corresponds to the “New-Style Stores (Newton OS 2.0 Only)” check box of the Output panel of the Settings dialog.

Copyright © 1997, Apple Computer, Inc.
All rights reserved

`packageSettings` A frame containing the following slots:

`packageName`	A string object containing the package name. It corresponds to the “Name” edit text item of the Package panel of the Settings dialog.
`version`	A string object containing the version typed into the “Version” edit text item of the Package panel of the Settings dialog. Note that while this is a string object, the text it contains must be convertible to an integer between 0 and 9999.
`copyright`	A string object containing the version typed into the “Copyright” edit text item of the Package panel of the Settings dialog.
`optimizeSpeed`	A boolean object controlling code generation. It corresponds to the “Use Compression” check box of the Package panel of the Settings dialog. Note that the boolean value stored is the opposite of setting of the check box.
`copyProtected`	A boolean object controlling package generation. It corresponds to the “Copy Protected” check box of the Package panel of the Settings dialog.
`deleteOnDownload`	A boolean object controlling package downloading. It corresponds to the “Delete Old Package on Download” check box of the Package panel of the Settings dialog.
`dispatchOnly`	A boolean object controlling package generation. It corresponds to the “Auto Remove Package” check box of the Package panel of the Settings dialog.
`newton20only`	A boolean object controlling code generation. It corresponds to the “Newton OS 2.0 Only” check box of the Project panel of the Settings dialog.
`fourByteAlignment`	A boolean object controlling package generation. It corresponds to the “Tighter Object Packing” check box of the Project panel of the Settings dialog.
`zippyCompression`	A boolean object controlling package generation. It corresponds to the “Faster Compression” check box of the Package panel of the Settings dialog.

`profilerSettings` A frame containing the following slots:

`memory`	An integer object controlling profiling. Currently hard-coded to 4K, it can’t be changed by the user.
`percent`	An integer object controlling profiling. Currently set to 4 (indicating 100%), but unused.
`compileForProfiling`	A boolean object controlling code generation. It corresponds to the “Compile for Profiling” check box of the Project panel of the Settings

Copyright © 1997, Apple Computer, Inc.
All rights reserved

dialog.

`compileForSpeed` A boolean object controlling code generation. It corresponds to the "Profile Native Functions" check box of the Project panel of the Settings dialog. Note that the boolean value stored is the opposite of setting of the check box. Note also that WinNTK 1.6 and 1.6.1 don't note the previous note, and treat the value backwards.

`detailedSystemCalls` A boolean object controlling profiling. Currently hard-coded to FALSE, and can't be changed by the user.

`detailedUserFunctions` A boolean object controlling profiling. Currently hard-coded to TRUE, and can't be changed by the user.

`windowRect` A frame containing `'top`, `'left`, `'bottom`, and `'right` slots, each containing an integer indicating the coordinates for the project. In NTK 1.6 and 1.6.1, the values correspond to the outer boundaries of the window. In NTK 1.6.2 and later, the values correspond to the inner boundaries of the window.

Project Items Frame

The `'projectItems` frame contains the following slots:

`sortOrder` An integer object with one of the following values:

0	Sort by build/sequence order
1	Sort by file name
2	Sort by file type
3	Sort by file size
4	Sort by file modification date
5	Sort by full file path name

`items` An array object containing the actual project items. This array always contains the items in build order. Each element of the array is a frame with the following format:

`file` A reference to the project item's file.

`type` An integer object indicating the project item's type. It can be one of the following values:

0	Layout file (also used for user-proto and print layout files)
1	Bitmap file
2	Metafile file (unused)
3	Sound file
4	Book file (deprecated in favor of script items)
5	Script file (NewtonScript source file)
6	Package file
7	Stream file
8	Native C++ code module file

Copyright © 1997, Apple Computer, Inc.
All rights reserved

`isMainLayout`	A Boolean object indicating if the given project item is considered the main layout of an application project. This slot is optional; it is present only if it would contain a non-NIL value. Only one project item should be marked as the main layout.

Macintosh

On the Macintosh, the data fork contains the list of project items, while the resource fork contains various preference settings.

