FLORA INDUSTRY DATABASE MANAGEMENT SYSTEM

PROGRAMMER/DEVELOPER MANUAL

by Sarah McEvoy

Department of Conservation and Land Management
February 1997

TABLE OF CONTENTS

FLORA INDUSTRY DATABASE MANAGEMENT SYSTEM

FIDMS Programmer's Help

This help system is for ObjectPAL programmers who would like to expand FIDMS and learn more about developing applications in Paradox for Windows. Click a topic for more information.

Common Variables (see p 2)	Overview of variables used throughout FIDMS
Tables (see p 3)	Overview of the tables used by FIDMS
Data models (see p 35)	Overview of the data models used by FIDMS Code Help)
Code (see p 36)	Source code for FIDMS, organized by object (to view this information within FIDMS, right-click an object, then choose Code Help)
Libraries (see p 71)	Overview of the libraries used by FIDMS
Reports (see p 144)	Overview of the reports used by FIDMS

Common Variables

When an application uses a variable for a similar purpose throughout its forms, it is helpful to give it one consistent name. Variables common to all a form's methods are defined in the form's Var section. Click a topic for more information on common variables used throughout FIDMS.

formName

Variable of type Form. Identifies the form being used. FIDMS uses formName to get the title of the active form. It also uses it to retrieve the form's name for context-sensitive Help.

theKey

Variable of type String. Set in keyPhysical methods--using eventInfo.VChar()--to the key pressed by the user. FIDMS uses kPhysical to provide shortcut keys to the user.

FIDMSLib

Variable of type Library. Set in open forms to the MENUS.LSL library, which provides methods for retrieving the active page name, processing menu commands, and checking whether the application started properly.

uio

Variable of type uiObject. Set in mouseEnter and mouseRightUp methods--using eventInfo.GetTarget(uio)--to the target of a mouse operation like pointing or right-clicking. It is used to get the name (stored in uioName) and class of objects.

uioName

Variable of type String. Set to the name of an object (indicated by uio) affected by a mouse operation. It is used to display programmer Help.

Tables

Application tables

FIDMS uses permanent application tables to store person and licence data, flora return data, endorsement data, display popup menus, and display context-sensitive Help. Go to the Table Name for more information on a table.

borquota.db	**burghop.db**	**end_no.db**	**endorse.db**
end_loc.db	**end_spec.db**	**geograph.db**	**grid_hop.db**
hbttaxon.db	**hbttrans.db**	**hop_burg.db**	**hopgenus.db**
hopseed.db	**hopsrce.db**	**licence.db**	**loc.db**
location.db	**loiscode.db**	**msghelp.db**	**msgusr.db**
persons.db	**picture.db**	**postcds.db**	**pri.db**
retex.db	**retno.db**	**retnum.db**	**vehicle.db**

Temporary tables

Tables which are used by the FIDMS application to hold temporary data which may be displayed in forms or reports. Click a topic for more information on a table.

cur_lic.db	**endprn.db**	**endrep.db**	**endtmp.db**
endtmpsp.db	**lic_stat.db**	**licloc.db**	**licprn.db**
norenew.db	**objlist.db**	**renew.db**	**retprn.db**
retstat.db	**speccnt.db**	**special.db**	**sploc.db**
tempsrc.db			

Application Tables

borquota.db

This table is used by the Boronia quota Report (borquota.rsl) to provide a summary of quotas of *Boronia megastigma* on Crown land taken as blossom, sprays or seed. It is issued to traditional pickers only.

Field	Field Type	Description
Region	A15	Name of CALM Region (Central Forest or Southern Forest)
District	A15	Name of CALM District
Block	A25	Name of area allocated (usually CALM block)
Product	A10	Either blossom, sprays or seed
Quota	I	Amount of quota (in kg)
Picker	A15	Name of traditional boronia picker allocated the block.

burghop.db

This table is used by the Statistics Form (STATS.FSL) as permanent data for comparison with statistics from "The Wildflower Industry" by Mark Burgman and Stephen Hopper (1982) and from Rye (1978). It holds the number of species which were exploited as seed and as stems.

Validity Checks
Keyfields Time Period

Field	Field Type	Description
Time Period	N	Year that the data applies to (i.e. 1978 or 1981)
Stems	N	Number of species exploited as stems
Seed	N	The number of species exploited as seed

end_no.db

Used to obtain unique endorsement number values by the endorsedb table. Not accessed directly by the user.

Field	Field Type	Description
Endorse_no	N	New unique number assigned to the endorse.db table when new endorsement is selected on menu.

endorse.db

Contains details of endorsements issued by Region/District Offices for valid current licence holders. Table endorse.db is sorted by Endorsement_no.

Validity Checks

Keyfields Endorsement_no

Secondary Index Licence_no

Field	Field Type	Description
Endorsement_no	N	Unique number automatically created when new endorsement is selected on menu. Links the endloc.db, endspec.db and vehicle.db tables.
Licence_number	A8	Unique number taken from Licences table. Values will only be accepted if the licence number is present in the licences.db table.
Issue_officer	A45	Name of CALM officer authorised to issue endorsement
Region	A20	CALM Region - selected using drop-down list
District	A20	CALM District - selected using drop-down list
Endorse_from	D	First day of endorsement - must be within valid period of licence
Endorse_to	D	Last day of endorsement - no longer than 3 months

end_loc.db

Contains details of areas allocated for endorsement issued by Region/District Offices for valid current licence holders. Table end_loc.db is sorted by Block_code and End_no.

Validity Checks

Keyfields Block_code
End_no

Secondary Index
End_no

Referential Integrity
End_no
grid

Field	Field Type	Description
Block_code	A10	LOIS codes (two letters) which are provided by lookup.
Location Name	A25	Usually a forest block
End_no	N	Unique number automatically created when new endorsement is selected on menu. Links the endloc.db, endspec.db and vehicle.db tables.
Location Description	A50	Description of area bounded by roads, rivers, etc.
Grid No	I	Corresponds to picker grid cells. Valid values enforced by link to geograph.db

end_spec.db

Contains details of species allocated to each area for endorsements issued by Region/District Offices for valid current licence holders. Table end_spec.db is sorted by Location Name, End_no and Species_code.

Validity Checks

Keyfields Location Name
End_no
Species_code

Field	Field Type	Description
Location Name	A25	Usually a forest block
End_no	N	Unique number automatically created when new endorsement is selected on menu. Links the endloc.db, endspec.db and vehicle.db tables.
Species_code	N -	Unique number used by Herbarium WACENSUS database (current only valid)
Genus	A30	Permitted genus; linked to WACENSUS (filled automatically)
Species	A30	Permitted species;linked to WACENSUS (filled automatically)
InfraSpRank	A9	InfraspRank name is linked to WACENSUS (filled automatically)
InfraSpName	A30	InfraspName name is linked to WACENSUS (filled automatically)
Quantity	N	The amount of flora the picker is permitted to harvest
Unit	A10	Should be either single or kg (NOT bunches)
Part	A10	The product permitted to be harvested
End_loc_no	N	Valid location or reserve number of area. Linked to Location_No in end_loc.db table

geograph.db

Stores lookup values for valid picking grid squares and the correponding CALM Region, CALM District or CALM Flora Industry Region. Table is sorted by Grid Square.

Validity Checks
keyfield Grid Square

Field	Field Type	Description
Grid Square	I	Corresponds to picker grid cells shown on reverse of flora return form.
Region	A20	CALM Region
District	A20	CALM District
Picking Region	A20	Flora Industry Region (as shown in WA Flora Management Program)

grid_hop.db

This table is used by the Statistics Form (STATS.FSL) as permanent data for comparison with statistics from "The Wildflower Industry" by Mark Burgman and Stephen Hopper (1982). It holds the number of species exploited as stems per grid square.

Validity Checks
Keyfields Time Period

Field	Field Type	Description
Grid Square	S	Corresponds to picker grid cells shown on reverse of a return form
NoSpecies	I	Number of species exploited as stems within each grid square
% total harvest	N	The percentage of the total harvest coming from each grid square

hbttaxon.db

Contains taxonomic details of all vascular taxa (both current and non-current) to provide validation of species data entered into FIDMS. Maintained by the WA Herbarium staff Table hbttaxon.db is sorted by Taxon ID. READ ONLY

Validity Checks
Keyfields Taxon Id

Field	Field Type	Description
Taxon Id	N	Unique number created by WA Herbarium. Provides validation of species data
Curnt	A1	Indicates whether a taxon id and name are valid.
Genus	A30	Genus name as determined by WA Herbarium
Species	A60	Species name as determined by WA Herbarium
InfraSpRank	A9	InfraspRank name as determined by WA Herbarium
InfraSpName	A70	InfraspName name as determined by WA Herbarium
DRFstatus	A1	Not currently in use.
Consstatus	A1	Not currently in use
Priority	A2	Not currently in use

hbttrans.db

Contains transaction details of all changes, updates, additions and deletions to hbttaxon.db, the taxonomic database. Maintained by the WA Herbarium staff Table hbttrans.db is sorted by Oldid and newid. READ ONLY.

Validity Checks
Keyfields Taxon Id

Field	Field Type	Description
OldId	N	Previous taxon id number created by WA Herbarium. Provides the link to the hbttaxon.db table via the taxonid field.
Newid	N	New unique number given to a taxon following the transaction.
Transno	N	Unique internal accounting number given by WA Herbarium staff
Transtype	A3	Describes the nature of the amendment, e.g. excluded, error, etc.
Transdate	@	Timestamp of when transaction was carried out
Transuser	A16	Log in user name of person carrying out transaction
Active	A1	Describes whether a transaction is active or not
Authorisedby	A60	Authority/reason for amendment, e.g. publication, taxonomic change
Authorisedon	@	Timestamp of when change was authorised
AuthorisedYr	N	Year of authorisation as number
AuthorisedMn	N	Month of authorisation as number
AuthorisedDy	N	Day of authorisation as number

hop_burg.db

This table is used by the Statistics Form (STATS.FSL) as permanent data for comparison with statistics from "The Wildflower Industry" by Mark Burgman and Stephen Hopper (1982). It holds the total amount of harvesting for each species exploited as stems (in single stems) and the percentage that each species is of the total harvest of all stems.

