Table of Contents

[Screen Shot of EXTCRS01 2](#_Toc323890750)

[Overview 2](#_Toc323890751)

[Forms Startup 2](#_Toc323890752)

[Main Menu 3](#_Toc323890753)

[Exit Button 3](#_Toc323890754)

[Data Fields (#1) 3](#_Toc323890755)

[Record processing 3](#_Toc323890756)

[Buttons (#2) 4](#_Toc323890757)

[View Journal 4](#_Toc323890758)

[The Data Fields 4](#_Toc323890759)

[Newer/Older Buttons 4](#_Toc323890760)

[Close Window Button 4](#_Toc323890761)

[Progress field (#3) 5](#_Toc323890762)

[TRIGGER CODE 6](#_Toc323890763)

[PRE-INSERT 6](#_Toc323890764)

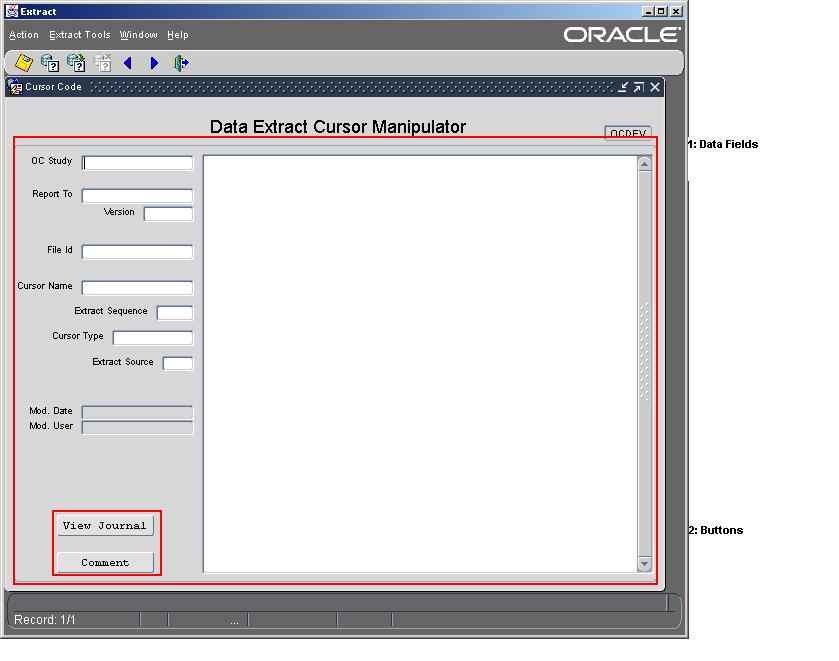
[PRE-UPDATE 6](#_Toc323890765)

[PRE-DELETE 7](#_Toc323890766)

Detailed analysis of EXTCRS01.FMB

Note: Used TEXTPAD to look inside form as the 6i Form Builder was unable to read form.

# Screen Shot of EXTCRS01



# Overview

This form is used to manipulate the cursor SQL code for a File Id of a defined extraction. It also allows the user to see previous version of the same cursor.

# Forms Startup

1. Security Checked at form startup. The following query checks userid to see if user is valid for APPLICATION. (WHEN-NEW-FORM-INSTANCE)

SELECT '1' INTO T\_WHAT

FROM CT\_EXT\_ACCOUNTS

WHERE (USER\_NAME = USER or User = 'CTEXT\_T')

AND ROWNUM = 1;

1. Menu Actions are turn off that would normally allow queries and editing.

# Main Menu

The main menu controls what other screens can be selected. They are controlled through ROLES. Only certain Oracle ROLES granted to the user would allow access. See EXTMENU1\_Analysis for more information.

# Exit Button

The exit button is used to close the application. When closing the application, the screen is checked for existing non-committed transactions and will not quite until the transactions are committed or rolled back.

# Data Fields (#1)

There are 9 enterable fields that make up the data field for a cursor definition. They are stored in the table CT\_EXT\_CRS\_CTL.

