VEGA

V2R7

VG-ADS/WINDOW Guide

Revision 2.7.1

May 2002

ACKNOWLEDGEMENTS

CA-ADS/OnLine and CA-IDMS/DC/UCF are proprietary trade marks or products of Computer Associates International.

WRITERS

Onni Kukkonen
Veli-Matti Niinioja

CONTENTS

Figures

2 INTRODUCTION

The CA-ADS/OnLine does not support windowing. At mapout time the terminal buffer is normally erased and thus the new map replaces the existing contents of the screen. Basic windowing capabilities are today needed to improve user interface and applications design flexibility.

Using the VG-ADS/WINDOW tool a CA-ADS/OnLine dialog may overlay the screen with smaller windows and also process the input data keyed into window fields.

User interface example

Figure 1. Application screen

```
             Order Entry

Customer   : 12345 CompAny Inc

Sales District :

Order Number   : 442211

Product        : PQWR01   Description

Quantity       : 44
```

In the **Figure 1** an end user needs the know the sales district code. He/she simply presses ENTER and the application responds with the screen of **Figure 2**.

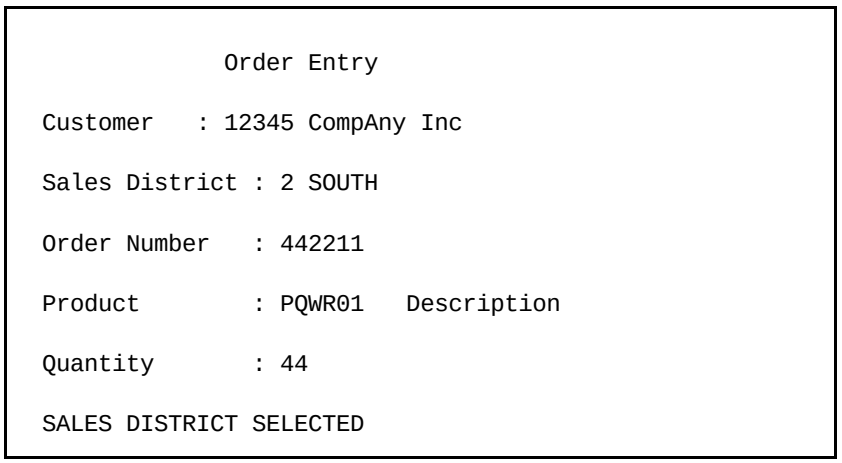
Figure 2. Application screen overlaid by a window

```
             Order Entry

Customer   : 12345 CompAny Inc
                 _____________
Sales District :! NORTH  (1) !
               ! SOUTH  (2) !
Order Number   :! WEST   (3) !
               ! EAST   (4) !
Product        :! ABROAD (9) !ription
               !_____________!
Quantity       : 44
```

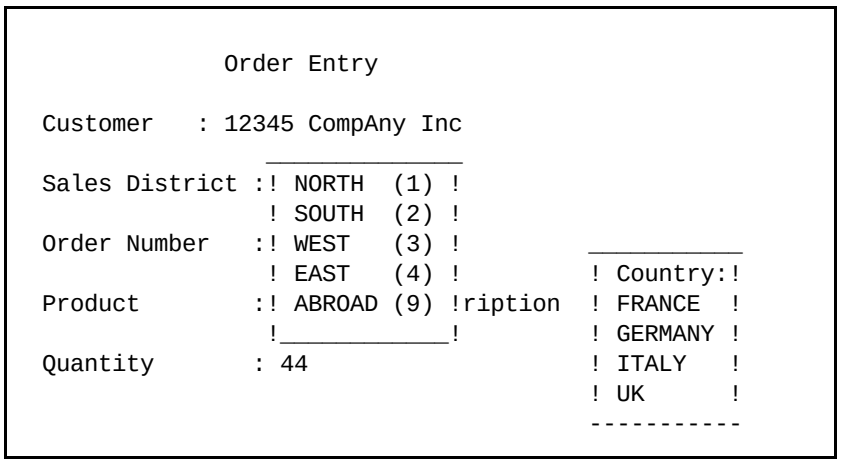
The user selects the sales district using cursor and the screen of **Figure 3** is displayed.

