Return from the Swamp

Case Studies

1. Given a client that has:

   Root: /perforce

   View:
   //depot_1/... //client_name/...
   //depot_2/... //client_name/path/...

a. What is the basic problem with this client mapping?

There is not a one-to-one mapping between the client and the server in both directions.

b. Given the above mapping, where on the client will //depot_1/path/... files be placed?

The files would not be synced to the client. Instead, the following error would be returned:

% p4 sync -f //depot_1/path/...
//depot_1/path/... - file(s) not in client view.

c. Given the above mapping, to what depot location are /perforce/path client files submitted?

The /perforce/path client files are submitted to the last matching entry in the client view:

% p4 add file.txt
//depot_2/file.txt#1 - opened for add

% p4 submit
Change 33 created with 1 open file
Submitting change 33.
Locking 1 files ...
add //depot_2/file.txt#1
Change 33 submitted.

2. In the “Excessive Wildcards Example”, what other solutions could have been used to integrate only files with selected file extensions (i.e. *.c, *.cc, *.h, *.hh, etc.)?

In addition to the solution in the presentation, there are several other good solutions. You could define a client that maps only the selected file extensions in the target path, such as:

View:

```
//depot/dir2/....c //wildcards-client2/depot/dir2/....c
//depot/dir2/....cc //wildcards-client2/depot/dir2/....cc
…
```

and then use the client to integrate using a branch specification with a more generalized view:

View:

```
//depot/dir1/... //depot/dir2/...
```