

A



Blueoak
Database Engineering

WHITE PAPER

BLUEOAK'S OCI SHIM

FAST TRACK YOUR OCI 7 MIGRATION TO
OCI 8 AND OCI 9

PREFACE

Oracle provides an application interface (API) to its database manager server via the Oracle Call Interface (OCI). This allows companies to use third-generation languages to access the Oracle database server. Frequently, the API portion of the applications is not upgraded along with the database server that they're accessing. Bugs found in the API are no longer fixed and consequently, the application works around the API bugs wasting time and effort.

Blueoak's OCI Shim (Shim) is a stand-alone C library that conceptually sits between the application and the Oracle database server. The application's OCI 7 calls are mapped to the respective OCI 8 and OCI 9 call by the Shim.

The Shim is a stand-alone C library that is linked into the application. It is not a translator or an emulator therefore there is no perceptible loss of performance.

In addition to providing an overview of the Shim, this paper discusses how Blueoak implements the Shim in parallel with a company's ongoing development effort and how those changes are merged with the main code line without loss of work.

OVERVIEW

Prior to starting the OCI port, the source trunk must be branched. The development staff continues to develop against the trunk while Blueoak starts the migration against the branch.

The branch is saved in its current state to form the baseline to be used for the generation of patch files. Patch files are generated for any code modifications that must be done in order to support OCI 8 and OCI 9. The patch files are used to merge back with the main trunk once the port is completed.

Before the migration is started, the branch is compiled and the test suite is run against it. This forms the baseline test suite that the migration needs to be reconciled against.

The Shim is a library of OCI 7 function names with the first letter capitalized. For example, `olog()` is the OCI 7 login function call. The Shim has a corresponding call named `Olog()` which accepts the exact same parameters as `olog()` with possibly one or two parameters prepended.

Blueoak has written a converter that will scan through all source files looking for OCI 7 function calls and convert them to the corresponding Shim call. The 'big-Oh' converter takes 60 seconds to run against 174,000 lines of code on a Linux, dual Athlon 1700+ machine.

After the 'big-Oh' converter is run, the application is compiled. Any additional code modifications are noted so patch files can be generated. Once the application is successfully compiled, the regression test suite is run.

After regressions are run, the results are reconciled against the baseline regression results. After any anomalies are corrected, the port is ready to be merged back to the main trunk.

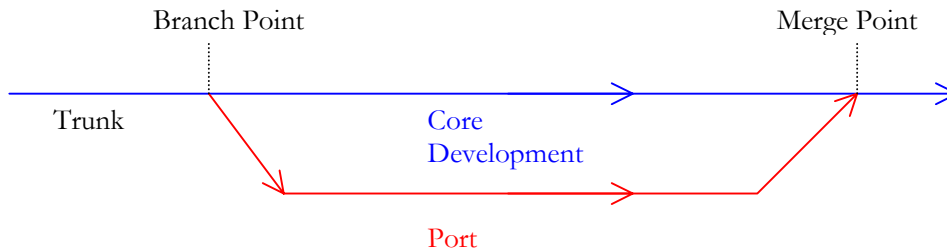
It's highly advisable that regressions are run against the main trunk prior to merging. This will allow Blueoak to ensure that after its merges, the state of the source tree is as it was prior to the merge.

The main trunk is frozen while Blueoak merges its changes. This is typically done over a weekend. The merging takes about one to two hours. The rest of the time is spent running the regression test suite and correcting any anomalies.

BRANCHING FROM THE TRUNK

Large software development efforts employ a source code control system such as CVS or SourceSafe. The files that comprise the system are known as the *source tree* since the files are typically laid out in one directory with subdirectories housing associated files. The hierarchy of the directory structures is visualized as a tree. During any point in time, there is a single *trunk* in the source tree. The main development effort is against the trunk. A *branch* is a snapshot of the trunk and is used to mark a release or to start a new effort.

The porting effort is done against a branch. This allows the core development staff to work in parallel while the migration work is in progress:



Although it's not critical, a source code management system should be in place for any software project. If there is no system in place or no concurrent development on the application, branching from the source trunk may be skipped.

