Market Requirements Document

Feature Name: Logical Delete

Version: 1 **Date Submitted:** 11/25/08 **Completed By:** Leon Guzenda

Description of the Problem

Background

1. When an object, container, or database is deleted from an Objectivity/DB federation the physical space in the client cache may or may not be freed and made available to the application. The logical construct is no longer reachable within that application. The physical resources on disk will be returned to free space or actually deleted (in the case of files) during a successful Commit or CommitAndHold operation.
2. Deleting a database automatically deletes all underlying containers and their contents.
3. Deleting a container automatically deletes its contents.
4. Deleting an object (explicitly or implicitly) automatically removes related links, index entries, map (hash-table) entries, names and (some) collection entries. It may also propagate the delete operation to other objects or containers via associations.
5. Some applications have extensive undo/redo operations, particularly in design or simulation systems.

Problem

Once an Objectivity/DB transaction has been committed there is no way to easily undo the changes it made. This makes it difficult to implement undo/redo operations in applications that need such functionality.

Description of the Requested Feature

1. A Logical_Delete operation that removes the target and all logically affected associates (e.g. via propagation or containment) from visibility within the current transaction and all subsequent transactions but doesn't actually delete it.
2. A Reinstate operation that returns the target and all ancillary information to the state it was in before the most recent Logical_Delete operation. Another client may invoke this operation after the Logical_Delete operation has been committed.
3. A Logical_Delete operation on a target that has been previously logically deleted but not reinstated should throw a warning exception.

4. Ambiguities that might arise as a result of subsequent transactions potentially causing a Reinstate operation to violate uniqueness constraints should be caught during the Reinstate operation and are to cause an exception at that time.

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

- This feature should be a standard part of Objectivity/DB.

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

Users have to implement their own mechanisms to achieve satisfactory undo/redo behavior. This is very difficult to achieve if system structures, such as scope names or unique indices, are used.

What languages must support this capability?

- C++, Java, .Net for C# and Python in that order of priority.

Which platforms must be supported?

- Windows and Linux first.
- Other Tier 2 platforms later.
- Other platforms as demand arises.

Do any competitors already have this feature?

- Some specialized DBMSs implement this feature.
- Any DBMS that uses a transaction journal is potentially able to implement this feature.

Customers who require this feature

- Applications where multiple courses of action may be explored prior to publishing a finished result.

Revenue at risk, or which could be won

- There is no immediate risk, but we could be disqualified in the cases where a prospect already has a homegrown solution with this feature.

When is this required?

- Post Release 10, preferably in late 2009.

Additional Notes

1. We will also need:

- Marketing collateral, including promotional material and updates to our web site.
- Amended technical publications and web based training material.
- New QA material to prove that the feature works and is interoperable with other platform and language combinations.