Validity Checks
Keyfields TaxonId

Field	Field Type	Description
TaxonID	N	Unique number created by WA Herbarium. Provides validation of species data and a link to other species data
Genus	A25	Genus name as determined by WA Herbarium
Species	A25	Species name as determined by WA Herbarium
Harvest Quantity(80/81)	I	Total quantity of harvest per species in single stems
% total harvest	N	The percentage of the total harvest for each species

hopgenus.db

This table is used by the Statistics Form (STATS.FSL) as permanent data for comparison with statistics from "The Wildflower Industry" by Mark Burgman and Stephen Hopper (1982). It holds the total amount of harvesting for each genus exploited as stems (in single stems) and the number of species that are exploited within each genus.

Validity Checks
Keyfields Genus

Field	Field Type	Description
Genus	A30	Genus name as determined by WA Herbarium
No of stems	N	Total quantity of harvest per genus in single stems
No of species	N	Number of species within each genus

hopseed.db

This table is used by the Statistics Form (STATS.FSL) as permanent data for comparison with statistics from "The Wildflower Industry" by Mark Burgman and Stephen Hopper (1982). It holds the total amount of harvesting for each genus exploited as seed (in single stems) and the number of species that are exploited within each genus.

Validity Checks
Keyfields Genus

Field	Field Type	Description
Genus	A30	Genus name as determined by WA Herbarium
No of stems	N	Total quantity of harvest per genus in kg seed
No of species	N	Number of species within each genus

hopsrce.db

This table is used by the Statistics Form (STATS.FSL) as permanent data for comparison with statistics from "The Wildflower Industry" by Mark Burgman and Stephen Hopper (1982). It shows the breakdown of harvest of single stems by month and land status (i.e. Crown or private).

Validity Checks
Keyfields Mth_num

Field	Field Type	Description
Mth_num	S	Month of harvest as number (joins with other data)
Month	A12	Month of harvest as an alphanumeric field (for display purposes)
Cqty	N	Amount of harvest on Crown land in single stems
Pqty	N	Amount of harvest on private land in single stems

licence.db

Contains licence details for all Commercial Purposes and Commercial Producer's/Nurseryman's Licences ever issued. Licence.db is sorted by Licence_no.

Validity Checks
Keyfield Licence_no

Field	Field Type	Description
Licence_no	A8	Unique number issued annually
Person _no	I -	Internal accounting unique number for this person (read-only)
Licence_type	A2	Either CP (Commercial Purposes) or PN (Commercial Producers or Nurserymans)
Valid_from	D	First valid date of the licence
Expiry_date	D	Last valid date of the licence
Date_of_issue	D	Date that the licence was issued

loc.db

Contains location details of all approved harvesting locations for Commercial Flora licences. Maintained by licensing staff. READ ONLY

Validity Checks
Keyfields Licence No
Location

Field	Field Type	Description
Licence_no	A8	Unique number issued annually
Location	A60	Details of approved land area

location.db

Contains details of all valid CALM Districts and Regions for lookup purposes in Endorsement and Reports Form. READ ONLY

Field	Field Type	Description
Location	A25	Name of CALM District or Region

loiscode.db

This table is used by the CALM Endorsement Form (ENDORSFM.FSL) as a lookup table to validate block names and codes. It lists a two letter block code as is used by the LOIS (Logging Operation Inventory System) system to provide a lookup code for the block name and convenience.

Validity Checks
Keyfields Taxonld

Field	Field Type	Description
Block_code	A10	Provided by the LOIS system as a lookup value
Region	A20	CALM Region
District	A20	CALM District
Block_name	A100	Name of block or area filled automatically

msghelp.db

Contains object name keywords that FIDMS uses to display programmer Help.

Validity Checks
Keyfields Form Name
Page Name
Object ID

Secondary Index
Context ID (maintained, case sensitive)

Field	Field Type	Description
Form Name	A8	Name of the object's form
Page Name	A35	Name of the page containing the object
Object ID	A35	Name of the object
Message Text	A75	Help text accessed by the user
Context ID	A75	Help context ID of the object

msgusr.db

Contains object name keywords that FIDMS uses to display user Help.

Validity Checks

Keyfields
- Form Name
- Page Name
- Object ID

Secondary Index

Context ID (maintained, case sensitive)

Field	Field Type	Description
Form Name	A8	Name of the object's form
Page Name	A35	Name of the page containing the object
Object ID	A35	Name of the object
Message Text	A75	Help text accessed by the user
Context ID	A75	Help context ID of the object

persons.db

Contains details for all persons. Person is sorted by Person_no.

Validity Checks
Keyfield Person_no

Field	Field Type	Description
Person _no	I -	Internal accounting unique number for this person (read-only)
Title	A12	The person's title (e.g. Mr, Ms)
Initials	A3	The person's initials of their first names (MT for Mary Therese)
Surname	A25	The person's surname (last), for example: Doe
Forename	A20	The person's first names (e.g. Mary Therese)
Address1	A30	The address where the person lives
Address2	A30	The address where the person lives
Address3	A30	The address where the person lives
Postcode	A8	The person's postal code (include formatting)
Home_phone_no	A15	The person's home phone no (include formatting and prefix codes)
Work_phone_no	A15	The person's work phone no (include formatting and prefix codes)
Fax_no	A15	Not currently used
File_no	A15	A reference to Record File on which the person's applications are held
Drivers_licence_no	A11	Not currently used
Vehicle_reg	A10	Not currently used

picture.db

Contains .bmp pictures of commercially exploited flora. Table picture.db is sorted by Taxon ID.

Validity Checks
Keyfields Taxon Id

Field	Field Type	Description
Taxon Id and link for	S	Unique number created by WA Herbarium. Provides validation species picture data
Picture	G	Graphic picture in .bmp (windows bitmap) format (additional pictures can be added to the table when available).

postcds.db

Contains postcode details of all valid postcodes in Western Australia and the corresponding CALM District and Region. READ ONLY

Validity Checks
Keyfields Postcode

Field	Field Type	Description
Postcode	A4	Valid unique postcode
Suburb	A21	Suburb belonging to postcode
Region	A15	CALM Region corresponding to postcode
District	A15	CALM District corresponding to postcode

pri.db

Contains details of all priority flora species, including Declared Rare. Provides validation that no priority taxa have been taken illegally. Maintained by Principal Botanist, Wildlife Branch. Stored in t:\pri directory and password protected. READ ONLY

Validity Checks

Keyfields Species
Taxonid

Field	Field Type	Description
Species	A80	Keyfield not used by FIDMS application
Taxonid	N	Unique number linked to WA Herbarium database. Provides validation of FIDMS data
Change	A3	Type of change since last list produced
Date	D	Date of last significant change
Informal	A2	Name not formally approved
Authority	A40	Person(s) authorising the taxonomy and/or priority code of a taxon
Pr_code	A1	Priority code of the taxon (e.g. R, X, 1, 2, 3, 4, 5 (commercially restricted)
Threat	A3	State ranking into IUCN categories
ESP	A3	Protected under Commonwealth Endangered Species Protection Act
Calm_reg	A15	CALM region where taxon occurs
District	A20	CALM district where taxon occurs
Dist	A110	Nearest place names to population
Fl_period	A12	Time of year when taxon flowers
Old_name	A80	Any previous name the taxon was known by
Fam_no	A4	Family number as used by WA Herbarium
File_no	A6	CALM file number
ManPlan	A1	Whether a managment plan has been prepared for taxa (survey, draft, final)
Genus	A30	Genus name as used in WACENSUS
Descriptor	A70	Species, rank and infraspecies name as determined in WACENSUS
SpCode	A8	Abbreviated alphabetic code

retno.db

Used to obtain unique retnum values by the retnum.db table. Not accessed directly by the user.

Field	Field Type	Description
Retnum	N	New unique number assigned to the retnum.db table when new return is selected on menu.

retnum.db

Contains details of flora returns submitted by licences. Table retnum.db is sorted by Retnum.

Validity Checks
Keyfields Retnum

Secondary Index
Licence_number

Field	Field Type	Description
Retnum	N	Unique number automatically created when new return is selected on menu. Links the retnum.db and retex.db tables.
Licence_number	A8 -	Unique number taken from Licences table. Values will only be accepted if the licence number is present in the licences.db table.
Collecting_date	D	Returns are due on 15th day of each month. Format as 15/__/199_
Days_collecting	N	A number of days between 0 (nil return) and 31 (maximum no. of days)

retex.db

Contains details of flora harvested for each flora return submitted by licences Licence.db is sorted by Retnum.