Figure 3. Application screen after sales district selection



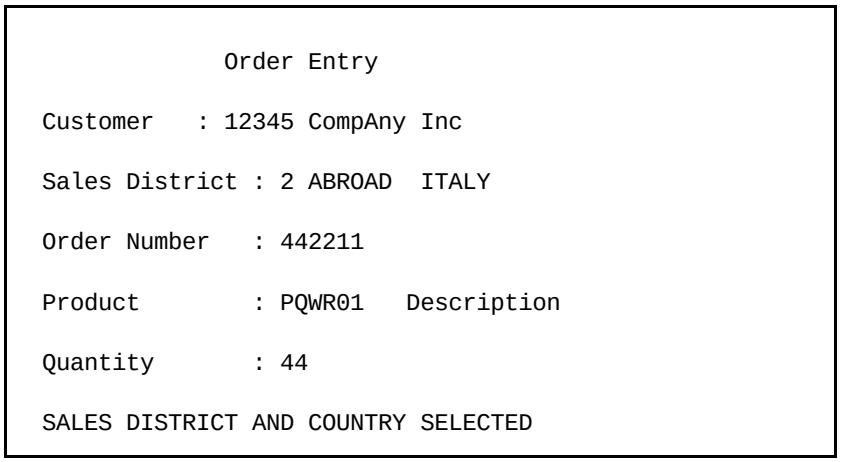
Suppose the user had selected the sales district "ABROAD". In that case the dialog displays an extended window (**Figure 4**) to select the country.

Figure 4. Application screen overlaid by two windows



The user selects the country using cursor and the screen of **Figure 5** is displayed.

Figure 5. Application screen after sales district and country selection



CA-ADS/OnLine implementation

Each window is implemented using a window dialog which is associated with a map. The map defines the window area, which should be filled with literals and data fields. A blank window area may be defined by literal with DARK attribute. Any delimiter characters may be used to indicate window border. The area outside the window should be left blank. Otherwise the window map is a standard map which may include automatic editing and message field. A pageable map is not allowed. A window may also consist of multiple non-contiguos subwindows.

A window dialog consists of a premap process and a window map. A non-window dialog should use LINK and INVOKE statements to execute the window dialog. The premap process of the window dialog displays the window map at any point using VG-ADS/WINDOW tool. The input data, cursor position and attention id used are available for processing in the window dialog. The options below are also available:

- multiple concurrent windows (only one may accept input)
- the map fields of the calling dialog may be protected before the window map is displayed (except the detail portion of a pageable map)
- the screen of the calling dialog may be redisplayed before the window is displayed (screen refresh)
- the map fields of the window displayed may be protected (e.g. screen is overlaid by a second window)
- an alternate map may be used (e.g. to display same window format on another screen position)
- an attention id (AID) key may be defined to print the screen, when the window dialog has the control

4 PROGRAMMERS GUIDE

Window dialog

A window is implemented with a dialog and a map. The premap process of the window dialog displays the window dialog's map using the VG-ADS/WINDOW utility. Never use DISPLAY statement in the premap process. No response processed are defined for the window dialog. Input processing and control statements should be written in the premap process. A standard non-window dialog normally transfers control to a window dialog using INVOKE or LINK statements.

Window map

A window map is defined using Online Mapping. The rectangular window area should be filled with data and literal fields. A blank window area may be defined by literal with DARK attribute. The required definitions for data fields are: OUTPUT DATA = YES and BACKSCAN = NO. A pageable map is not allowed.

