37.1. `ic` — Access to the Mac OS X Internet Config¶

Platforms: Mac

This module provides access to various internet-related preferences set through **System Preferences** or the **Finder**.

Note

This module has been removed in Python 3.x.

There is a low-level companion module `icglue` which provides the basic Internet Config access functionality. This low-level module is not documented, but the docstrings of the routines document the parameters and the routine names are the same as for the Pascal or C API to Internet Config, so the standard IC programmers' documentation can be used if this module is needed.

The `ic` module defines the `error` exception and symbolic names for all error codes Internet Config can produce; see the source for details.

exception `ic.error`¶
Exception raised on errors in the `ic` module.

The `ic` module defines the following class and function:

class `ic.IC`([*signature*[, *ic*]])¶
Create an Internet Config object. The signature is a 4-character creator code of the current application (default `'Pyth'`) which may influence some of ICs settings. The optional *ic* argument is a low-level `icglue.icinstance` created beforehand, this may be useful if you want to get preferences from a different config file, etc.

`ic.launchurl`(*url*[, *hint*])¶
`ic.parseurl`(*data*[, *start*[, *end*[, *hint*]]])¶
`ic.mapfile`(*file*)¶
`ic.maptypecreator`(*type*, *creator*[, *filename*])¶
`ic.settypecreator`(*file*)¶
These functions are "shortcuts" to the methods of the same name, described below.

37.1.1. IC Objects¶

`IC` objects have a mapping interface, hence to obtain the mail address you simply get `ic['MailAddress']`. Assignment also works, and changes the option in the configuration file.

The module knows about various datatypes, and converts the internal IC representation to a "logical" Python data structure. Running the `ic` module standalone will run a test program that lists all keys and values in your IC database, this will have to serve as documentation.

If the module does not know how to represent the data it returns an instance of the `ICOpaqueData` type, with the raw data in its `data` attribute. Objects of this type are also acceptable values for assignment.

Besides the dictionary interface, `IC` objects have the following methods:

`IC.launchurl`(*url*[, *hint*])¶
Parse the given URL, launch the correct application and pass it the URL. The optional *hint* can be a scheme name such as `'mailto:'`, in which case incomplete URLs are completed with this scheme. If *hint* is not provided, incomplete URLs are invalid.

`IC.parseurl`(*data*[, *start*[, *end*[, *hint*]]])¶
Find an URL somewhere in *data* and return start position, end position and the URL. The optional *start* and *end* can be used to limit the search, so for instance if a user clicks in a long text field you can pass the whole text field and the click-position in *start* and this routine will return the whole URL in which the user clicked. As above, *hint* is an optional scheme used to complete incomplete URLs.

`IC.mapfile`(*file*)¶

Return the mapping entry for the given *file*, which can be passed as either a filename or an `FSSpec()` result, and which need not exist.

The mapping entry is returned as a tuple `(version, type, creator, postcreator, flags, extension, appname, postappname, mimetype, entryname)`, where *version* is the entry version number, *type* is the 4-character filetype, *creator* is the 4-character creator type, *postcreator* is the 4-character creator code of an optional application to post-process the file after downloading, *flags* are various bits specifying whether to transfer in binary or ascii and such, *extension* is the filename extension for this file type, *appname* is the printable name of the application to which this file belongs, *postappname* is the name of the postprocessing application, *mimetype* is the MIME type of this file and *entryname* is the name of this entry.

`IC.maptypecreator`(*type*, *creator*[, *filename*])¶

Return the mapping entry for files with given 4-character *type* and *creator* codes. The optional *filename* may be specified to further help finding the correct entry (if the creator code is `'????'`, for instance).

The mapping entry is returned in the same format as for *mapfile*.

`IC.settypecreator`(*file*)¶
Given an existing *file*, specified either as a filename or as an `FSSpec()` result, set its creator and type correctly based on its extension. The finder is told about the change, so the finder icon will be updated quickly.

Table Of Contents

Previous topic

37. Mac OS X specific services

Next topic

37.2. `MacOS` — Access to Mac OS interpreter features

This Page

- Show Source

Navigation

- index
- modules |
- next |
- previous |
- Python v2.6.4 documentation »
- The Python Standard Library »
- 37. Mac OS X specific services »