Using Slice Checksums in Ruby

The Slice compilers can optionally generate checksums of Slice definitions. For `slice2rb`, the `--checksum` option causes the compiler to generate code that adds checksums to the hash collection `Ice::SliceChecksums`. The checksums are installed automatically when the Ruby code is first parsed; no action is required by the application.

In order to verify a server's checksums, a client could simply compare the two hash objects using a comparison operator. However, this is not feasible if it is possible that the server might return a superset of the client's checksums. A more general solution is to iterate over the local checksums as demonstrated below:

Ruby

```
serverChecksums = ...
for i in Ice::SliceChecksums.keys
    if not serverChecksums.has_key?(i)
        # No match found for type id!
    elif Ice::SliceChecksums[i] != serverChecksums[i]
        # Checksum mismatch!
    end
end
```

In this example, the client first verifies that the server's dictionary contains an entry for each Slice type ID, and then it proceeds to compare the checksums.

See Also

- Slice Checksums