

Ruby Mapping for Dictionaries

Here is the definition of our [EmployeeMap](#) once more:

Slice

```
dictionary<long, Employee> EmployeeMap;
```

As for [sequences](#), the Ruby mapping does not create a separate named type for this definition. Instead, *all* dictionaries are simply instances of Ruby's hash collection type. For example:

Ruby

```
em = {}

e = Employee.new
e.number = 31
e.firstName = "James"
e.lastName = "Gosling"

em[e.number] = e
```

The Ice run time validates the elements of a dictionary to ensure that they are compatible with the declared type; a `TypeError` exception is raised if an incompatible type is encountered.

See Also

- [Dictionaries](#)
- [Ruby Mapping for Identifiers](#)
- [Ruby Mapping for Modules](#)
- [Ruby Mapping for Built-In Types](#)
- [Ruby Mapping for Enumerations](#)
- [Ruby Mapping for Structures](#)
- [Ruby Mapping for Sequences](#)
- [Ruby Mapping for Constants](#)
- [Ruby Mapping for Exceptions](#)
- [Ruby Mapping for Interfaces](#)
- [Ruby Mapping for Operations](#)