Functions required: Enter Query, EDIT/UPDATE, DELETE (May have to look into ROLE based display of these options)

Note: When a record is modified, the “before modified” version of the record must be copied to the Cursor Journal table. Also, a Comment must be captured when a record is modified.

|  |  |  |
| --- | --- | --- |
| Screen Field | Table Column | Description |
| OC Study | OC\_STUDY | Study identifier |
| Report To | REPORT\_TO | The Report Type of the definition |
| Version | VERSION | The Version of the Report Type |
| File Id | FILE\_ID | The File Type Identifier |
| Cursor Name | CRS\_NAME | The name of the cursor |
| Extract Sequence | EXT\_SEQ | The sequence in which to execute this cursor when there are more than one cursor associated to a File Id. |
| Cursor Type | CRS\_TYPE | The type of cursor defined. Mainly NORMAL, COM, NORMAL\_DEL, COM\_DEL, LAB\_OTHER, etc. See below for Cursor Type and Execution |
| Extract Source | EXT\_SOURCE | The Extract Source type, “N”,”O” or “Y”. See below for Extract Source details |
| “Cursor Code” | TEXT | This is the actual SQL code that makes up the select statement to extract the data. NOTE: This is stored in a LONG column. |
| Mod. Date | MODIFIED\_BY | User who last modified the record. |
| Mod. User | MODIFIED\_DATE | The date the records was modified. |
|  | CREATE\_BY | User who created record, NOT DISPLAYED |
|  | CREATION\_DATE, | Date record created, NOT DISPLAYED |
|  | NOTE | Comment field used to explain/describe modifications NOT DISPLAYED. |

### Record processing

DELETES: Before a record is committed as a delete from the table, a copy of it is placed into the Journal Table, and journaling information is stored with the record. See PRE-DELETE trigger for entire PL/SQL code.

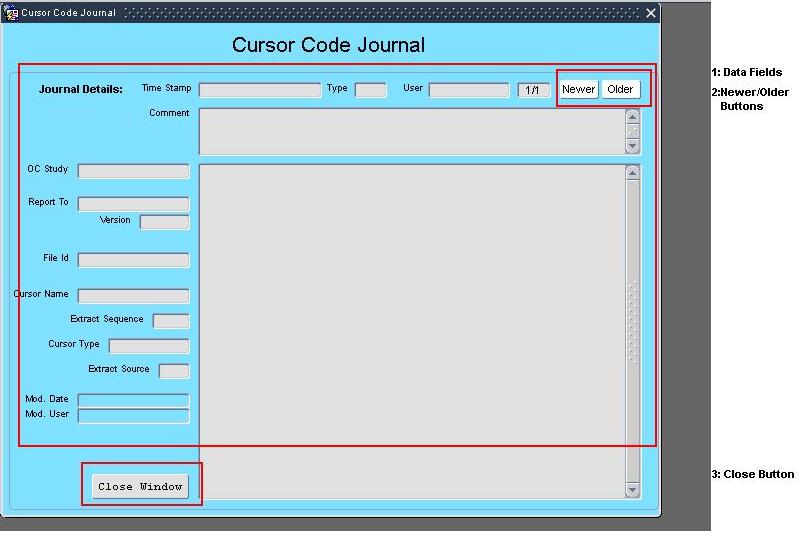
UPDATES: Before a record is committed as an update, a copy of it is placed into the Journal Table, and journaling information is stored with the record. See PRE-UPDATE trigger for entire PL/SQL code.

INSERTS: Before a new record is committed, a copy of it is placed into the Journal Table, and journaling information is stored with the record. See PRE-UPDATE trigger for entire PL/SQL code.

# Buttons (#2)

### View Journal

The View Journal Button is used to query past cursor definitions versions.



* Only fires if OC\_STUDY, REPORT\_TO, VERSION, and CRS\_NAME fields have been entered.
* Displays all journal entries for the cursor displayed, using the following query:

where CT\_EXT\_CRS\_CTL$JN.OC\_STUDY = :CT\_EXT\_CRS\_CTL.OC\_STUDY

and CT\_EXT\_CRS\_CTL$JN.REPORT\_TO= :CT\_EXT\_CRS\_CTL.REPORT\_TO

and CT\_EXT\_CRS\_CTL$JN.VERSION = :CT\_EXT\_CRS\_CTL.VERSION

and CT\_EXT\_CRS\_CTL$JN.CRS\_NAME = :CT\_EXT\_CRS\_CTL.CRS\_NAME

#### The Data Fields

The data fields are read-only and come from the table CT\_EXT\_CRS\_CTL$JN. Data is sorted by most recent record.

#### Newer/Older Buttons

These buttons will traverse the user through the journal records.

#### Close Window Button

This button is used to close the journal window and return the user to the primary data entry form.

### Comment

The comment button is used to place a comment on the current record. Comments are also collected during updates and deletes.