The data fork contains the following:

`4 bytes`	Format version number. Currently set to 103, which this document describes.
`2 bytes`	File count. Number of project item aliases in this file.
`4 bytes`	Sort criteria. An integer indicating the following: 1 Sort by build/sequence order 2 Sort by file name 3 Sort by file type 4 Sort by file size 5 Sort by file modification date 6 Sort by full file path name

If the project contains one or more project items, the following appear next in the project file:

`n bytes`	A sequence of “File Count” project item references. Each reference begins with a 4-byte integer containing the length of an alias, followed by the alias itself. The project item references are stored in build order, not according to the user’s sort criteria.
`2 bytes`	Main layout number. A 1-based index of the project item designated as the main layout. This value will be zero if there is no main layout. Note that the index is valid only when the list of project items is sorted according to the user’s sort criteria, not according to the order in which the project item references are stored on disk (which is always in build order).

The resource fork contains the following resources:

`'PJPF'(9999)` Project Preferences

`Str32`	`33 bytes`	ApplicationName: A Pascal string containing the name of the application in the built package. It corresponds to the “Name” edit text item of the Output Settings panel of the Project Settings dialog.
	`1 byte`	padding
`Str32`	`33 bytes`	IconName: A Pascal string containing the name of the `'PICT'` resource used for the application’s icon.

Copyright © 1997, Apple Computer, Inc.
All rights reserved

	1 byte	padding
Str32	33 bytes	Platform: A Pascal string containing the platform name.
	1 byte	padding
Str32	33 bytes	PackageName: A Pascal string containing the name of the package. It corresponds to the “Name” edit text item of the Package Settings panel of the Project Settings dialog.
	1 byte	padding
Str32	33 bytes	ApplicationSymbol: A Pascal string containing the symbol of the application in the built package. It corresponds to the “Symbol” edit text item of the Output Settings panel of the Project Settings dialog.
	1 byte	padding
Str32	33 bytes	Version: A Pascal string containing the version typed into the “Version” edit text item of the Package Settings panel of the Project Settings dialog. This string must be convertible to an integer between 0 and 9999.
	1 byte	padding
Str63	64 bytes	Copyright: A Pascal string containing the version typed into the “Copyright” edit text item of the Package Settings panel of the Project Settings dialog
Boolean	1 byte	OptimizeSpeed: A boolean controlling code generation. It corresponds to the “Use Compression” check box of the Package Settings panel of the Project Settings dialog. Note that the boolean value stored is the opposite of setting of the check box.
Boolean	1 byte	CopyProtected: A boolean controlling package generation. It corresponds to the “Copy Protected” check box of the Package Settings panel of the Project Settings dialog.
Boolean	1 byte	DeleteOnDownload: A boolean controlling package downloading. It corresponds to the “Delete Old Package on Download” check box of the Package Settings panel of the Project Settings dialog.
Boolean	1 byte	DebugBuild: A boolean indicating whether or not the package will be built with debugging features. It corresponds to the “Compile for Debugging” check box of the Project Settings panel of the Project Settings dialog.
Boolean	1 byte	AutoClose: A boolean specifying whether or not the application should close when another “auto close” application is launched. It corresponds to the “Auto Close” check box of the Output Settings panel of the Project Settings dialog.
	1 byte	padding
Str63	64 bytes	IconFile: A Pascal string containing the name of the file containing the `'PICT'` resource used for the application’s icon.

