Running the IcePatch2 Client

This page describes how to un the IcePatch2 client.

On this page:

Patching with `icepatch2client`

Once the `icepatch2server` is running, you can use `icepatch2client` to get a copy of the data directory that is maintained by the server. For example:

```
$ icepatch2client --IcePatch2Client.Proxy="IcePatch2Server/server:tcp -h somehost.com -p 10000" .
```

The client expects the data directory as its single command-line argument. You must provide a proxy to the IcePatch2 server with the `IcePatch2Client.Proxy` property, as shown above.

If you have not run the client previously, it asks you whether you want to do a thorough patch. You must reply "yes" at this point (or run the client with the `-t` option to perform a thorough update). The client then executes the following steps:

1. It traverses the local data directory and creates a local `IcePatch2.sum` checksum file.
2. It obtains the relevant list of checksums from the server and compares it to the list of checksums it has built locally:
    - The client deletes each file that appears in the local checksum file but not in the server's file.
    - The client retrieves every file that appears in the server's checksum file, but not in the local checksum file.
    - The client patches every file that, locally, has a checksum that differs from the corresponding checksum on the server side.

When the client finishes, the contents of the data directory on the client side exactly match the contents of the data directory on the server side. However, only the uncompressed files are created on the client side — the server stores the compressed version of the files simply to avoid redundantly compressing a file every time it is retrieved by a client.

On the initial patch, any files that exist in the client's data directory are deleted or, if they have the same name as a file on the server, will be overwritten with the corresponding file as it exists on the server.

Using `icepatch2client` for Partial Updates

Once you have run the client, the client-side data directory contains an `IcePatch2.sum` file that reflects the contents of the data directory. If you run `icepatch2client` a second time, the program uses the contents of the local checksum file: for each entry in the local checksum file, the client compares the local checksum with the server-side checksum for the same file; if the checksums differ, the client updates the corresponding file. In addition, the client deletes any files that appear in the client's checksum file but not in the server's checksum file, and it fetches any files that appear in the server's checksum file but are missing from the client's checksum file.

If you edit a client-side file and change its contents, `icepatch2client` does *not* realize that this has happened and therefore will not patch the file to be in sync with the version on the server again. This is because the client does not automatically recompute the checksum for a file to see whether the stored checksum in `IcePatch2.sum` still agrees with the actual checksum for the current file contents.

Similarly, if you create an arbitrary file on the client side, but that file is not mentioned in either the client's or the server's checksum file, that file will simply be left alone. In other words, a normal patch operates on the differences between the client's and server's checksum files, not on any differences that could be detected by examining the contents of the file system.

If you have locally created files that have nothing to do with the distribution or if you have locally modified some files and want to make sure that those modified files are updated to reflect the contents of the same files on the server side, you must run a thorough patch with the `-t` option. This forces the client to traverse the local data directory and recompute the checksum for each file, and then compare these checksums against the server-side ones. As a result, if you edit a file locally so it differs from the server-side version, `-t` forces that file to be updated. Similarly, if you have added a file of your own on the client side that does not have a counterpart on the server side, that file will be deleted by a thorough patch.

Preventing Deletion of Local Files

By default, a normal patch deletes any files that appear in the client's checksum file but that are absent in the server's checksum file. Similarly, by default, a thorough patch deletes all files in the local data directory that do not appear in the server's checksum file. If you do not want this behavior, you can set the `IcePatch2Client.Remove` property to 0 (the default value is 1). This prevents deletion of files and directories that exist only on the client side, whether the patch is a normal patch or a thorough patch.

Patching Software Installed from Media

Suppose you distribute your application on a DVD that clients use to install the software. The DVD might be out of date so, after installation, the install script needs to perform a patch to update the application to the latest version. The script can perform a thorough patch to do this but, for large file sets, this is expensive because the client has to recompute the checksum for every file in the distribution.

To avoid this cost, you can place all the files for the distribution into a directory on the server and run `icepatch2calc -Z` on that directory. With the `-Z` option, `icepatch2calc` creates a checksum file with the correct checksums, but with a file size of 0 for each file, that is, the `-Z` option omits compressing the files (and the considerable cost associated with that). Once you have created the new `IcePatch2.sum` file in this way, you can include it on the DVD and install it on the client along with all the other files.

This guarantees that the checksum file on the client is in agreement with the actual files that were just installed and, therefore, it is sufficient for the install script to do a normal patch to update the distribution and so avoid the cost of recomputing the checksum for every file.

Setting Transfer Size for `icepatch2client`

You can set the `IcePatch2Client.ChunkSize` property to control the number of bytes that the client fetches per request. The default value is 100 kilobytes.

`icepatch2client` Command Line Options

The client supports the following options:

- `-h, --help`
  Displays a help message.
- `-v, --version`
  Displays a version number.
- `-t, --thorough`
  Do a thorough patch, recomputing all checksums.

See Also

- Running the IcePatch2 Server
- IcePatch2Client Properties