Chapter 1: Getting started

The Dialog standard library is a framework for creating interactive stories in English, coded in the Dialog programming language. The library provides the basic implementation layer of an interactive game world—classification of objects, standard actions, parser—and encodes the knowledge required to deal with a multitude of special cases and gotchas.

At roughly 6000 lines of high-level code, the standard library is intended to be substantial yet accessible. Both casual readers and determined explorers are invited to dive into its inner workings, and see how everything fits together.

The world provided by the library is deliberately bland, serving as a neutral backdrop for the stories that you will write. Dialog makes it easy to sketch the outlines of an interactive story or old-school text adventure by merely defining a couple of objects and extending the action-handling rules of the standard library. For more ambitious works, you are encouraged to read and modify the library code, adapting and reshaping the functionality, and wrapping it tightly around the characteristics of your particular story.

Architecture of the library

The chart below illustrates the overall architecture of the standard library, and they ways in which it interfaces with story code. In the chart, blocks generally only make queries to blocks that are directly below them; that's a simplification, but it will do for now.

Running the code examples

This part of the manual contains numerous examples, and readers are encouraged to try them out. Copy the source code, put it in a text file, compile it together with the standard library, and run the resulting story file using your favourite Z-code interpreter. A quick reminder:

```
dialogc -t z8 story.dg stdlib.dg
```

where story.dg is your source code file, and stdlib.dg is the standard library source code file. The order of the filenames is significant; the story must appear before the library. In the above example, the output filename will be story.z8 (based on the output format and the name of the first source code file), but this can be changed with the -o option.

You are encouraged to put the command in a makefile or batch file, to save typing.

As you compile the example programs, you will get warnings about the lack of story title, author, and IFID. Since the example mini-stories aren't meant to be published, you may ignore these warnings. Before you release a work of IF written in Dialog, please add the required story metadata, including a freshly-generated IFID.

A minimal story

The following is a perfectly functional, albeit simple, Dialog game:

(current player
#player)

```
(current player #player)
(#player is #in #room)
(room #room)
```

[Copy to clipboard]

This game has two objects (#player and #room) in addition to a handful of default objects provided by the standard library (such as #in).

The first line says that when the game starts, the player character is represented by the #player object.

The second line says that the #player object is initially located inside another object, called #room.

The third line says that the object called #room is, in fact, a room: This tells the library that the object can be entered, that it may have connections to other rooms, and so on.

You should try this game now! There's not much of a plot, but you can poke around at the two objects (they can be referred to as ME and HERE), and try a few standard verbs such as LOOK (L), INVENTORY (I), JUMP, and QUIT (Q).

Onwards to “Chapter 2: Objects and state” • Back to the Table of Contents