Configuration storage

Introduction and Overview

Application and system configuration is held in a group of central repositories. Applications may be shared between a number of different users, and therefore their configurations should not be locked to that application. As a result of this, it is possible to have a hierarchy of configuration details which allows (in the most common case) the application to use system defaults, or user specific settings.

Technical Details

Two system variables are used in the configuration of applications - *Choices$Path (on page 7)* and *Choices$Write (on page 6)*. Applications should not attempt to process `Choices$Path` directly, but should use it only in via the filename `Choices:`.

Applications must read from `Choices:*` and write to `Choices$Write.*`. Reading and writing should not be performed simultaneously, as these entries refer to the same file. Applications should never read from `Choices$Write.*`, or write to `Choices:*`.

Applications may assume that `Choices$Write` exists and is writable. No other part of the `Choices$Write` or `Choices:` may be assumed to exist.

Naming conventions

The generic form of naming of files within the Choices structure is :

<application>.<file>

Any number of files may be stored within the application specific directory of the Choices structure, but authors should be aware that the filing system on which the choices are stored may be limited to 77 files per directory, or 10 character filenames as determined by the file system itself. Whilst authors should not worry unduly about this, it should be a factor in naming such files.

Whilst it is possible to store files without using a sub-directory - ie, just storing a file in the hierarchy with the same name as that of the application, this is not recommended. Such use precludes the additional of other configuration files for that application, should future versions require them.

Grouping by author

An alternative form of naming files which may be used is :

<author>.<application>.<file>

In this case, the *<author>* should be the well known name of the group issuing the product, such as `MegaCorp`, or `JBloggs`. As with the files themselves, the filesystems constraints on naming should be considered when selecting such names.

Grouping by category

A more wide spread and preferred form of naming files which may be used is that of grouping by the category into which the application falls :

`<category>.<application>.<file>`

The *<category>* name should be chosen to avoid clashes between it and application names.

Categories currently known to be in use - this is in no way an exhaustive list - include :

Category	Contents
!ZapUser	Zap modules and resources for the user live here.
Boot	Resources relating to the booting of the machine live here.
Audio	Audio applications should store their details here.
IRC	IRC related applications should store their details here.
ScrSavers	RISC OS 4-style screen savers should store their details here.
USB	USB devices should store their details here.
WWW	World Wide Web related applications should store their details here.

Authors wishing to create new categories should consider whether this is in fact the best course of action. Consultation with other developers working on similar projects is recommended.

Name registration

When you register an application name (and have had it confirmed), this also includes an allocation in the Choices hierarchy of the same name. This should be borne in mind when naming applications.

If a new category name is to be used, it should also be registered, as should author name groups.

Configuration file contents

Whilst it is recommended that configuration files contain textual configuration data, it is entirely the choice of the application what data it should store.

Robustness

No application should ever crash because of a poorly formatted configuration file. Loading an entire file into a structure (in C or Assembler) or fixed length buffer is likely to cause problems if the file is corrupt in any way. Reading textual strings in BASIC should be checked for overly long strings. Non-existent tags (in tagged files) should be given default values. It is recommended that if a corrupt configuration file is identified it be ignored and the defaults used.

When Choices should be written

In general, applications should not write choices unless the user explicitly requests this. This is not practical in some situations, particularly where the choices require a reload to take effect. Care should be taken to ensure as far as is possible, that the configuration presented to the user is preserved fully within the files written.

In particular applications should not write choices when they quit. Doing so may cause invalid choices to be written, or (in the case of defaulted choices) overwrite the user selected choices. The only exception to this is where the user has chosen to write choices on application quit.

RISC OS 3.1 and earlier

Previous versions of this document dealt with support for systems where `Choices$*` were unset. Such system are should no longer be supported. It is strongly recommended that applications fault the lack of `Choices$Write` in their `!Run` file.

An application, `!Choices` has been provided for systems without a suitable boot sequence, should this be necessary. In general, support for such systems should be discontinued.

System variables

Choices$Dir

Obsolete description of the Choices hierarchy

Use

This variable contains a reference to the highest level (usually the users) configuration directory. It is now obsolete and should not be used by any applications.

Related system variables

Choices$Write (on page 6)
Choices$Path (on page 7)

Choices$Write

Directory into which application configuration details should be written

Use

This variable contains the name of the directory into which configuration details should be written. It is used to write out user-specific configuration details, which will override any others which may have been given in *Choices$Path (on page 7)*.

Applications should never use the directory given in this variable for any form of read operation. The directory specified may be assumed to exist and be writable.

Related system variables

Choices$Path (on page 7)

Choices$Path

Comma separated list of paths to scan for configuration details

Use

This variable contains a comma separated list of paths which will be searched to locate configuration details. This list will be searched in order from beginning to end until a match is found. It is usual for the first entry in `Choices$Path` to be `Choices$Write.`, which ensures that the user configuration is searched after any other system, or administrator set configuration. Applications should always access the choices hierarchy via `Choices:` rather than attempting to expand this variable themselves.

This variable should never be used directly by applications. The path `Choices:` should never be used for write operations.

Related system variables

Choices$Write (on page 6)

Document information

Maintainer(s): Justin Fletcher <gerph@innocent.com>

History:

Revision	Date	Author	Changes
0.00	04 Jun 1998	JRF	● Released to various people via IRC and email for comments. ● Sent to Dave Walker for confirmation that I'm not talking rubbish.
1.00	29 Jun 1998	JRF	● Suggestions, comments and complaints added to the document. ● Reformatted into sections.
1.01	08 Jan 1999	JRF	● Reformatted as HTML for the 'info' section of my website.
2.00	09 Jan 2002	JRF	● Re-written from scratch, using information in the original. ● Support for RISC OS 3.1 is no longer given and is strongly discouraged. ● We no longer provide examples; applications programmers capable of using files should be able to construct their own code. The removal of support for RISC OS 3.1 makes such examples redundant anyhow. ● This documentation is based on the PRMinXML structures, rather than HSC.

Disclaimer: This document is based on my own experiences as a developer and is not an officially sanctioned document. It describes what I believe to be the Best Common Practices for applications using Choices.