Validity Checks

Keyfields
- Retnum
- Taxonid
- Quantity
- Part
- Crown_private
- Grid square
- Dealer_1

Referential Integrity
- grid

Field	Field Type	Description
Retnum	N	Unique number automatically created when new return is selected on menu. Links the retnum.db and retex.db tables.
TaxonID	N -	Unique number used by Herbarium WACENSUS database (current only valid)
Quantity	N	Amount of flora harvested. Should be converted if number of bunches is given
Part	A15	Part of flora taken. Drop down list allows you to choose a valid value
Crown_private	A1	Land status of area from which flora was harvested. Valid values are C (Crown), P (private property natural stand), A (artificially propagated)
Grid square	I	Corresponds to grid squares shown on reverse of flora return form. Valid values are enforced by link to geograph.db table.
Dealer_1	A26	Corresponds to first dealer flora is supplied to (cross-referenced with Dealer number - when available)
Genus	A30	Genus name is linked to WACENSUS (filled automatically)
Species	A30	Species name is linked to WACENSUS (filled automatically)
InfraSpRank	A9	InfraspRank name is linked to WACENSUS (filled automatically)
InfraSpName	A30	InfraspName is linked to WACENSUS (filled automatically)
Unit	A10	E.g. single, kg., etc. Drop down list allows you to choose a valid value
Owner/Company Name	A35	Identifies private property owner, forest block name, reserve no.
Dealer_2	A25	Corresponds to 2nd dealer flora is supplied to (cross-referenced with dealer number - when available)
Dealer_3	A25	Corresponds to 3rd dealer flora is supplied to (cross-referenced with dealer number when available)
Dealer_4	A25	Corresponds to 4th dealer flora is supplied to (cross-referenced with dealer number - when available)
Dealer_5	A25	Corresponds to 5th dealer flora is supplied to (cross-referenced with dealer number when available)

vehicle.db

Contains details of vehicles for each endorsement issued by Region/District Offices for valid current licence holders. Table vehicle.db is sorted by End_no and Vehicle_reg.

Validity Checks

Keyfields End_no
Vehicle_reg

Field	Field Type	Description
End_no	N	Unique number automatically created when new endorsement is selected on menu. Links the endloc.db, endspec.db and vehicle.db tables.
Vehicle_reg	A8 -	Vehicle registration number of picker's vehicle.
Colour	A10	Colour of picker's vehicle (assists in field identification)
Vehicle Make	A10	Manufacturer of vehicle (e.g. Ford, Mazda)
Vehicle Model	A15	The model of the vehicle, e.g Laser, Landcruiser
Vehicle Type	A10	The sort of vehicle , e.g. sedan, 4WD, tray-top

Temporary Tables

cur_lic.db

This table is used by the Report Form (REPORT.FSL) to hold data produced by the Reports|Licences query option. It shows licences by postcode for various options, including by CALM region, CALM district, current licences or historical licences..

Field	Field Type	Description
Surname	A25	The person's surname (last), for example: Doe
Initials	A3	The person's initials of their first names (MT for Mary Therese)
Address1	A30	The address where the person lives
Address2	A30	The address where the person lives
Address3	A30	The address where the person lives
Postcode	A8	The person's postal code (include formatting)
Licence_no	A8	Unique number issued annually
Valid_from	D	First valid date of the licence
Expiry_date	D	Last valid date of the licence

endprn.db

This table is used by the CALM Endorsement Form (ENDORSFM.FSL) to hold details of a user-specified endorsement when the print option is chosen.

Field	Field Type	Description
Surname	A25	The person's surname (last), for example: Doe
Title	A12	The person's title (e.g. Mr, Ms)
Forename	A20	The person's first names (e.g. Mary Therese)
Address1	A30	The address where the person lives
Address2	A30	The address where the person lives
Address3	A30	The address where the person lives
Postcode	A8	The person's postal code (include formatting)
Licence_no	A8	Unique number issued annually
Endorsement_no	N	Unique number which validates and links other endorsement information
Issue_officer	A45	Name of CALM officer authorised to issue endorsement
Region	A20	CALM Region - selected using drop-down list
District	A20	CALM District - selected using drop-down list
Endorse_from	D	First day of endorsement - must be within valid period of licence
Endorse_to	D	Last day of endorsement - no longer than 3 months

endrep.db

This table is used by the Report Form (REPORT.FSL) to hold data produced by the Reports|Endorsement query option when the species option is not chosen. It shows details of endorsements by district or region.

Field	Field Type	Description
Licence_no	A8	Unique number issued annually
Endorsement_no	N	Unique number which validates and links other endorsement information
Issue_officer	A45	Name of CALM officer authorised to issue endorsement
Region	A20	CALM Region - selected using drop-down list
District	A20	CALM District - selected using drop-down list
Endorse_to	D	Last day of endorsement - no longer than 3 months
Location name	A25	Usually a forest block
Location description	A50	Description of area bounded by roads, rivers, etc.

endtmp.db

This table is used by the Report Form (REPORT.FSL) to hold data produced by the Reports|Endorsement query option when the species option is not chosen. It shows details of vehicles held by endorsees by district or region.

Field	Field Type	Description
Licence_no	A8	Unique number issued annually
Endorsement_no	N	Unique number which validates and links other endorsement information
Issue_officer	A45	Name of CALM officer authorised to issue endorsement
Region	A20	CALM Region - selected using drop-down list
District	A20	CALM District - selected using drop-down list
Endorse_to	D	Last day of endorsement - no longer than 3 months
Location name	A25	Usually a forest block
Location description	A50	Description of area bounded by roads, rivers, etc.
Vehicle_reg	A8 -	Vehicle registration number of picker's vehicle.
Colour	A10	Colour of picker's vehicle (assists in field identification)
Vehicle Make	A10	Manufacturer of vehicle (e.g. Ford, Mazda)
Vehicle Model	A15	The model of the vehicle, e.g Laser, Landcruiser
Vehicle Type	A10	The sort of vehicle , e.g. sedan, 4WD, tray-top

endtmpsp.db

This table is used by the Report Form (REPORT.FSL) to hold data produced by the Reports|Endorsement query option when the species option is chosen. It shows details of species endorsement in each block by district or region.

Field	Field Type	Description
Region or District	A20	CALM Region or District (depending on option chosen)
Endorse_to	D	Last day of endorsement - no longer than 3 months
Location name	A25	Usually a forest block
Genus	A30	Permitted genus; linked to WACENSUS (filled automatically)
Species	A30	Permitted species;linked to WACENSUS (filled automatically)
Unit	A10	Should be either single or kg (NOT bunches)
Part	A10	The product permitted to be harvested
Sum of Quantity	N	The total amount of the species which is endorsed to be harvested within the block

lic_stat.db

The lic_stat.db table is used by the GEN LIC Dialog box when the user chooses the Locate|Licence Number query option. It is used to show the results of a licence search of a user-defined licence number in the Person and Licensing Form mainPage find_lic method.

Field	Field Type	Description
Person _no	I -	Internal accounting unique number for this person (read-only)
Surname	A25	The person's surname (last), for example: Doe
Initials	A3	The person's initials of their first names (MT for Mary Therese)
Forename	A20	The person's first names (e.g. Mary Therese)
Address1	A30	The address where the person lives
Address2	A30	The address where the person lives
Address3	A30	The address where the person lives
Postcode	A8	The person's postal code (include formatting)
Home_phone_no	A15	The person's home phone no (incl. formatting and prefix codes)
Work_phone_no	A15	The person's work phone no (incl. formatting and prefix codes)
Licence_no	A8	Unique number issued annually
Valid_from	D	First valid date of the licence
Expiry_date	D	Last valid date of the licence
Location	A60	Details of approved land area

licloc.db

The licloc.db table is used by the LICLOC Dialog box is opened when a date which is not valid for the active licence number is entered. A query is performed which shows all licences held by the person and their valid period so the user can choose the correct licence number.

Field	Field Type	Description
Person _no	I -	Internal accounting unique number for this person
Licence_no	A8	Unique number issued annually
Valid_from	D	First valid date of the licence
Expiry_date	D	Last valid date of the licence

licprn.db

The licprn.db table is used by the Flora Licence Report when the user chooses the Print button option in the Person and Licensing Form. It is used to show the results of a licence search of a user-defined licence number in the Person and Licensing Form Print button method.

Field	Field Type	Description
Surname	A25	The person's surname (last), for example: Doe
Title	A12	The person's title, e.g. Mr, Ms
Forename	A20	The person's first names (e.g. Mary Therese)
Address1	A30	The address where the person lives
Address2	A30	The address where the person lives
Address3	A30	The address where the person lives
Postcode	A8	The person's postal code (include formatting)
Licence_no	A8	Unique number issued annually
Valid_from	D	First valid date of the licence
Expiry_date	D	Last valid date of the licence

norenew.db

This table is used by the Report Form (REPORT.FSL) to hold data produced by the Reports|Renew Letters query option. It produces names and addresses of licensees whose licences are due to expire within a user-defined period and who have not submitted flora returns in accordance with licence conditions. The table is used by the Reports|Print|Renew letters and Renew list menu options to produce letters to be sent out each month.

Field	Field Type	Description
Surname	A25	The person's surname (last), for example: Doe
Title	A12	The person's title, e.g. Mr, Ms
Forename	A20	The person's first names (e.g. Mary Therese)
Address1	A30	The address where the person lives
Address2	A30	The address where the person lives
Address3	A30	The address where the person lives
Postcode	A8	The person's postal code (include formatting)
Licence_no	A8	Unique number issued annually
Expiry_date	D	Last valid date of the licence

objlist.db

This table holds the results of a query (pageObjsQBE) which runs as part of the listPageObjects method in the FIDMS library to find all value setting objects on the current form & page.

Field	Field Type	Description
ObjectPath	A100	Holds the values of objects on the current form and page

renew.db

This table is used by the Report Form (REPORT.FSL) to hold data produced by the Reports|Renew Letters query option. It produces names and addresses of licensees whose licences are due to expire within a user-defined period and who have submitted required flora returns in accordance with licence conditions. The table is used by the Reports|Print|Renew letters and Renew list menu options to produce letters to be sent out each month.

Field	Field Type	Description
Surname	A25	The person's surname (last), for example: Doe
Title	A12	The person's title, e.g. Mr, Ms
Forename	A20	The person's first names (e.g. Mary Therese)
Address1	A30	The address where the person lives
Address2	A30	The address where the person lives
Address3	A30	The address where the person lives
Postcode	A8	The person's postal code (include formatting)
Licence_no	A8	Unique number issued annually
Expiry_date	D	Last valid date of the licence

retprn.db

This table is used by the Flora Industry Return Form (FLORARET.FSL) to hold the results of a query generated by the Print pushbutton for return details of a user-defined licence number. The table is used by the floraret.rsl report to show the results.