# TRIGGER CODE

#### PRE-INSERT

DECLARE

T\_TEXT VARCHAR2(120) := ' ';

BEGIN

T\_TEXT := ' Inserting journaling information failed.';

Insert into CT\_EXT\_CRS\_CTL$JN (

JN\_OPERATION ,JN\_TIMESTAMP

,JN\_SN

,JN\_ORACLE\_USER

,OC\_STUDY ,REPORT\_TO

,VERSION ,FILE\_ID

,CRS\_NAME ,CRS\_TYPE

,EXT\_SEQ ,TEXT

,EXT\_SOURCE ,NOTE )

VALUES (

'INS' ,SYSDATE

,TO\_NUMBER(TO\_CHAR(SYSDATE, 'YYYYMMDDHH24MISS'))

,USER

,:CT\_EXT\_CRS\_CTL.OC\_STUDY ,:CT\_EXT\_CRS\_CTL.REPORT\_TO

,:CT\_EXT\_CRS\_CTL.VERSION ,:CT\_EXT\_CRS\_CTL.FILE\_ID

,:CT\_EXT\_CRS\_CTL.CRS\_NAME ,:CT\_EXT\_CRS\_CTL.CRS\_TYPE

,:CT\_EXT\_CRS\_CTL.EXT\_SEQ ,:CT\_EXT\_CRS\_CTL.TEXT

,:CT\_EXT\_CRS\_CTL.EXT\_SOURCE ,:CT\_EXT\_CRS\_CTL.NOTE

);

IF NVL(:CT\_EXT\_CRS\_CTL.NOTE,' ') <> NVL(:CT\_EXT\_CRS\_CTL.NOTE\_DISPLAY,' ') THEN

:CT\_EXT\_CRS\_CTL.NOTE\_DISPLAY := :CT\_EXT\_CRS\_CTL.NOTE;

END IF;

EXCEPTION

WHEN OTHERS THEN

MESSAGE(T\_TEXT, ACKNOWLEDGE);

RAISE FORM\_TRIGGER\_FAILURE;

END;

#### PRE-UPDATE

DECLARE

T\_TEXT VARCHAR2(120) := ' ';

H\_OC\_STUDY CT\_EXT\_CRS\_CTL.OC\_STUDY%Type;

H\_REPORT\_TO CT\_EXT\_CRS\_CTL.Report\_to%Type;

H\_VERSION CT\_EXT\_CRS\_CTL.Version%Type;

H\_FILE\_ID CT\_EXT\_CRS\_CTL.File\_id%Type;

H\_CRS\_NAME CT\_EXT\_CRS\_CTL.Crs\_name%Type;

H\_CRS\_TYPE CT\_EXT\_CRS\_CTL.Crs\_type%Type;

H\_EXT\_SEQ CT\_EXT\_CRS\_CTL.Ext\_seq%Type;

H\_TEXT CT\_EXT\_CRS\_CTL.Text%Type;

H\_EXT\_SOURCE CT\_EXT\_CRS\_CTL.Ext\_source%Type;

H\_NOTE CT\_EXT\_CRS\_CTL.Note%Type;

H\_Mod\_by CT\_EXT\_CRS\_CTL.Modified\_by%Type;

H\_Mod\_date CT\_EXT\_CRS\_CTL.Modified\_date%Type;

h\_rowid varchar2(30) :=NAME\_IN(:SYSTEM.CURRENT\_BLOCK || '.ROWID');

BEGIN

T\_TEXT := ' Updating journaling information failed.';

:ct\_ext\_CRS\_ctl.modified\_by := user;

:ct\_ext\_CRS\_ctl.modified\_date := sysdate;

select OC\_STUDY, REPORT\_TO, VERSION, FILE\_ID,

CRS\_NAME, CRS\_TYPE, EXT\_SEQ, TEXT, EXT\_SOURCE, NOTE,

MODIFIED\_BY, MODIFIED\_DATE

into H\_OC\_STUDY, H\_REPORT\_TO, H\_VERSION, H\_FILE\_ID,

H\_CRS\_NAME, H\_CRS\_TYPE, H\_EXT\_SEQ, H\_TEXT, H\_EXT\_SOURCE, H\_NOTE,

H\_MOD\_BY, H\_MOD\_DATE

from CT\_EXT\_CRS\_CTL

where rowid = h\_rowid;

Insert into CT\_EXT\_CRS\_CTL$JN (

JN\_OPERATION ,JN\_TIMESTAMP

,JN\_SN

,JN\_ORACLE\_USER

,OC\_STUDY ,REPORT\_TO

,VERSION ,FILE\_ID

,CRS\_NAME ,CRS\_TYPE

,EXT\_SEQ ,TEXT

,EXT\_SOURCE ,NOTE

,MODIFIED\_BY ,MODIFIED\_DATE )

VALUES (

'UPD' ,SYSDATE

,TO\_NUMBER(TO\_CHAR(SYSDATE, 'YYYYMMDDHH24MISS'))