Copyright © 1997, Apple Computer, Inc.
Subject to change *All rights reserved*

`Boolean`	`1 byte`	CustomPart: A boolean indicating that the “Custom Part” radio button of the Output Settings panel of the Project Settings dialog is set.
	`1 byte`	padding
`OSType`	`4 bytes`	PartType: A value indicating the type of package built. It can be one of the following values: `'form'` Application `'book'` Book `'auto'` Auto Part `'soup'` Store Part It can also be whatever the user specified in the edit text item following the Custom Part radio button of the Output Settings panel of the Project Settings dialog.
`Str255`	`256 bytes`	TopFrameExpression: A Pascal string containing the expression typed into the “Result” edit text item of the Output Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	MakeStream: A boolean indicating that the “Stream File” radio button of the Output Settings panel of the Project Settings dialog is set.
`Boolean`	`1 byte`	DispatchOnly: A boolean controlling package generation. It corresponds to the “Auto Remove Package” check box of the Package Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	Newton20Only: A boolean controlling code generation. It corresponds to the “Newton 2.0 Platform Only” check box of the Project Settings panel of the Project Settings dialog.
	`1 byte`	padding
`Boolean`	`1 byte`	CompileForProfiling: A boolean controlling code generation. It corresponds to the “Compile for Profiling” check box of the Project Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	CompileForSpeed: A boolean controlling code generation. It corresponds to the “Profile Native Functions” check box of the Project Settings panel of the Project Settings dialog. Note that the boolean value stored is the opposite of setting of the check box.
`Boolean`	`1 byte`	DetailedSystemCalls: A boolean controlling profiling. Currently hard-coded to FALSE, and can’t be changed by the user.
	`1 byte`	padding
`short`	`2 bytes`	Memory: An integer controlling profiling. Currently hard-coded to 4K, it can’t be changed by the user.
`Byte`	`1 byte`	Percent: An integer controlling profiling. Currently set to 4 (indicating 100%), but unused.

Copyright © 1997, Apple Computer, Inc.
All rights reserved
Subject to change

`Boolean`	`1 byte`	DetailedUserFunctions: A boolean controlling profiling. Currently hard-coded to TRUE, and can't be changed by the user.
`Str63`	`64 bytes`	Language: A Pascal string containing the specified language. It corresponds to the "Language" edit text item of the Project Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	IgnoreNative: A boolean indicating how the NewtonScript "`native`" keyword should be handled. It corresponds to the "Ignore Native Keyword" check box of the Project Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	CheckGlobalFunctions: A boolean specifying whether or not global functions should be checked against a list of known global functions during compile time. It corresponds to the "Check Global Function Calls" check box of the Project Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	OldBuildRules: A boolean specifying compatibility mode for projects created by Macintosh NTK 1.0. It corresponds to the "NTK 1.0 Build Rules" check box of the Project Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	UseStepChildren: A boolean specifying how child views should be handled. It corresponds to the "Use stepChildren Slot" check box of the Project Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	SuppressByteCodes: A boolean controlling code generation. It corresponds to the "Suppress Byte Code" check box of the Project Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	FasterFunctions: A boolean controlling code generation. It corresponds to the "Faster Functions (2.0 Only)" check box of the Project Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	FasterSoups: A boolean controlling code generation. It corresponds to the "New-Style Stores (2.0 Only)" check box of the Output Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	FourByteAlignment: A boolean controlling package generation. It corresponds to the "Tighter Object Packing (2.0 Only)" check box of the Project Settings panel of the Project Settings dialog.
`Boolean`	`1 byte`	ZippyCompression: A boolean controlling package generation. It corresponds to the "Faster Compression (2.0 Only)" check box of the Package Settings panel of the Project Settings dialog.
	`1 byte`	padding

`'PJPF'(10000)` Newton Application Icon Preferences

In NTK 1.6.4, support for "picture families" was added, allowing developers to specify a set of related images to appear as the application's icon in the Extras drawer on the Newton OS device.

Copyright © 1997, Apple Computer, Inc.
All rights reserved

This resource contains a flattened NewtonScript array object in NSOF. The first element of the array contains a picture reference frame for the normal icon images. The second element of the array contains a picture reference frame for the highlighted icon images.

The `IconName` and `IconFile` fields of the `'PJPF'(9999)` resource are maintained for backward compatibility.

`'PJST'(9999)` Window Settings

`VRect` `16 bytes` Window location/size. Bounding box of content region in global coordinates.

