UniObjects .Net

A few years ago I started migrating my base applications to .Net environment. One of the applications requires the use of UODOTNET.dll replacing UniObjects.dll. The latter I use for more than 15 years running on VB6 without the slightest inconvenience although has the same behavior.

The UODOTNET generated an initial problem with the transactional handling that was solved by installing a more updated version of the client package (UniDK).

Currently the problem focuses on the repeated reading of a non-existent record with blocking (READU). Functions within the component respond to any error conditions by throwing an exception. It is the only way for the component to report that something different from what was expected has happened. If the Read function can read the record, no exception is thrown, so we can consider the error code to be zero.

If an exception is thrown, the source or cause that generated it must be determined. The exception message informs an error code, which can be queried from the uvoaif.txt or intcall.txt files, which are included in the UniDK. Of these codes we are interested in 30001 and 30002, which report record not on file and record blocked.

If a locked read is performed on a nonexistent item in any file, on the first attempt, the component responds with an exception code 30001. If we perform a second reading under the same conditions, without unlocking the previously read record, the component does not generate any exception, reporting that the registry was blocked and read correctly. This is clearly wrong. If we continue indefinitely, we get the same result.

To verify this behavior, I attach a test project to verify the above. The test was performed from VS2022 Community 17.5.3, UniVerse 10.1.18 and UODOTNET 3.171.1.9003. To perform the test, the project requires that you report the name of the UniVerse server, any account, username and connection password.

Test Conditions

The test can be divided into 10 parts to check the operation of the component (UniFile class). Open the project and complete the missing information to make the connection to the server that is available. In all cases the project carries out 3 equal executions. Of all the tests performed, the only one that shows a problem is 2.

Item	*Test #*	*No Lock by another process*			*Lock by another process*
		Read	*Readu*	*Release after Read*	
LIXTU (missing on file VOC)	*1*	*X*			
	2		*X*		
	3		*X*	*X*	
	4	*X*			*X*
	5		*X*		*X*
LISTU (present on file VOC)	*6*	*X*			
	7		*X*		
	8		*X*	*X*	
	9	*X*			*X*
	10		*X*		*X*