,USER

,H\_OC\_STUDY ,H\_REPORT\_TO

,H\_VERSION ,H\_FILE\_ID

,H\_CRS\_NAME ,H\_CRS\_TYPE

,H\_EXT\_SEQ ,H\_TEXT

,H\_EXT\_SOURCE ,H\_NOTE

,H\_MOD\_BY ,H\_MOD\_DATE );

IF NVL(:CT\_EXT\_CRS\_CTL.NOTE,' ') <> NVL(:CT\_EXT\_CRS\_CTL.NOTE\_DISPLAY,' ') THEN

:CT\_EXT\_CRS\_CTL.NOTE\_DISPLAY := :CT\_EXT\_CRS\_CTL.NOTE;

END IF;

EXCEPTION

WHEN OTHERS THEN

MESSAGE(T\_TEXT, ACKNOWLEDGE);

RAISE FORM\_TRIGGER\_FAILURE;

END;

#### PRE-DELETE

DECLARE

T\_TEXT VARCHAR2(120) := ' ';

H\_OC\_STUDY CT\_EXT\_CRS\_CTL.OC\_STUDY%Type;

H\_REPORT\_TO CT\_EXT\_CRS\_CTL.Report\_to%Type;

H\_VERSION CT\_EXT\_CRS\_CTL.Version%Type;

H\_FILE\_ID CT\_EXT\_CRS\_CTL.File\_id%Type;

H\_CRS\_NAME CT\_EXT\_CRS\_CTL.Crs\_name%Type;

H\_CRS\_TYPE CT\_EXT\_CRS\_CTL.Crs\_type%Type;

H\_EXT\_SEQ CT\_EXT\_CRS\_CTL.Ext\_seq%Type;

H\_TEXT CT\_EXT\_CRS\_CTL.Text%Type;

H\_EXT\_SOURCE CT\_EXT\_CRS\_CTL.Ext\_source%Type;

H\_NOTE CT\_EXT\_CRS\_CTL.Note%Type;

H\_Mod\_by CT\_EXT\_CRS\_CTL.Modified\_by%Type;

H\_Mod\_date CT\_EXT\_CRS\_CTL.Modified\_date%Type;

h\_rowid varchar2(30) :=NAME\_IN(:SYSTEM.CURRENT\_BLOCK || '.ROWID');

BEGIN

T\_TEXT := ' Deleting journaling information failed.';

select OC\_STUDY, REPORT\_TO, VERSION, FILE\_ID,

CRS\_NAME, CRS\_TYPE, EXT\_SEQ, TEXT, EXT\_SOURCE, NOTE,

MODIFIED\_BY, MODIFIED\_DATE

into H\_OC\_STUDY, H\_REPORT\_TO, H\_VERSION, H\_FILE\_ID,

H\_CRS\_NAME, H\_CRS\_TYPE, H\_EXT\_SEQ, H\_TEXT, H\_EXT\_SOURCE, H\_NOTE,

H\_MOD\_BY, H\_MOD\_DATE

from CT\_EXT\_CRS\_CTL

where rowid = h\_rowid;

Insert into CT\_EXT\_CRS\_CTL$JN (

JN\_OPERATION ,JN\_TIMESTAMP

,JN\_SN

,JN\_ORACLE\_USER

,OC\_STUDY ,REPORT\_TO

,VERSION ,FILE\_ID

,CRS\_NAME ,CRS\_TYPE

,EXT\_SEQ ,TEXT

,EXT\_SOURCE ,NOTE

,MODIFIED\_BY ,MODIFIED\_DATE )

VALUES (

'DEL' ,SYSDATE

,TO\_NUMBER(TO\_CHAR(SYSDATE, 'YYYYMMDDHH24MISS'))

,USER

,H\_OC\_STUDY ,H\_REPORT\_TO

,H\_VERSION ,H\_FILE\_ID

,H\_CRS\_NAME ,H\_CRS\_TYPE

,H\_EXT\_SEQ ,H\_TEXT

,H\_EXT\_SOURCE ,H\_NOTE

,H\_MOD\_BY ,H\_MOD\_DATE );

IF NVL(:CT\_EXT\_CRS\_CTL.NOTE,' ') <> NVL(:CT\_EXT\_CRS\_CTL.NOTE\_DISPLAY,' ') THEN

:CT\_EXT\_CRS\_CTL.NOTE\_DISPLAY := :CT\_EXT\_CRS\_CTL.NOTE;

END IF;

EXCEPTION

WHEN OTHERS THEN

MESSAGE(T\_TEXT, ACKNOWLEDGE);

RAISE FORM\_TRIGGER\_FAILURE;

END;