Work record WWINDIS

A window dialog should have work record WWINDIS, which consists of control fields for window processing.

```
01  WWINDIS.

        02 WINSKIPD         PICTURE IS  X.
        02 WINSKIPW         PICTURE IS  X.
        02 WINAID           PICTURE IS  X.
        02 WINREC           PICTURE IS  X         VALUE 'Y'.
        02 WINROW           PICTURE IS  S9(4)     USAGE IS COMP.
        02 WINCOL           PICTURE IS  S9(4)     USAGE IS COMP.
        02 WINALTM          PICTURE IS  X(8).
        02 WINPCLS          PICTURE IS  S9(4)     USAGE IS COMP.
        02 WINPDEST         PICTURE IS  X(8).
        02 WINPAID          PICTURE IS  X         VALUE '<'.
        02 WINREFR          PICTURE IS  X         VALUE 'N'.
        02 FILLER           PICTURE IS  X(16).
```

Pre-window actions

The control fields below in the WWINDIS record should be set to a value desired:

WINSKIPD The protection of the upper level dialog's map fields before the window is displayed:

Y The map fields of the upper level dialog is protected before the window is displayed. The window display tool sends TEMPORARY SKIP attribute to all data fields in the map to be protected. This map should be the only map which is displayed on the screen. This value is normally used in conjunction with the first window output.

Note. The protection has no effect on the detail portion of a pageable map.

N No upper level dialog's map protection.

WINSKIPW The protection of the window map's fields when the control is to be returned back to the premap process.

Y The window map is protected when the control is to be returned back to the premap process. This value is useful in the case of multiple concurrent windows.

N No window map protection.

WINALTM This field controls the alternate window map output. The alternate window map is created from the base window map using the COPY function of OLM. No critical changes are allowed, only the screen positions, attributes and literal values may be changed. The alternate map is useful to use same window dialog to display one window format on different screen positions.

SPACE Alternate map is not used.

NONSPACE Each byte of the map name associated with the window dialog is set to the byte of this field in the corresponding position. The bytes which correspond to byte * in the WINALTM field remain unchanged. The map name created is used as a window map.

Example: Dialogs map MAXXYY0
WINALTM ******1
Window map MAXXYY1

WINPAID Attention id (AID) key, which is reserved for print screen operation when the window dialog has control. See IDD element **DC-AID-IND-V** for available values. If this field is set then the fields below are also used:

WINPCLS Printer class (1-64) used.

WINPDEST Printer destination used.

If **WINPCLS** = 0 and **WINPDEST** = SPACE then CA-IDMS/DC/UCF definitions for printer class/destination are applied.

WINREFR The refreshment of the upper level dialog's map before the window display.

Y The mapout of the upper level dialog's map is executed before the window display. This value is used in the case of multiple windows if the previously displayed window(s) should be cleared.

N No refreshment.

Window display

Use INCLUDE process **WINDOW** to display the window map associated to the window dialog. Control is returned to the next statement following the INCLUDE. You may display the window multiple times e.g. to scroll information. Automatic editing defined for the window map is applied. If there are edit errors user may use CLEAR key to return back to the premap process of the window dialog.

Post-window actions

As the control returns back to the premap process of the window dialog, it may handle the input data as well as the values of the control fields below.

WINAID The attention id (AID) key used in the last window input. The condition names below are defined for this field:

Condition name:	**AID**
ENTER-HIT	ENTER
PFnn-HIT	PFnn
PAnn-HIT	PAnn
CLEAR-HIT	CLEAR

? CLEAR-key clears the terminal buffer. Therefore the premap process should return back to upper full screen dialog or refresh the upper level screen (see WINREFR).