Field	Field Type	Description
Surname	A25	The person's surname (last), for example: Doe
Title	A12	The person's title (e.g. Mr, Ms)
Forename	A20	The person's first names (e.g. Mary Therese)
Address1	A30	The address where the person lives
Address2	A30	The address where the person lives
Address3	A30	The address where the person lives
Postcode	A8	The person's postal code (include formatting)
Licence_no	A8	Unique number issued annually
Collecting_date	D	Date that the flora was harvested (15/mm/yy)
TaxonID	N -	Unique number used by Herbarium WACENSUS database (current only valid)
Genus	A30	Genus name is linked to WACENSUS
Species	A30	Species name is linked to WACENSUS
Quantity	N	Amount of flora harvested. Should be converted if number of bunches is given
Unit	A10	E.g. single, kg., etc.
Part	A15	Part of flora taken.
Grid square	I	Corresponds to grid squares shown on reverse of flora return form.

retstat.db

The retstat.db table is used by the RetStat Dialog box when the user requests the return status for a specified licence number. This data is required before issuing a licence.

Field	Field Type	Description
Surname	A25	The person's surname (last), for example: Doe
Title	A12	The person's title, e.g. Mr, Ms
Forename	A20	The person's first names (e.g. Mary There
Licence_no	A8	Unique number issued annually
Collecting_date	D	Returns are due on 15th day of each month. Format as 15/__/199_

speccnt.db

This table is used by the Statistics Form (STATS.FSL) to hold the results of queries for the number of species exploited for either stems or seed in all years. It is not a permanent table because data is continually updated and entered.

Validity Checks
Keyfields TaxonId

Field	Field Type	Description
Yr	N	Year that the data applies to
Part	A15	Either seed or stems validation of
Count of TaxonId	N	Number of species harvested in each category for given year

special.db

This table is used by the Report Form (REPORT.FSL) to hold data produced by the Reports|Returns query option. It shows details of species picked in a district, region or picking region.

Field	Field Type	Description
Genus	A30	Permitted genus; linked to WACENSUS (filled automatically)
Species	A30	Permitted species;linked to WACENSUS (filled automatically)
Unit	A10	Should be either single or kg (NOT bunches)
Part	A10	The product permitted to be harvested
Crown_private	A1	Land status of area from which flora was harvested. Valid values are C (Crown), P (private property natural stand), A (artificially propagated)
Grid square	I	Corresponds to grid squares shown on reverse of flora return form.
Owner/Company Name	A25	Identifies private property owner, forest block name, reserve no. etc.
Collecting Date	D	Date that collection took place (as 15/mm/yy)
Region	A20	CALM region
District	A20	CALM District (depending on option chosen)
Picking Region	A20	Biogeographic region
Sum of Quantity	N	The total amount of the species which is endorsed to be harvested

sploc.db

The sploc.db table is used by the Species Locate Dialog box (SPLOC.FSL) when the user requests the taxonid for an unknown taxon.

Field	Field Type	Description
Taxon Id	N	Unique number created by WA Herbarium.
Genus	A30	Genus name as determined by WA Herbarium
Species	A60	Species name as determined by WA Herbarium
InfraSpRank	A9	InfraspRank name as determined by WA Herbarium
InfraSpName	A70	InfraspName name as determined by WA Herbarium
Curnt	A1	Indicates whether a taxon id and name are valid.

tempsrc.db

The tempsrc.db table is used by the prog_dev method in the Programmer/Developer Form (PROG_DEV.FSL) to list all the source code for a user-defined form (including all objects, methods and pages).

Field	Field Type	Description
Object	A128	Name of the ObjectPal button, field, page, etc. within the nominated form
MethodName	A128	Name of method (either built-in or custom)
Source	M64	Listing of all code for an object or method

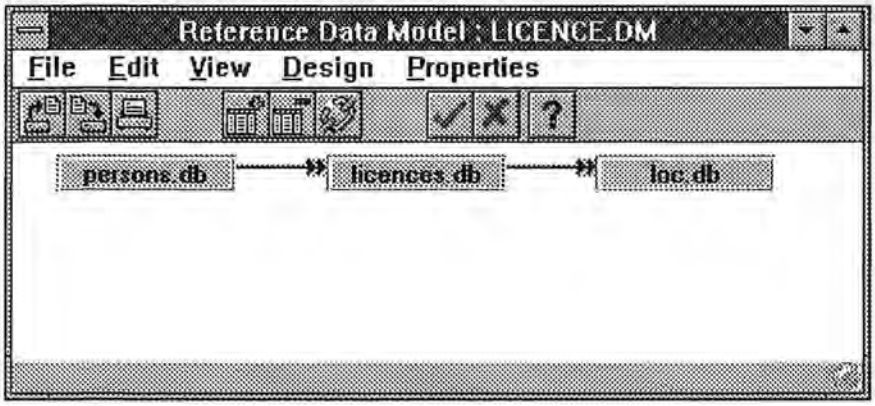
Data models

A data model in Paradox is a diagram of table relationships. Data models exist in two ways:

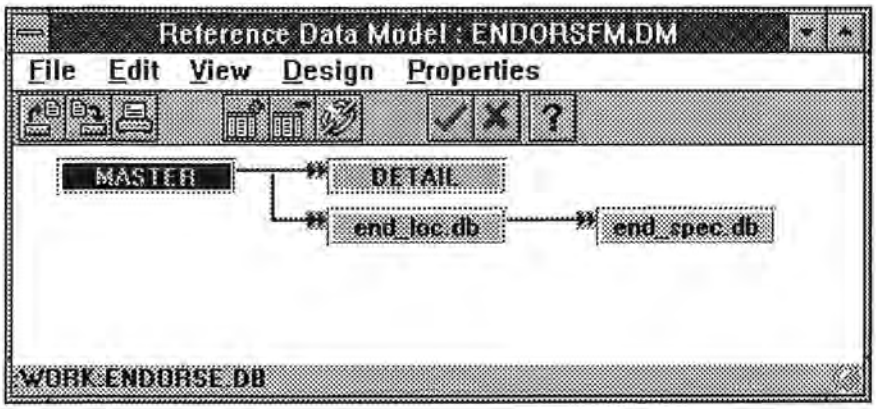
- As part of a a design document. Using this kind of data model, you can bind tables to documents and specify how they are linked to each other.
- As a separate file. This kind of data model, known as a reference data model, can be used to modify the data model of a design document. Data model reference files have a file extension of .DM.

The following are the data models in FIDMS shown. Each is shown as a diagram of table relationships.

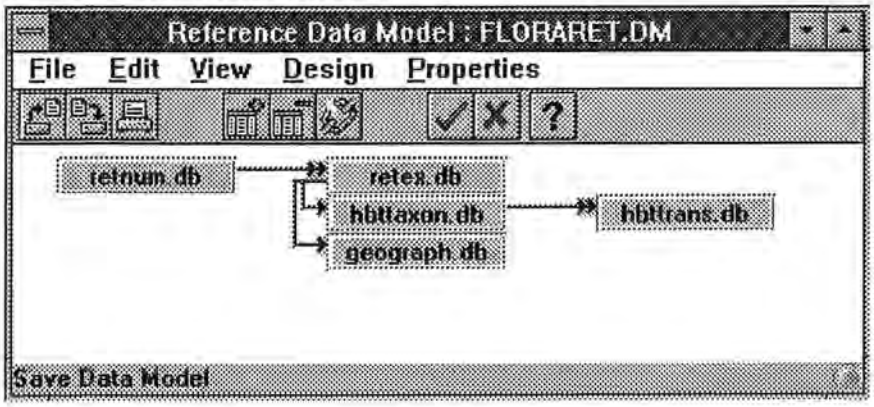
Person and Licensing Form Data Model



EndorsFM Form Data Model



Floraret Form Data Model



Programmer's Code

ObjectPAL is an event-driven, object-oriented programming language, different from a traditional procedural language where you create a file of commands that execute one after another. The following sections list areas where Paradox stores code modules (called methods).

Forms
The user interface displayed in FIDMS, including dialog boxes, consists of a series of forms. Using ObjectPAL, you place design objects (for example, buttons and fields) in a form and attach methods that execute when something happens to the object. Go to a topic for information on a form and its code. (You can also display this information from within the application by right-clicking an object and choosing Code Help.)

- **ABTFIDMS Dialog Box (ABTFIDMS.FSL) (see p 38)**
- **FIDMS Module Manager Form (FLORADB.FSL) (see p 40)**
- **CALM Endorsement Form (ENDORSFM.FSL) (see p 44)**
- **Endorsements by Region/District Dialog Box (END_REP.FSL) (see p 59)**
- **Endorsements by Species Dialog Box (END_SPEC.FSL) (see p 67)**
- **Licence Locate Dialog Box (GEN_LIC.FSL) (see p 61)**
- **Licence Update Dialog Box (LICLOC.FSL) (see p 65)**
- **Flora Industry Return Form (FLORARET.FSL) (see p 49)**
- **Person and Licensing Form (LICENCE.FSL) (see p 35)**
- **Programmer/Developer Form (PROG_DEV.FSL) (see p 55)**
- **Report Form (REPORT.FSL) (see p 5251)**
- **Return Status Dialog Box (RETSTAT.FSL) (see p 63)**
- **Species Locate Dialog Box (SPLOC.FSL) (see p 57)**
- **Statistics Form (STATS.FSL) (see p 53)**

Scripts
A script consists of code in its own file, not attached to a form. It's an object, and displays on the Desktop as an icon. Use a script when you want to execute code without opening and displaying a form window. FIDMS can use START.SSL to start the application

START.SSL

START.SSL lets the user start FIDMS.

