OSpell2 v2.1 Glossary

This document replaces Technical Note 0004.

Due to a number of issues on the Windows platform, the OSpell2 glossary was redesigned for OSpell2 version 2.1. A new approach has been taken to intercept keyboard events and as a result libraries that have implemented the glossary may need to be changed.

Important Note: The glossary does not work on the Linux platform. We are looking into solutions but for the time being this feature is not supported on Linux.

The following is a guide to using the new external glossary object.

Glossary object class

If you have modeled your own glossary object on the object provided by the example library, there are probably no changes required. However, it may be worthwhile looking at changes made for OWrite to the glossary object. For the latest examples please download the OWrite/OSpell2 demo software from our website. These changes are also documented below.

Attaching a field

When attaching a field to the glossary you call $setfield just as before. However, it is no longer possible to attach the glossary to a container of the target field. We recommend that you call your glossary object's $setfield method on the evBefore event of the target field.

We also recommend that you no longer stop the glossary during an evAfter event. Omnis Studio still exhibits problems in generating equally balanced evBefore and evAfter events. Under some circumstances, evAfters are generated and the focus returns to the same field without generating a matching evBefore.

Example:

```
evBefore
     Do ivGlossary.$setfield($cobj)
```

Calling $start() and $stop()

It is no longer necessary to call $start after calling $setfield of your glossary object. When the glossary's $hwnd property is assigned to a field, the glossary goes into action immediately. It is also not necessary to call $stop prior to attaching another field or when the current field loses the focus. We recommend that you only call $stop() and $start() if you want to stop the glossary momentarily. To stop the glossary permanently, call $setfield with zero.

Example:

```
Do ivGlossary.$stop()
; do something that is not to be intercepted by the glossary
Do ivGlossary.$start()
```

New methods

Two new methods have been provided for improved handling of glossary actions with OWrite. The methods $beginreplace and $endreplace can be overloaded by the library's glossary object class to capture and control the undo operation of OWrite. The method $beginreplace is called by the glossary prior to a replace

action and $endreplace is called immediately after. If you call OWrite's $startundo and $endundo it will combine all key events that are generated by the replace action into one OWrite undo operation.

You can detect that the current field is an OWrite field in your glossary's $setfield method and implement the appropriate code.

Example:

```
Method oGlossary.$setfield
     ; store the reference of the current field in your glossary instance
     Set reference ivEditor to pField
     ; attach the current field to the external glossary object
     Calculate $cinst.$hwnd as ivEditor.$framehwnd
     ; is the current field an OWrite control
     Calculate ivIsOWrite as
               (ivEditor.$objtype=kComponent)&(ivEditor.$componentlib="OWrite")

Method oGlossary.$beginreplace
     If (ivIsOWrite)
          Do ivEditor.$startundo()
     End if

Method oGlossary.$endreplace
     If (ivIsOWrite)
          Do ivEditor.$endundo("Glossary")
     End if
```

After a glossary action, OWrite will show the text "Undo Glossary" in the Undo option of the Edit menu.