GAP.APP – A NATIVE MACOS INTERFACE FOR GAP TECHNICAL DETAILS

RUSS WOODROOFE

This technical document records some details of Gap.app's communication with GAP.

1. Package mode

Gap.app uses GAP's package mode (also called Window handler mode in the GAP source code), which is invoked by starting GAP with the -p flag. Package mode multiplexes the communication stream with GAP to allow certain meta commands.

From the GAP side, these are invoked with the WindowCmd function. The WindowCmd function is not documented, and should be. It takes a single array as argument. The array begins with a 3 character command code. Depending on the command code, the array should also include additional arguments.

Gap.app supports a large number of WindowCmd codes, organized into the following protocols. It would be worthwhile to formalize the notion of protocol, so that GAP code may query the capabilities of the client on which it is running.

1.1. **XGAP protocol.**

Gap.app supports all XGAP commands. The associated WindowCmd codes all begin with the letter X, and are documented in the XGAP package.

1.2. **HTML protocol.**

Gap.app supports HTML output via the HTM WindowCmd code.

- Code: HTM Arguments: String

  "HTML output". The String argument to the HTM WindowCmd code is parsed as an HTML snippet, and sent to Gap.app's output. The parsing is handled by the NSAttributedString class of macOS. The subset of HTML supported by this class is limited, but one can do some reasonably sophisticated styling of text with it.

  Using a better HTML widget, such as that offered by WebKit, is a possibility for the future. This feature is generally subject to change.

1.3. **Internal protocol.**

Gap.app supports several internal commands via WindowCmd codes beginning with an underscore "_".

- Code: _HU Arguments: String

  "Help URL". The String argument to the _HU WindowCmd code is parsed as a URL containing GAP Help. Currently, Gap.app opens this in the default macOS browser. In the future, Gap.app may support a custom help browser window.
- Code: _GI Arguments: String

  "Get Info". Currently used to get the Gap.app application bundle path, for storage in GAPInfo.

- Code: _SI Arguments: String, String
  "Set Info". Currently used to inform Gap.app about Help Viewers available in GAP.

2. Terminal codes

In addition to the powerful styling commands available via the HTM WindowCmd code, Gap.app has limited support for ANSI / VT100 terminal codes. Currently, this only supports reset (SGR code 0) and foreground color (SGR codes 30-37). It would be easy to support additional styling commands, and these may be implemented as interest arises.

Gap.app will not support the ANSI cursor positioning commands, as doing so would limit other options for more sophisticated display.

3. GAP library

Gap.app includes a small GAP library, found in

.../Gap.app/Contents/Resources/gapdotapp_init.g.

This file:

- installs a small GAP Help Viewer (which calls the _HU WindowCmd),
- installs an entry in GAPInfo.GapdotappBundlePath for the path of the Gap.app application bundle (usually /Applications/Gap.app/), and
- installs routines to support saving .gapsession files.

The GAP library also works around some minor GAP mis-features:

- Sets GAPInfo.TermEncoding to UTF-8 (the XGAP library currently resets this setting).
- PrintFormattedString from GAPDoc is careless about breaking UTF-8 characters, so Gap.app replaces it (if present) with an alternative routine.