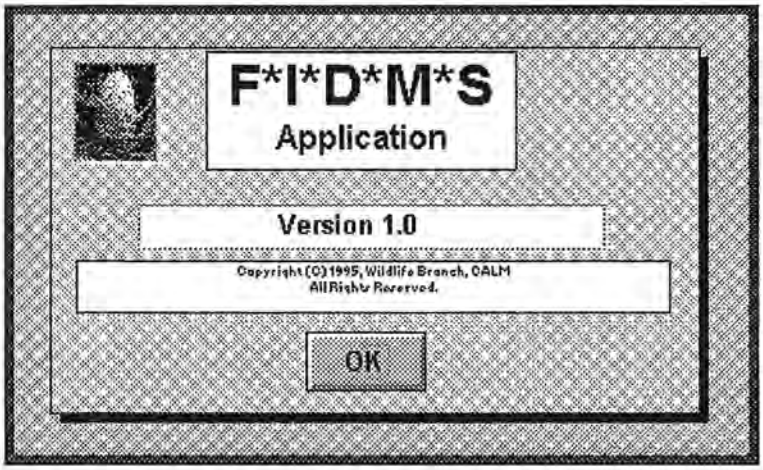
run

Called when the user runs START.SSL. It changes the title of the window to FIDMS, loads FIDMS Module Manager Form (FLORADB.FSL) and hides the Toolbar.

```
; This method starts FIDMS
method run(var eventInfo Event)
var
  projForm,formToOpen form
  app       Application
  lib       Library
  tcSystem        tCursor
  tblSystem       string
endVar

  ; Set the Paradox window's title
  app.setTitle("FIDMS")

  hideToolBar()

  if projectViewerIsOpen() then
     ; If the Project View is open, close it
     projectViewerClose()
  endif

  ; Display the FloraDB Form
  formToOpen.open("FloraDB")

endmethod
```

ABTFIDMS Dialog Box

ABTFIDMS displays information about the FIDMS program. It is a dialog box and can only be accessed through other forms. The form appears as shown below (but in colour).



ABTFIDMS.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Methods (ABTFIDMS.FSL)	**Objects**
Uses	abtPg
Var	
Const	
mouseRightUp	
open	
Objects (abtPg)	
Button16	

Full listings of the code follows.

Object : AbtFIDMS

MethodName : Var

Source :
```
Var
  callerFm Form
        uio UIObject
        formName Form
        fidmsLib Library
        wasCalled Logical
endVar
```

Object : AbtFIDMS

MethodName : Uses

Source :
```
Uses ObjectPal
        createObjMenu(var formName Form,
          pageName String, uio UIObject)
        pageNameOf(ui UIObject) String
        returnIsCalled() Logical
endUses
```

Object : AbtFIDMS

MethodName : Const

Source :
```
Const
  helpfile = ":WORK:fidmsuser.hlp"
endConst
```

Object : AbtFIDMS

MethodName : close

Source :
```
method close(var eventInfo Event)
if eventInfo.isPrefilter() then
        doDefault
else
        if floradbFm.isAssigned() then
                floradbFm.close()
        else
                if floraretFm.isAssigned() then
                        floraretFm.close()
                endif
                if endorseFm.isAssigned() then
                        endorseFm.close()
                endif
                if licenceFm.isAssigned() then
                        licenceFm.close()
                endif
                if reportsFm.isAssigned() then
                        reportsFM.close()
                endif
                FIDMSlib.formCanClose()
        endif
        showSpeedBar()
        removeMenu()
        helpQuit(":fidms:floradb.hlp")
        helpquit(":fidms:fidprog.hlp")
endif
endmethod
```

Object : AbtFIDMS

MethodName : mouseRightUp

Source :
```
method mouseRightUp(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio) ; what is the object?
        if uio.name <> "diveFlagBox" then
                while uio.class = "Text" or uio.class = "Bitmap"
                        uio.attach(uio.ContainerName)
```

```
        endif
        fidmsLib.createObjMenu(formName, fidmslib.pageNameOf(uio), uio)
endif
endmethod
```

Object : AbtFIDMS

MethodName : open

Source :

```
method open(var eventInfo Event)
if not eventInfo.isPreFilter() then
        if not fidmsLib.open("fidmsLib.ldl", globalToDesktop) then
                msgStop("Failure", "fidmslib could not be opened.")
                close()
                disableDefault
        else
                wasCalled = fidmsLib.returnIsCalled()
                if wasCalled = False then
                        msgStop("Stop!", "You cannot open this form directly.
                          It must be opened from floradb.fsl.")
                        close()
                        disableDefault
                else
                        formName.attach()
                endif
        endif
endif
endmethod
```

Object : AbtPg

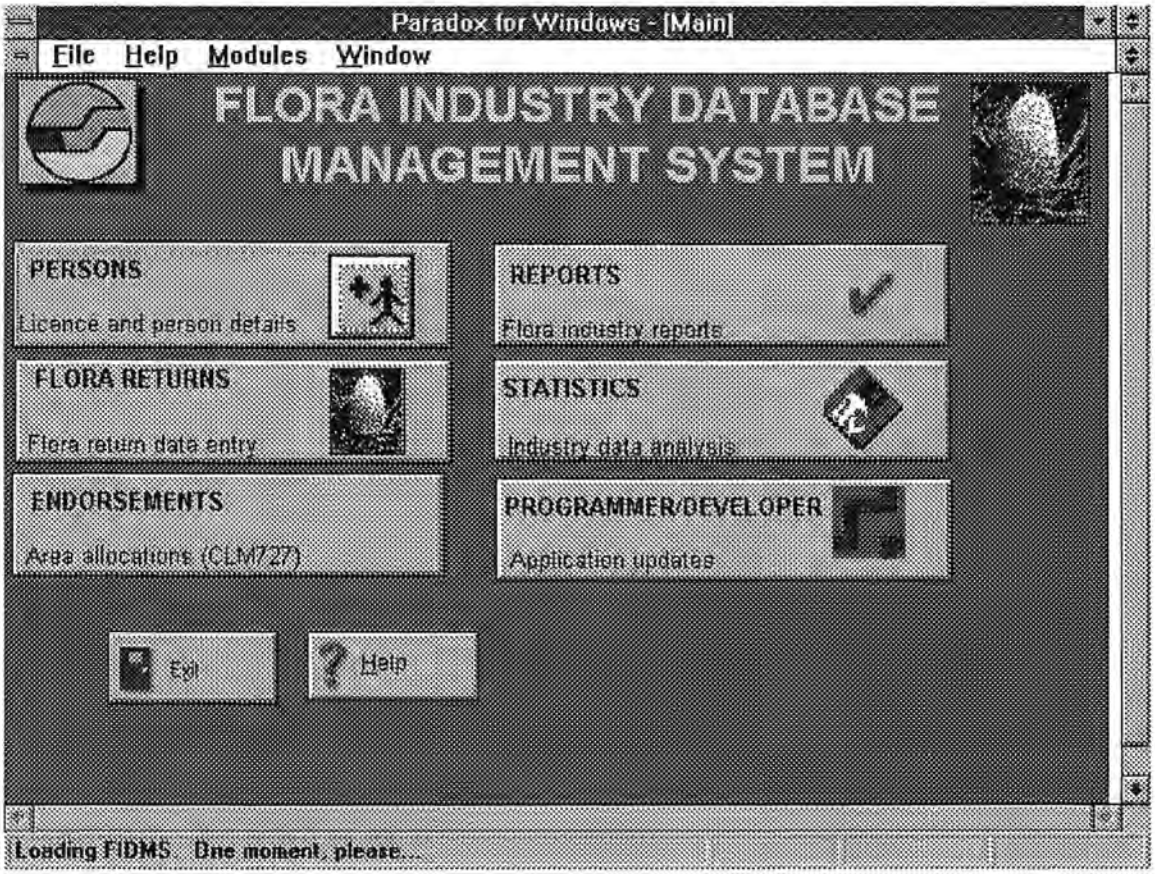
MethodName : AbtBttn

Source :

```
method pushButton(var eventInfo Event)
        formReturn()
endmethod
```

FIDMS Module Manager Form (FLORADB.FSL)

FIDMS Module Manager controls several independent forms and is the main form for FIDMS. From here you display the other forms. The form appears as shown below (but in colour).



FLORADB.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Methods (FLORADB.FSL)	**Objects**
Uses	mainPage
Var	
Const	
close.	
keyPhysical	
menuAction	
mouseEnter	
mouseExit	
mouseRightup	

Methods (mainPage)	**Objects (mainPage)**
Uses	exitButton
Var	helpButton
Const	endorsebttn
	floraretbutton
menuAction	licencebutton
open	processButton
	prog_dev
	reportbutton

Full listings of the code follows.