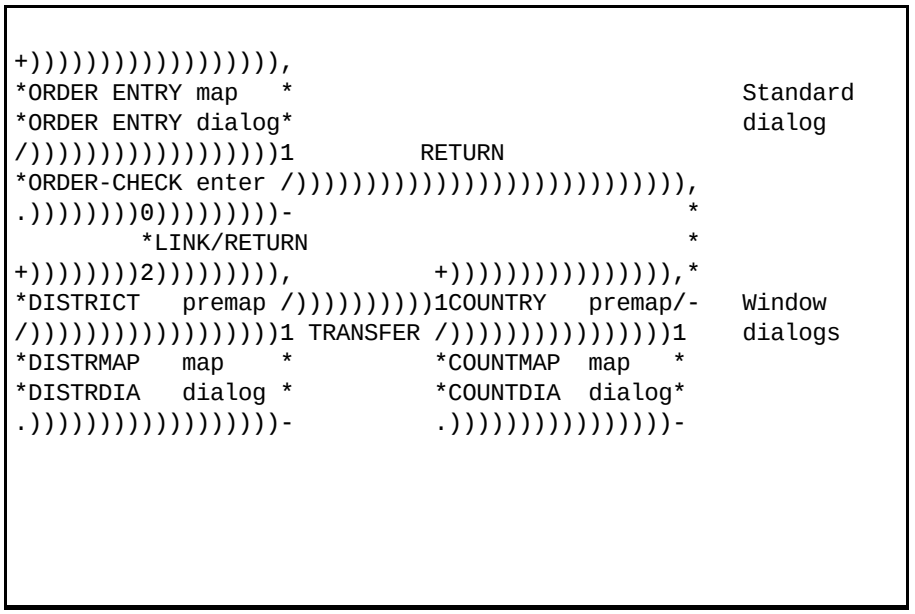
WINROW Cursor row in the last window input. Use this field instead of CA-ADS/OnLine variable CURSOR-ROW.

WINCOL Cursor column in the last window input. Use this field instead of CA-ADS/OnLine variable CURSOR-COLUMN.

Example

The code below is an implementation of the example in the section "Introduction".

Figure 6. Diagram of dialogs involved




```
 ADD
 PROCESS NAME IS ORDER-CHECK VERSION IS 1
     DESCRIPTION IS 'ORDER VALIDATION'
     PUBLIC ACCESS IS ALLOWED FOR ALL
     MODULE SOURCE FOLLOWS
LINK  'DISTRDIA'.
IF ORDC-COUNTRY-NAME NE SPACE
   DISPLAY MSG TEXT 'SALES DISTRICT AND COUNTRY SELECTED'.
ELSE
IF ORDC-DISTR-ID NE 0
   DISPLAY MSG TEXT 'SALES DISTRICT SELECTED'.
ELSE
   DISPLAY MSG TEXT 'PRESS ENTER TO SELECT SALES DISTRICT'.
     MSEND
     .


 ADD
 PROCESS NAME IS DISTRICT   VERSION IS 1
     DESCRIPTION IS 'DISTRICT RETRIEVAL'
     PUBLIC ACCESS IS ALLOWED FOR ALL
     MODULE SOURCE FOLLOWS
$    PUT DISTRICTS IN WINDOW
READY.
```

```
OBTAIN FIRST DISTRICT WITHIN IX-DISTRICT.
MOVE 1 TO I.
WHILE I LE 5 AND DB-STATUS-OK REPEAT.
   MOVE DISTR-NAME TO WINDOW1-DISTR-NAME(I).
   MOVE DISTR-ID  TO  WINDOW1-DISTR-ID(I).
   OBTAIN NEXT DISTRICT WITHIN IX-DISTRICT.
   ADD 1 TO I.
END.
MOVE 'Y' TO WINSKIPD.           $ PROTECT THE ORDER ENTRY MAP
MOVE 'Y' TO WINSKIPW.           $ PROTECT THIS WINDOW
INCLUDE WINDOW.                 $ WINDOW  DISPLAY
$    SELECT THE DISTRICT
IF CLEAR-HIT THEN DO.
   MOVE SPACE TO ORDC-DISTR-NAME.
   MOVE 0     TO ORDC-DISTR-ID.
   RETURN.
END.
ELSE DO.
  COMPUTE I = WINROW - 7.
  IF I > 0 AND I < 6 THEN DO.
     MOVE WINDOW1-DISTR-ID(I)   TO ORDC-DISTR-ID.
     MOVE WINDOW1-DISTR-NAME(I) TO ORDC-DISTR-NAME.
     IF ORDC-DISTR-ID = 9 THEN $ ABROAD
        TRAN 'COUNTDIA'.       $ EXTENDED WINDOW
     ELSE
        RETURN.
  ELSE DO.
     MOVE SPACE TO ORDC-DISTR-NAME.
     MOVE 0     TO ORDC-DISTR-ID.
     RETURN.
  END.
END.
     MSEND
     .


 ADD
 PROCESS NAME IS COUNTRY VERSION IS 1
     DESCRIPTION IS 'COUNTRY RETRIEVAL'
     PUBLIC ACCESS IS ALLOWED FOR ALL
     MODULE SOURCE FOLLOWS
$    PUT COUNTRIES IN WINDOW
READY.
OBTAIN FIRST COUNTRY WITHIN IX-COUNTRY.
MOVE 1 TO I.
WHILE I LE 4 AND DB-STATUS-OK REPEAT.
   MOVE COUNTRY-NAME TO WINDOW2-COUNTRY-NAME(I).
   OBTAIN NEXT COUNTRY WITHIN IX-COUNTRY.
   ADD 1 TO I.
END.
```

```
    MOVE 'N' TO WINSKIPD.          $ NO PROTECTION
    MOVE 'N' TO WINSKIPW.          $ NO WINDOW PROTECTION
    INCLUDE WINDOW.                $ WINDOW DISPLAY
    $    SELECT THE COUNTRY
    IF CLEAR-HIT THEN DO.
       MOVE SPACE TO ORDC-COUNTRY-NAME.
       RETURN.
    END.
    ELSE DO.
      COMPUTE I = WINROW - 11.
      IF I > 0 AND I < 5 THEN DO.
         MOVE WINDOW2-COUNTRY-NAME(I) TO ORDC-COUNTRY-NAME.
         RETURN.
      ELSE DO.
         MOVE SPACE TO ORDC-COUNTRY-NAME.
         RETURN.
      END.
    END.
    RETURN.
MSEND
.
```