GENERATING PATCH FILES

There are two Unix utilities that are employed to generate and apply patches: `diff` and `patch`. `diff` requires two files, the baseline and the changed file. The utility will generate a file with only the changes between the two files. Given an additional argument, the utility will generate a *context-difference*. A context-difference contains preceding lines and succession lines around the difference. This is a patch file that can be used by the Unix `patch` utility to patch a later version of the file.

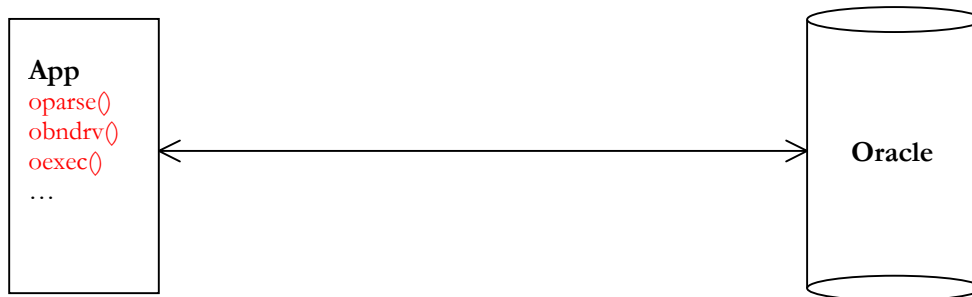
If the later version of the file is affected only in areas outside the patch, the patch will succeed. In the event that concurrent development affected the same area, the patch will fail. For these cases, resolution is handled by using editors that have merge capabilities such as `xemacs`.

HPDBE'S OCI SHIM

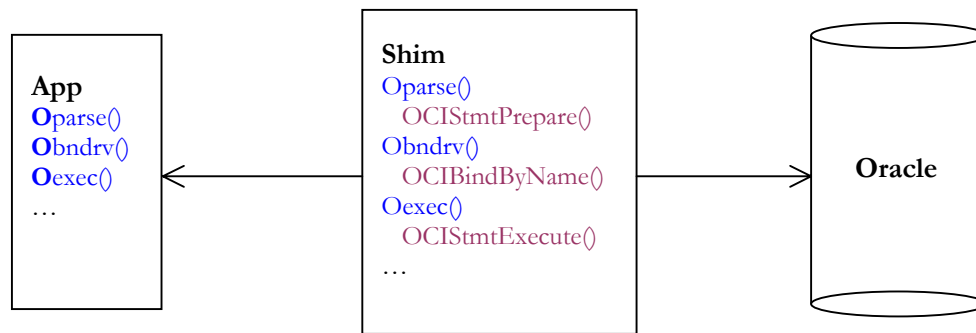
The Shim provides OCI 7 emulation using native OCI 8 and OCI 9 function calls. The OCI 7 calls in the application are modified by the [Big-Oh Converter](#) to call the corresponding Shim call. To improve readability and minimize developer training, the corresponding Shim calls are named the same as the OCI 7 call except the name is initially capitalized:

OCI 7 Call	Shim Call
<code>olog()</code>	<code>Olog()</code>
<code>obreak()</code>	<code>Obreak()</code>
...	...

The following is an example of an application making native OCI 7 calls:



The following is an example of an application calling the Shim, which in turn is issuing native OCI 8/OCI 9 calls – this example of the Shim is greatly simplified:



BIG-OH CONVERTER

The Big-Oh Converter is a parser written by Blueoak to scan the source tree and convert the existing native OCI 7 calls into Shim calls. The tool works on DOS or Unix file types. The tool can quickly scan through the source files and convert their call interface. Recently, a customer's entire source tree was converted in 60 seconds: 174,000 lines of code over 1,400 source files.

MERGING THE BRANCH TO THE TRUNK

Merging the branch to the trunk is relatively straightforward. After the port branch has passed the regression test suite, it's time to merge the branch back to the trunk.

Ideally, all functional code by the core development team is checked into the trunk before the merge. The regression test suite is run against the trunk to create a pre-merge baseline. The trunk is tagged **before-merge** to facilitate the unlikely event of a rollback after the merge.

The porting team merges their changes by checking the before-merge copy of the source tree out for write. The tree is patched using the [patch files generated](#) and the Big-Oh Converter is run on the tree. The application is compiled and the regression test suite is run. The results of the after-merge test results are compared against the results of the before-merge. Any issues are handled and the changes are checked into the trunk with a tag of **after-merge**.

The merge is now complete.