Object : Floradb

MethodName : Uses

Source :
```
Uses ObjectPal

  clearPageValues(formName String, pageName String)
  createObjMenu(var formName Form,
          pageName String, uio UIObject)

  FIDMSObjMsg(formName String, pageName String, objID String)
  pageNameOf(ui UIObject) String
  setIsCalledTrue()
  setIsCalledFalse()
  formCanClose()

endUses
```

Object : Floradb

MethodName : Var

Source :
```
Var
uioName, choice String
  uio UIObject
  formName, floradbFm, floraretFm, licenceFm, reportsFM, endorseFM, aboutFm, statsFM, progFM Form
  fidmslib Library
  closeVCRLg Logical
  okToExit Logical

endVar
```

Object : Floradb

MethodName : Const

Source :
```
Const
  helpFile  = ":fidms:floradb.hlp" ; path to Help file
  headerHelp = LongInt(20006)
  detailHelp = LongInt(1)
  proghelp  = ":fidms:fidprog.hlp" ; path to programmer help file
endConst
```

Object : Floradb

MethodName : close

Source :
```
method close(var eventInfo Event)

if eventInfo.isPrefilter() then
        doDefault
else
        if floradbFm.isAssigned() then
                floradbFm.close()
        else
                if floraretFm.isAssigned() then
                        floraretFm.close()
                endif
                if endorseFm.isAssigned() then
                        endorseFm.close()
                endif
                if licenceFm.isAssigned() then
                        licenceFm.close()
                endif
                if reportsFm.isAssigned() then
                        reportsFM.close()
                endif
                FIDMSlib.formCanClose()
        endif
        showSpeedBar()
        removeMenu()
```

```
        helpquit(":fidms:fidprog.hlp")
endif

endmethod
```

Object : Floradb

MethodName : mouseEnter

Source :

```
method mouseEnter(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio)  ; what is the object?
        uioName = uio.Name
        if uio.name <> self.name AND (uio.class = "Field" OR uio.class = "Multirecord"
          OR uio.class = "Button") then
                fidmslib.FIDMSObjMsg(formName.name, fidmslib.pageNameOf(uio), uioName)
        endif
endif
endmethod
```

Object : Floradb

MethodName : mouseExit

Source :

```
method mouseExit(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio)
        if uio.class <> "Text" and uio.class <> "Bitmap" then
                message("")
        endif
endif
endmethod
```

Object : Floradb

MethodName : mouseRightUp

Source :

```
method mouseRightUp(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio) ; what is the object?
        if uio.name <> "calmlogoBox" then
                while uio.class = "Text" or uio.class = "Bitmap"
                        uio.attach(uio.ContainerName)
        endWhile
        endif
        fidmslib.createObjMenu(formName, fidmslib.pageNameOf(uio), uio)
endif
endmethod
```

Object : Floradb

MethodName : keyPhysical

Source :

```
method keyPhysical(var eventInfo KeyEvent)
var
 theKey String
 thePage String
endvar

if eventInfo.isPreFilter() then
        theKey = eventInfo.vChar()  ; tell me what key was pressed
        if eventInfo.isAltKeyDown() then
                switch
                        case theKey = "E" or theKey = "e" : disableDefault
                        mainpage.endorsebttn.pushbutton()
                        case theKey = "P" or theKey = "p" : disableDefault
                        mainPage.processButton.pushButton()
                        case theKey = "F" or theKey = "f" : disableDefault
                        mainPage.floraretButton.pushButton()
                        case theKey = "R" or theKey = "r" : disableDefault
                        mainPage.reportprocessbutton.pushbutton()
                        case theKey = "X" or theKey = "x" : disableDefault
                        mainPage.exitButton.pushButton()
```

```
                endswitch
        else
                switch
                        case theKey = "VK_F1" : disableDefault
                helpShowContext(helpfile, detailhelp)
                otherwise : doDefault
                endSwitch
        endif
endif
endmethod
```

Object : Floradb

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
if not eventInfo.isPreFilter() then
        if eventInfo.reason() = menuNormal and eventInfo.id() = MenuCanClose and not okToExit then
                eventInfo.setErrorCode(CanNotDepart)
        disableDefault
        message("Use the Exit button to exit FLORADB.")
        endIf
endIf
endmethod
```

Object : MainPage

MethodName : Uses

Source :

```
Uses ObjectPal
        setCallFormName(const CallerTitle String, xPos LongInt, yPos LongInt)
        VCRisOpen()
        VCRisClosed()
        isVCROpen() Logical
        vcrName() String
        listPageObjects(formName String, pageName String) Logical
endUses
```

Object : MainPage

MethodName : Var

Source :

```
Var
        wtext,maintext array[12] String
endVar
```

Object : MainPage

MethodName : Const

Source :

```
Const
        #IDH_MENUS_WINDOW_CURRENTWINDOW = "Current Open Windows"
        #IDH_MENUS_WINDOWS = "&Window"

        HelpHelp = 101 ; Help | Using Help command
        HelpList = 102 ; Help | Help Index command
        Help_Use = 103 ; Help | System Help command
        Prog_help = 104 ; Help | Programmer Help

        Florarets = 201 ;Modules | Flora Returns command
        Licences = 202 ; Modules | Licences command
        Stats    = 202 ; Modules | Statistics command
        Reports  = 203 ; Modules | Reports command
        Endorse  = 204 ; Modules | Endorsements command
        progdev  = 205 ; Modules | Programmer/Developer

        file_print = 300
        Exit     = 301 ; Exit command
        file_print_setup = 302
endConst
```

Object : MainPage

Source :

```
method open(var eventInfo Event)
delayScreenUpdates(Yes)
if not isFile("floradb.fdl") then
        msgInfo("Startup Error!", "The FIDMS application files must be " +
        "in the working directory.")
        close()
endif
if not fidmslib.open("fidmslib.ldl", globalToDesktop) then
        msgStop("Failure", "fidmslib could not be opened")
endif
wText[1]=#IDH_MENUS_WINDOW_CURRENTWINDOW
mainText[5]=#IDH_MENUS_WINDOWS
formName.attach()
hideSpeedBar()
maximize()
okToExit = False
delayScreenUpdates(No)
Message("Loading FIDMS.  One moment, please...")
delayScreenUpdates(Yes)
if not isFile("floradb.fdl") then
        disableDefault
        beep()
        msgStop("Directory Error", "The Paradox for Windows " +
        "working directory must be set to the directory " +
        "containing this form, for example: " +
        "T:\\fidms\\")
        close()
        blankAsZero(No)
        fidmslib.open("fidmslib.fdl", globalToDesktop)
        okToExit = False
        doDefault
        self.postAction(dataBeginEdit)
        delayScreenUpdates(No)
endIf
endmethod
```

Object : MainPage

MethodName : arrive

Source :

```
method arrive(var eventInfo MoveEvent)
var
        mainMenu Menu      ; The main menu
        filemenu,
        ModuleMenu,        ; The Module menu
        HelpMenu,
        windowMenu         popupmenu   ; The Help menu
endVar

        fileMenu.addtext ("&Print", Menuenabled, Usermenu + file_print)
        fileMenu.addtext ("Printer &Setup", MenuEnabled, Usermenu + File_print_setup)
        filemenu.addtext ("E&xit", menuEnabled, userMenu + exit)
        mainmenu.addpopup ("&File", filemenu)

        HelpMenu.addText("Using &Help\tCtrl+F1", menuEnabled, userMenu + helpHelp)
        HelpMenu.addText("Help &Index\tShift+F1", menuEnabled, userMenu + helpList)
        HelpMenu.addText("&Using this Form\tF1", menuEnabled, userMenu + help_Use)
        HelpMenu.addText("&Programmer Help", menuEnabled, userMenu + Prog_help)
        MainMenu.addPopUp("&Help", HelpMenu)

        ModuleMenu.addtext ("&Flora Returns", menuEnabled, userMenu + Florarets)
        ModuleMenu.addtext ("&Statistics", menuEnabled, userMenu + Stats)
        ModuleMenu.addtext ("&Reports", menuEnabled, userMenu + Reports)
        ModuleMenu.addtext ("&Licences", menuEnabled, userMenu + Licences)
        ModuleMenu.addtext ("&Endorsements", menuEnabled, UserMenu + Endorse)
        ModuleMenu.addtext ("&Programmer/Developer", menuEnabled, userMenu + progdev)
        MainMenu.addpopup("&Modules",ModuleMenu)

        windowmenu.addStaticText(wText[1])
        mainMenu.addPopUp(mainText[5],windowmenu)
```

```
        setTitle("Main")
endmethod
```

Object : MainPage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
var
        menu_Cmd        smallInt ; The constant returned by a menu command
        editTest        Logical  ; Flag variable used to test the type of object that is active
        calledBy        Form     ; Holds the handle of the form that opened this form
endVar

menu_Cmd = eventInfo.id()
switch
        case menu_Cmd = MenuControlClose : disableDefault
        exitbutton.pushButton()
        case menu_Cmd = MenuControlMaximize or menu_Cmd = MenuControlMinimize :
        if formcaller(calledBy) then
                disableDefault
                beep()
                msgInfo("Oops!", "Because this form was opened by the " +
                  calledBy.getTitle() + " form, the Maximize and " +
                  "Minimize buttons are disabled.  Sorry...")
        endIf
        case menu_Cmd = userMenu + Florarets : Floraretbutton.pushbutton()
        case menu_Cmd = userMenu + Licences : Licencebutton.pushbutton()
        case menu_Cmd = userMenu + Stats        : processButton.pushbutton()
        case menu_Cmd = userMenu + Reports  : reportprocessbutton.pushbutton()
        case menu_Cmd = userMenu + Endorse  : endorsebttn.pushbutton()
        case menu_Cmd = userMenu + progdev  : prog_dev.pushbutton()
        case menu_Cmd = userMenu + Exit         : exitButton.pushbutton()
        case menu_Cmd = userMenu + HelpHelp : helpOnHelp()
        case menu_Cmd = userMenu + Prog_help  : helpShowindex(progHelp)
        case menu_Cmd = userMenu + HelpList : helpshowContext(helpfile,1000)
        case menu_Cmd = userMenu + Help_Use : helpShowContext(helpFile, detailHelp)
        case menu_cmd = usermenu + file_print: active.menuAction(menuFilePrint)
        case menu_cmd = usermenu + file_print_setup: active.menuAction(menuFilePrinterSetup)
endswitch
endmethod
```

Object : MainPage.prog_dev

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(progFM) then
        progFM.bringtoTop()
        progFM.wait()  ; put progFM into wait state
        progFM.hide()
        maximize()
        mainpage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if progFM.Open("prog_dev.fdl", WinStyleMaximize) then
                progFM.Wait()  ; put progFM into wait state
                progFM.hide()
                maximize()
                mainpage.moveTo()
        else
                msgInfo("Status", "Sorry, the Programmer/Developer form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : MainPage.endorsebttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
```