Layout File Format

Windows

`layoutSettings` A frame containing the following slots:

- `ntkPlatform` An integer object containing one of the following values:
  - 0 Macintosh platform
  - 1 Windows platform
- `fileVersion` An integer object indicating the file version number. The current version number is 2. This number is informational only; NTK currently doesn't do anything based on the version number.
- `windowRect` A rectangle frame containing the coordinates of the layout window.
- `layoutName` Name of the layout. Optional and unused.
- `layoutType` An integer object containing one of the following values:
  - 0 Normal layout
  - 1 Print format
  - 2 User proto
- `layoutSize` A point frame containing the size of the layout.
- `gridSize` A point frame containing the grid size of the layout window.
- `gridState` A boolean object specifying whether or not gridding is shown.
- `gridSnap` A boolean object specifying whether or not gridding is active
- `linkedTo` An optional string or file reference object. Unused.

`templateHierarchy` The root template of all the templates in this layout file. Templates are frames, the contents of which are described the section "View Template Format".

Copyright © 1997, Apple Computer, Inc.
Subject to change *All rights reserved*

Macintosh

The data fork contains the root template of all the templates in this layout file. Templates are frames, the contents of which are described in the section “View Template Format”.

The resource fork contains the following resources:

`'FMST'(9999)` Project Preferences

`VRect`	`16 bytes`	Layout window position. Bounding box of content region in global coordinates.
`Str63`	`64 bytes`	Obsolete.
`VPoint`	`8 bytes`	Layout size.
`short`	`2 bytes`	File version (currently 7).
`Boolean`	`1 byte`	A boolean indicating whether or not this layout is linked to a linkedSubview.
	`1 byte`	padding.
`Str255`	`256 bytes`	If this layout is linked to a linkedSubview, contains the name of the layout containing the linkedSubview. Unused.
`GridInfo[2]`	`28 bytes`	Two GridInfo structs containing information about the layout window. The first GridInfo contains information pertaining to vertical attributes of the layout window; the second GridInfo does likewise for the horizontal attributes. Note that both `show` fields are always both TRUE or both FALSE.

View Template Format

The “view” objects that users create and manipulate in the NTK browser and layout window are actually called “view templates”. They aren’t views themselves, but they play one on TV. That is, they tell the Newton OS how to create the actual view at runtime.

The view template is what NTK produces when building a package. The information used to create a view template is stored in the layout file as a “view template descriptor”. View template descriptors are frames containing the following slots:

`value`	A frame containing slots that will be added to the view after it is compiled. Each slot contains a frame called a “slot descriptor”, described later.
`__ntDeclare`	If the view is declared, this slot contains a reference to the view template descriptor to which it is declared.
`__ntExternFile`	If the view is a linked subview, this slot contains a file reference to the linked-to layout.
`__ntID`	A symbol indicating what type of view this is. The symbol stored here can be one of

Copyright © 1997, Apple Computer, Inc.
Subject to change *All rights reserved*

the ones you see in the popup menu containing Newton OS proto names (e.g., `'protoApp`, `'clView`), or the special symbols `'userProto` or `'linkedSubview`.

`__ntName` — If the view has a name, this slot contains a string object holding the name.

`__ntObjectPointer` — (Windows only) Pointer to C++ object. This slot is valid only during runtime; it has no significance when the view is stored on disk.

`__ntParent` — (Windows only) Reference to parent view template descriptor. This slot is valid only during runtime; it has no significance when the view is stored on disk.

The `'value` slot is a frame containing the slots that will populate the view after it is compiled. The name of each slot is the name that will appear in the final view, but the content of the slot—rather than being the value specified in NTK's browser—is a frame called a "slot descriptor". Each slot descriptor has the following slots:

`value` — The real value, as specified by the user in NTK's browser.

`__ntDataType` — A binary object containing a 4-byte integer indicating the type of data stored in the 'value slot. The 4-byte integer can be one of the following:

`'EVAL'`	Evaluate slot
`'NUMB'`	Number
`'INTG'`	Integer (same as Number)
`'REAL'`	Floating point number
`'BOOL'`	Boolean
`'RECT'`	Rectangle
`'TEXT'`	Text/string slot
`'FONT'`	Font (unused)
`'SCPT'`	Script slot
`'ARAY'`	Array (only used internally for stepChildren slot)
`'PICT'`	Picture

`__ntFlags` — An integer object comprising a set of bit-flags describing slot access. The bits are defined as follows:

2	If set, this slot needs to be overridden.
3	If set, this slot can't be edited.
4	If set, this slot can't be deleted.
5	If set, this slot is being edited.
6	If set, this slot doesn't appear in the browser.

