Market Requirements Document

Feature Name: System Diagnostics

Version: Draft 1 **Submitted By:** Leon Guzenda

Description of the Problem

Objectivity/DB runtime errors are very difficult to diagnose after the fact, especially in heavily loaded, concurrent applications. Specifically:

- Most of the current diagnostic mechanisms either show a historical statistical summary [ooRunstatus] or a static situation [oospace, oolockmon].
- The API call trace [debug mode] is verbose and too slow for high performance applications.
- Error messages are useful to Objectivity engineers but don't provide many clues to the application developer as to their probable cause.
- Diagnostic tools are limited to a single thread/process or server. It is difficult to interpret interactions between clients.

Description of the Requested Feature

This document defines improvements to the API call trace and runtime error messages. It also defines a new feature that will enable any client thread/process or an administrator to request a snapshot of vital Objectivity/DB state information. There are three distinct requirements:

- Flexible API call trace mechanism
- User-friendly error messages
- Snapshots (client and system wide)

Flexible API call trace mechanism

The current call trace mechanism is turned on and off with environment variables. It is not clear that all new APIs actually invoke the trace mechanism. The new mechanism should have at least the following features:

- The trace may be turned on by setting environment variables before starting a process or thread. It can also be set or changed by individual API calls.
- The destination for the trace output may be: an ASCII file; a container (with writes to that container untraced); an ooTrace object (for individual API calls only); a TCP/IP socket; or a cyclic buffer in a (potentially shared) memory area.
- Destination files or containers may be specified via an environment variable or an API call. Sockets or memory areas may only be specified via an API call.
- The level of granularity of the trace may be set to:
    - Level_0: No trace [Default]

    - Level_1: Record the name of the API call on entry and on exit, together with an indication of success or an error [exception] code.
    - Level_2: As for Level_1, but also trace input and output parameters.
    - Level_3: As for Level_2, but trace all internal Objectivity calls as well. This level might only be made usable by supplying a special password, issued by Customer Support. Internal calls will increment and decrement a "depth" counter before the entry and exit traces, respectively, to make it easier to format trace output.
- The level of granularity may be set by an environment variable. It may also be set or changed via an API call.
- Each trace entry includes a timestamp and a counter. The timestamp may be omitted. This option may be controlled by an environment variable. It may also be set or changed via an API call.
- Trace output written to a container may be viewed via an ooAssist [Eclipse] plugin.
- Trace output written to a memory area may be output to a file or a container via an API call. The memory area must be easily human readable in virtual machine dumps. The counter will make it possible to identify the sequence of entries in the cyclic buffer.

User-friendly Error Messages

If the kernel currently generates an error message it is not always easy to decode exactly what was happening at the time. For example, suppose that the user is deleting an object and that this implicitly causes the kernel to open another database and container to remove a reverse link from an associated object. It may be impossible to obtain the update lock on the remote container. The application receives an exception from the delete() call but it may not be obvious why. The current error message simply says "Error deleting object #12-34-56-7", or something similar, even though the real error occurred trying to obtain access to a completely different object. A more user-friendly message is required, e.g. "Error: Object #12-34-56-7 could not be deleted because container #15-2-0-0 is locked for update, making it impossible to remove a bi-directional association from object #15-2-25-1".

The requested feature will add advisory messages at the API level to make it easier to interpret the most probable cause of a sequence of error messages.

This may require keeping the error numbers in a stack and then providing hints based on the order in which they occur; or it may require that the error numbers are much more specific and are fully documented in the user manuals or on our support site. In the latter case the API level message can simply refer the developer to the documentation or knowledge base.

Snapshots

If an exception occurs in a client application there is very little information available about the state of the client thread/process and the associated servers. The kernel should be enhanced to optionally provide:

- A snapshot of the current date/time and Objectivity software version.
- A snapshot of the currently set transaction/session variables.
- A snapshot of the currently open databases, containers and objects, indicating whether they are open for read or update.
- A snapshot of the state of the cache providing enough detail to know: which cache pages are currently in use; whether or not they are dirty; whether or not they have been previously written during the current transaction; the logical #DB-OC-PN in each page; and the contents of the two most recently accessed pages.
- A snapshot of all statistics maintained for ooRunStatus().
- A snapshot of all active file descriptors, correlated with the databaseIDs.
- A snapshot of all active Objectivity/DB TCP/IP connections and their purpose (e.g. connection to Lock Server on HostX, connection to AMS on HostY).
- Each of the above snapshots may be enabled via an environment variable. They may also be enabled or disabled via API calls.
- The destination for the snapshots will be a file defined via an environment variable or an API call.
- The application may request a snapshot at any time.
- There should be an optional plug-in for the ooAssist framework that will take a snapshot of a thread, a process or a process and all of its threads and display the snapshot contents in a user-friendly manner.
- An API that enables a thread or process to freeze all current activity for that process and snapshot the state of the process and all of its threads.
- An API that enables a thread, process or DBA tool to freeze the activity of all associated threads and servers. It then snapshots the state of the process and all of its threads and causes the associated servers to snapshot their state.
- An API and a DBA tool that suspends the action of a particular lock server and causes it to snapshot the information available via oolockmon and oolockwait to a designated file.
- An API and a DBA tool that releases the suspension of a designated lock server.
- An API and a DBA tool that suspends the action of a particular AMS and causes it to snapshot information about its current connections and file descriptors to a designated file.
- An API and a DBA tool that releases the suspension of a designated AMS.

Part of an existing feature or does it require another feature, if so, which one?

- Flexible API call trace mechanism - Enhancement to the debug feature.
- User-friendly error messages - Enhancement to exception handling feature.
- Snapshots (client and system wide) - NEW.

How is this problem being solved now, and why isn't that acceptable?

Application developers have to add their own diagnostic facilities, or request assistance from Objectivity System Engineers or Customer Support staff. Difficult problems often require the attention of Engineering developers. Some problems have been impossible to reproduce within Objectivity's engineering environment.

The lack of adequate diagnosis capabilities decreases customer satisfaction, increases our overheads and adversely affects Engineering schedules.

What languages must support this capability?

All APIs, as the capabilities are all to be provided by the kernel.

Which platforms must be supported?

All platforms.

Do any competitors already have this feature?

Oracle, DB2, Sybase and ObjectStore (limited).

Customers who require this feature

All, particularly customers who have appeared on the weekly Warm and Hot lists.

Revenue at risk, or which could be won

Diagnosing errors accounts for a large proportion of most development efforts and is a prime source of customer dissatisfaction with deployed products. Every serious problem in a deployed application increases the risk of losing a customer. Evaluators will be more likely to choose a product that has good development and deployment tools.

When is this required?

Flexible API call trace mechanism:

- Release 9 need only implement the file and memory area options as destinations for the trace. The other destination options may be added later.
- Likewise, Level_3 trace could be provided after Release 9, if absolutely necessary.

User-friendly Error Messages

- This requirement may be satisfied by incremental improvements to the error reporting, starting with documentation of frequently encountered errors on the Customer Support web site by Release 9.

Snapshots

- Release 9, but the ooAssist plug-in may be deferred to a later release.
- The APIs and DBA tools for forcing a lock server or AMS snapshot may be deferred to Release 9.1

Additional Notes

We will also need:

- Marketing collateral
- Sales training material

- New training material