```
if isAssigned(endorseFm) then
        endorseFm.bringtoTop()
        endorseFm.wait()  ; put endorseFm into wait state
        endorseFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if endorseFm.Open("endorsfm.fdl", WinStyleMaximize) then
                endorseFm.Wait()  ; put endorseFm into wait state
                endorseFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Endorsement form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : MainPage.reportprocessbutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(reportsFm) then
        reportsFm.bringtoTop()
        reportsFm.wait()  ; put reportsFm into wait state
        reportsFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if reportsFm.Open("report.fdl", WinStyleMaximize) then
                reportsFm.Wait()  ; put reportsFm into wait state
                reportsFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Reports form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : MainPage.helpButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        helpShowIndex(helpFile)
endMethod
```

Object : MainPage.processButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(statsFM) then
        statsFM.bringtoTop()
        statsFM.wait()  ; put statsFM into wait state
        statsFM.hide()
        maximize()
        mainpage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if statsFM.Open("Stats.fdl", WinStyleMaximize) then
                statsFM.Wait()  ; put statsFM into wait state
                statsFM.hide()
                maximize()
                mainpage.moveTo()
```

```
        else
                msgInfo("Status", "Sorry, the Statistics form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : MainPage.exitButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)

choice = msgQuestion("Quit F*I*D*B*M*S*", "Do you want to quit " +  "the application?")
if choice= "Yes" then
        okToExit = True
        close()
else
        eventInfo.setErrorCode(cannotDepart)
endif
endmethod
```

Object : MainPage.licencebutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(licenceFM) then
        licenceFm.bringtoTop()
        licenceFm.wait()  ; put licence into wait state
        licenceFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if licenceFM.Open("licence.fdl", WinStyleMaximize) then
                licenceFm.Wait()  ; put licence into wait state
                licenceFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the licence form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : MainPage.Floraretbutton

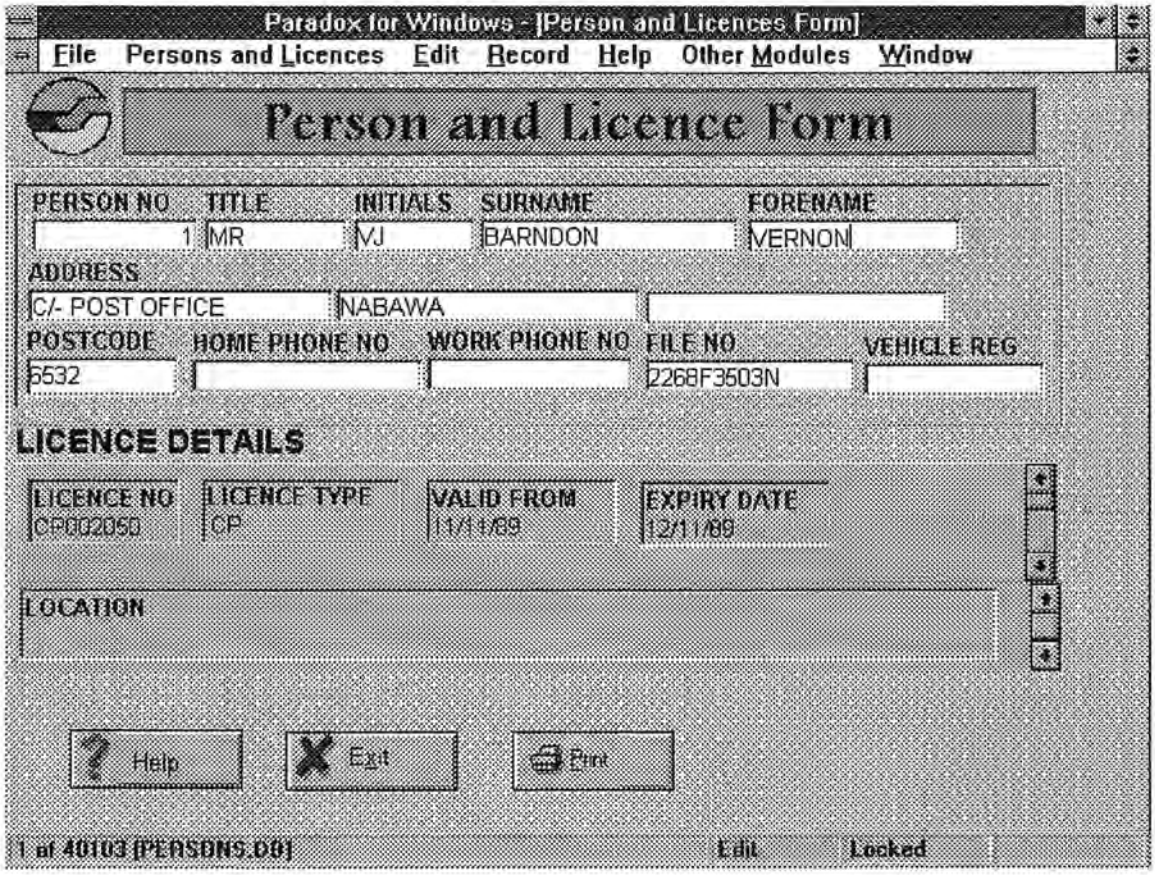
MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(floraretFm) then
        floraretFm.bringtoTop()
        floraretFm.wait()  ; put floraretFm into wait state
        floraretFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmsLib.setIsCalledTrue()
        if floraretFm.Open("floraret.fdl", WinStyleMaximize) then
                floraretFm.Wait()  ; put floraretFm into wait state
                floraretFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Flora Return form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Person and Licensing Form

The Person and Licensing Form is used to look up person details and details of any commercial flora licences held. The data is read-only and is downloaded from the VAX ORACLE database management system via the Programmer/Developer Form. The form appears as shown below (but in colour).



LICENCE.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Methods (LICENCE.FSL)	**Objects**
Uses	mainPage
Var	
Const	
close	
keyPhysical	
menuAction	
mouseEnter	
mouseExit	
mouseRightUp	

Methods (mainPage)	**Objects (mainPage)**
Uses	endorse
Var	exitbutton
Const	floraretbutton
	helpbutton
action	printBtn
arrive	processButton
close	reportprocessbutton
find_lic	
findretbttn	
menuAction	
open	
pers_find	
surname_find	

Full listings of the code follows.

Object : Licence

MethodName : Uses

Source :
```
Uses ObjectPal
        createObjMenu(var formName Form, pageName String, uio UIObject)
        FidmsObjMsg(formName String, pageName String, objID String)
        pageNameOf(ui UIObject) String
        setIsCalledTrue()
        setIsCalledFalse()
endUses
```

Object : Licence

MethodName : Var

Source :
```
Var
        uioName, choice String
        uio UIObject
        formName, floradbFm, floraretFm, licenceFm, reportsFM, endorseFM, aboutFm,StatsFM Form
        fidmslib Library
        closeVCRLg Logical
        okToExit Logical
        myBar Form
        wasCalled Logical
        newVal String
endVar
```

Object : Licence

MethodName : Const

Source :
```
Const
        helpFile   = ":fidms:floradb.hlp" ; path to Help file
        headerHelp = LongInt(2)
        detailHelp = LongInt(20007)
        proghelp   = ":fidms:fidprog.hlp" ; path to programmer help file
endConst
```

Object : Licence

MethodName : close

Source :
```
method close(var eventInfo Event)
if eventInfo.isPrefilter() then
        doDefault
else
        if floradbFm.isAssigned() then
                floradbFm.close()
        endif
        helpQuit(":fidms:floradb.hlp")
        helpquit(":fidms:fidprog.hlp")
endif
endmethod
```

Object : Licence

MethodName : mouseEnter

Source :
```
method mouseEnter(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio)  ; what is the object?
        uioName = uio.Name
        if uio.name <> self.name AND (uio.class = "Field" OR uio.class = "Multirecord"
          OR uio.class = "Button") then
                fidmslib.FIDMSObjMsg(formName.name, fidmslib.pageNameOf(uio), uioName)
        endif
endif
endmethod
```

Object : Licence

MethodName : mouseExit

Source :

```
method mouseExit(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio) ; what is the object?
        if uio.class <> "Text" and uio.class <> "Bitmap" then
                message("")
        endif
endif
endmethod
```

Object : Licence

MethodName : mouseRightUp

Source :

```
method mouseRightUp(var eventInfo MouseEvent)
var
  uiTarget      uiObject        ; The object that was right-clicked
  thisForm      Form            ; Form variable for fidmsLIB.createObjMenu()
  Obj_Name      String   ; Used to make sure we search for the right object when we call
                                   createObjMenu.
endVar
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uiTarget)
        obj_Name = uiTarget.Name
        While obj_Name.subStr(1, 1) = "#" and uiTarget.containerName <> ""
                uiTarget.attach(uiTarget.containerName)
                obj_Name = uiTarget.Name
        endWhile
        thisForm.attach()
        fidmsLib.createObjMenu(thisForm, "mainpage", uiTarget)
        if uiTarget.class = "Field" then
                if uiTarget.tabStop then
                self.postAction(uiTarget.moveTo())
                endIf
        endIf
else
endIf
endMethod
```

Object : Licence

MethodName : keyPhysical

Source :

```
method keyPhysical(var eventInfo KeyEvent)
var
        theKey  String
        thePageString
endvar
if eventInfo.isPreFilter() then
        theKey = eventInfo.vChar()  ; tell me what key was pressed
        if eventInfo.isAltKeyDown() then
                switch
                        case theKey = "P" or theKey = "p" : disableDefault
                    mainpage.printBtn.pushButton()
                        case theKey = "X" or theKey = "x" : disableDefault
                    mainpage.exitButton.pushButton()
                        otherwise: doDefault
                endswitch
        else
                switch
                        case theKey = "VK_F1" : disableDefault
                        helpShowContext(helpfile, detailhelp)
                        otherwise : doDefault
                endSwitch
        endif
endif
endmethod
```