`__ntEffect` — If the slot descriptor corresponds to the viewEffect slot, this slot contains an integer object holding the menu item index of the view effect menu. This is different from what's store in the `'value` slot, which contains the actual view effect bits used by the Newton OS view system.

`__ntStatusInfo` — (Windows only) Frame indicating slot status. This slot is valid only during runtime;

Copyright © 1997, Apple Computer, Inc.
All rights reserved

it has no significance when the view is stored on disk.

`state` (Windows only) Edit status. This slot is valid only during runtime; it has no significance when the view is stored on disk.

`by` (Windows only) Object editing this slot. This slot is valid only during runtime; it has no significance when the view is stored on disk.

The view template descriptor's `'value` slot not only contains the user-defined slots; it also contains the following special slots:

`stepChildren` This slot is merely a slot descriptor for an array, where each element is the view template descriptor for a child view template. Thus, stepChildren's value slot is the array of children, `'__ntDataType` is `'ARAY'`, and `'__ntFlags` is 64, indicating that the slot should not be displayed.

`__ntTemplate` This slot descriptor corresponds to the `'_proto` or `'viewClass` slot of the view template. It determines if a `'_proto` or `'viewClass` slot gets added to the view template, and what the contents of the slot are. It contains the following slots:

- `value` If this view template is based on a Newton OS proto (e.g., `'protoApp`), this slot contains an integer object representing the proto's magic pointer value. If this view is based on a primitive view (e.g., `'clView`), this slot contain an integer object representing to the view's class ID. If this view is based on a user proto, this slot contains a reference to the corresponding file.
- `__ntDataType` Classifies this view template into one of four types. This slot holds a binary object containing one of the following 4-byte integers:

`'PROT'`	View is based on Newton OS proto.
`'CLAS'`	View is based on primitive view.
`'USER'`	View is based on user proto.
`'LINK'`	View is linked to another view.

- `__ntFlags` An integer object comprising a set of bit-flags describing view access. The bits are defined as follows:

1	If set, can't add slots to this view.
3	If set, this view can't be edited.

`beforeScript` If present, this is a slot descriptor for a script value containing a script to be run before the view has been constructed. The final view template does not contain a `'beforeScript` slot.

`afterScript` If present, this is a slot descriptor for a script value containing a script to be run after the view has been constructed. The final view template does not contain an `'afterScript` slot.

Copyright © 1997, Apple Computer, Inc.
All rights reserved

Native Code Module Format

Native code modules are files created by the Newton C++ Tools for inclusion in NTK projects. These files contain data in NSOF, and are the same on both platforms. The object in the file is a frame with the following slots:

`class` — A symbol object containing `'NativeModule`.

`version` — An integer object with the version of this frame. The initial value of 1 implies that at a minimum, the slots listed in this frame will always be present in this and future versions of the module frame.

`CPUType` — A symbol object containing `'ARM610`.

`name` — A symbol object identifying the name of the module, as specified by the first entry in the .exp file.

`code` — A binary object of class 'code consisting of one or more Asm/C/C++ functions linked together as a single Read-Only Code Area (as created by the ARM Ltd. C++ compiler). The code is linked with a base address zero. Describing the contents of this binary object is currently beyond the scope of this document.

`relocations` — A binary object of class `'relocs` containing the relocation information. The first 4 bytes of the binary object contain the number of relocation entries in the rest of the binary object. The rest of the binary object is composed of an array of 4 byte offsets into the `'code` block where relocation needs to occur. These offsets are always sorted in ascending order.

`debugFile` — A string object containing the name of the file containing the debugging information.

`entryPoints` — An array object containing entry point information. Each element in the array is a frame containing the following slots:

- `name` — A symbol object containing the name of the entry point.
- `offset` — An integer object with the offset (in bytes) into the code block of the first instruction of the entry point.
- `numArgs` — An integer object with the number of arguments the entry point takes.

Copyright © 1997, Apple Computer, Inc.
Subject to change *All rights reserved*