6 INSTALLATION GUIDE

The installation tape

See VEGA Installation Guide section 2 how to allocate and load source and link libraries.

The first two characters of the source library member name indicate the grouping:

AW	Members for VG-ADS/WINDOW

Populate the load library and the primary dictionary

The VEGA load library is used as the VG-ADS/WINDOW load library. If VEGA is not installed customise and submit the source library member GEINS00 to allocate the load library.

Customise and submit the source libarary member AWINS01 to link edit executable module in the load library.

The PASSWRD step in the AWINS01 member is used to apply the password, which is required in order to use VG-ADS/WINDOW. In the line

```
REP 0010 NNNN,NNNN
```

modify NNNN,NNNN to the password supplied with the installation package.

Modify the CA-IDMS/DC/UCF system number NN in STEP4.

Modify CA-IDMS/DC/UCF startup JCL

Add the VG-ADS/WINDOW load library in the CDMSLIB.

Modify UCF/CICS definitions

If VG-ADS/WINDOW is used in the CA-IDMS/UCF environment, define RESETKB=TASKEND in the #UCFCICS macro. Assemble and link edit UCFCICS.

Restart CA-IDMS/DC/UCF and test

The dialogs DWINEX, DWINEX1 and DWINEX2 are generated using CA-ADS release 10.21. If your CA-ADS release is 10.2 generate maps SWINEX, SWINEX1 and SWINEX2 as well as dialogs DWINEX, DWINEX1 and DWINEX2.

Enter ADS DWINEX to execute the demonstration application.

Load the VG-ADS/WINDOW definitions to application dictionaries

Load definitions in each application dcitionary used in VG-ADS/WINDOW application development. Customise and submit the source library member AWINS01A.

Remember to modify the following member for each dictionary:

SIGNONA Valid DDDL signon statement for the application dictionary.

APPENDIX 1. ABEND CODES

Code	Description	Corrective action
WIN+	Storage id WIN+ not found	Contact your VG-ADS/WINDOW technical support
WIN*	Storage id WIN* not found	Report error to VG-ADS/WINDOW technical support.
VG??	Password expired	Report error to VG-ADS/WINDOW technical support.