Object : Licence

MethodName : menuAction

```
method menuAction(var eventInfo MenuEvent)
```

Source :

```
if not eventInfo.isPreFilter() then
        if eventInfo.id() = MenuControlMaximize or eventInfo.id() = MenuControlMinimize then
        disableDefault
        beep()
        message("Form cannot be resized.")
        else
                if eventInfo.reason() = menuNormal and eventInfo.id() = MenuCanClose
                 and not okToExit then
                        eventInfo.setErrorCode(CanNotDepart)
                        disableDefault
                        message("Use the Exit button to exit Flora Return Entry Form.")
                endIf
        endIf
  endIf
endmethod
```

Object : mainpage

MethodName : Uses

Source :

```
Uses ObjectPal
        setCallFormName(const CallerTitle String, xPos LongInt, yPos LongInt)
        VCRisOpen()
        VCRisClosed()
        isVCROpen() Logical
        vcrName() String
        listPageObjects(formName String, pageName String) Logical
endUses
```

Object : mainpage

MethodName : Var

Source :

```
Var
        personTC Tcursor
        wtext,maintext array[12] String
        licenceTC Tcursor
endVar
```

Object : mainpage

MethodName : Const

Source :

```
Const
  ; Define the menu constants
        #IDH_MENUS_WINDOW_CURRENTWINDOW = "Current Open Windows"
        #IDH_MENUS_WINDOWS = "&Window"

        file_print_setup = 101
        Findret  = 102 ; Persons and Licences| Return status
        Ends_Prt = 103 ; Persons and Licences| Print command
        EndExit  = 104 ; Persons and Licences| Exit command

        Edit_Cut = 201 ; Edit | Cut command
        EditCopy = 202 ; Edit | Copy command
        EditPast = 203 ; Edit | Paste command

        RecFirst = 301 ; Record | First command
        Rec_Prev = 302 ; Record | Previous command
        Rec_Next = 303 ; Record | Next command
        Rec_Last = 304 ; Record | Last command
        Rec_Plus = 305 ; Record | Insert command
        Rec_Nuke = 306 ; Record | Delete command
        Rec_Canc = 307 ; Record | Cancel Changes command
        Rec_Look = 308 ; Record | LookUp Help command

        HelpHelp = 401 ; Help | Using Help command
        HelpList = 402 ; Help | Help Index command
        Help_Use = 403 ; Help | System Help command
        Prog_help = 404 ; Help | Programmer Help

        Endments = 501 ; Other Modules | Endorsements command
```

```
        Floraret = 502 ; Other Modules | Flora Returns command
        Reports  = 503 ; Other Modules | Reports command
        Stats    = 504 ; Other Modules | Statistics command

        Find_ret = 601 ; Returns | Locate | Person No
        FindLic  = 603 ; Returns | Locate | Licence Number
        FindName = 604 ; Returns | Locate | Surname

        Exit     = 701 ; Exit command

endConst
```

Object : mainpage

MethodName : open

Source :

```
method open(var eventInfo Event)

Message("Loading Person and Licences.  One moment, please...")
delayScreenUpdates(Yes)
if not fidmslib.open("fidmslib.ldl", globalToDesktop) then
        msgStop("Failure", "fidmslib could not be opened")
endif
formName.attach()
hideSpeedBar()
maximize()
okToExit = False
delayScreenUpdates(No)
wText[1]=#IDH_MENUS_WINDOW_CURRENTWINDOW
mainText[5]=#IDH_MENUS_WINDOWS
personTC.open("persons.db")
personTC.edit()
licenceTC.open ("licences.db")
licenceTC.edit()
if not isFile("licence.fdl") then
        disableDefault
        beep()
        msgStop("Directory Error", "The Paradox for Windows " +
        "working directory must be set to the directory " +
        "containing this form, for example: " +
        "T:\\FIDMS")
        close()
else
        blankAsZero(No)
        fidmslib.open("fidmslib.ldl", globalToDesktop)
        okToExit = False
        doDefault
        self.postAction(dataBeginEdit)
        delayScreenUpdates(No)
endIf
endmethod
```

Object : mainpage

MethodName : close

Source :

```
method close(var eventInfo Event)
if fidmslib.isVCROpen() then
        myBar.close()
        fidmslib.VCRisClosed()
endif
endmethod
```

Object : mainpage

MethodName : arrive

Source :

```
method arrive(var eventInfo MoveEvent)
var
        mainMenu        Menu    ; The main menu
        PersonMenu,             ;  The Person and Licence popup menu
        EditMenu,       ; The Edit popup menu
        Filemenu,               ; The File menu
```

```
            Windowmenu,             ; The Window menu
            Rec_Menu,               ; The Record menu
            ModuleMenu,             ; The Module menu
            HelpMenu,       ; The Help menu
            FindMenu        popupMenu  ; The Person/Licence | Locate menu
endVar

fileMenu.addText("&Print", menuEnabled, userMenu + ends_Prt)
fileMenu.addtext ("Printer &Setup", MenuEnabled, Usermenu + File_print_setup)
filemenu.addtext ("E&xit", menuEnabled, userMenu + endexit)
mainmenu.addpopup ("&File", filemenu)

FindMenu.addText("&Person Number", menuEnabled, userMenu + find_ret)
FindMenu.addSeparator()
FindMenu.addText("&Licence Number", menuEnabled, userMenu + findLic)
FindMenu.addText("Licensee &Name", menuEnabled, userMenu + findName)
FindMenu.addSeparator()

PersonMenu.addPopUp("&Locate", FindMenu)
PersonMenu.addText("&Return Status",menuEnabled, userMenu + Findret)
PersonMenu.addSeparator()
MainMenu.addPopUp("Persons and &Licences", PersonMenu)

EditMenu.addText("Cu&t\tShift+Del", menuDisabled + menuGrayed, userMenu + edit_Cut)
EditMenu.addText("&Copy\tCtrl+Ins", menuDisabled + menuGrayed, userMenu + editCopy)
EditMenu.addText("&Paste\tShift+Ins", menuDisabled + menuGrayed, userMenu + editPast)
MainMenu.addPopUp("&Edit", EditMenu)

Rec_Menu.addText("&First\tCtrl+F11", menuEnabled, userMenu + recFirst)
Rec_Menu.addText("&Previous\tF11", menuEnabled, userMenu + rec_Prev)
Rec_Menu.addText("&Next\tF12", menuEnabled, userMenu + rec_Next)
Rec_Menu.addText("&Last\tCtrl+F12", menuEnabled, userMenu + rec_Last)
Rec_Menu.addSeparator()
Rec_Menu.addText("Lookup &Help\tCtrl+Space", menuEnabled, userMenu + rec_Look)
Rec_Menu.addText("&Insert\tInsert", menuEnabled, userMenu + rec_Plus)
Rec_Menu.addText("&Delete\tCtrl+Del", menuEnabled, userMenu + rec_Nuke)
Rec_Menu.addSeparator()
Rec_Menu.addText("&Cancel Changes\tAlt+BkSp", menuEnabled, userMenu + rec_Canc)
Rec_Menu.addText("&Record lookup\TCtrl+SpBar", MenuEnabled, Usermenu + Rec_Look)
MainMenu.addPopUp("&Record", Rec_Menu)

HelpMenu.addText("Using &Help\tCtrl+F1", menuEnabled, userMenu + helpHelp)
HelpMenu.addText("Help &Index\tShift+F1", menuEnabled, userMenu + helpList)
HelpMenu.addText("&Using this Form\tF1", menuEnabled, userMenu + help_Use)
HelpMenu.addText("&Programmer Help", menuEnabled, userMenu + Prog_help)
MainMenu.addPopUp("&Help", HelpMenu)

ModuleMenu.addtext ("&Endorsements", menuEnabled, userMenu + Endments)
ModuleMenu.addtext ("&Statistics", menuEnabled, userMenu + Stats)
ModuleMenu.addtext ("&Reports", menuEnabled, userMenu + Reports)
ModuleMenu.addtext ("&Flora Returns", menuEnabled, userMenu + Floraret)
MainMenu.addpopup ("Other &Modules", Modulemenu)

windowmenu.addStaticText(wText[1])
mainMenu.addPopUp(mainText[5],windowmenu)

mainMenu.show()

endmethod
```

Object :	mainpage

MethodName :	action

Source :	method action(var eventInfo ActionEvent) setTitle("Person and Licences Form") endmethod

Object :	mainpage

MethodName :	menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
var
        menu_Cmd        smallInt ; The constant returned by a menu command
        editTest        Logical  ; Flag variable used to test the type of
                                 ; object that is active
        findTest        String   ; Flag variable that holds the results of
                                 ; a search attempt
        calledBy        Form     ; Holds the handle of the form that opened
                                 ; this form
endVar

menu_Cmd = eventInfo.id()
switch
        case menu_Cmd = menuInit :
        try
        editTest = active.Editing
        onFail
        editTest = False
        endTry
        if editTest then
        if active.selectedText <> "" then
                        setMenuChoiceAttributeByID(userMenu + edit_Cut, menuEnabled)
                        setMenuChoiceAttributeByID(userMenu + editCopy, menuEnabled)
        else
                        setMenuChoiceAttributeByID(userMenu + edit_Cut, menuDisabled +
menuGrayed)
                        setMenuChoiceAttributeByID(userMenu + editCopy, menuDisabled +
menuGrayed)
        endIf
        endIf
        case menu_Cmd = MenuControlClose :
        disableDefault
        exitbutton.pushButton()
        case menu_Cmd = MenuControlMaximize or menu_Cmd = MenuControlMinimize :
        if formcaller(calledBy) then
        disableDefault
        beep()
        msgInfo("Oops!", "Because this form was opened by the " +
                                calledBy.getTitle() + " form, the Maximize and " +
                                "Minimize buttons are disabled.  Sorry...")
        endIf
        case menu_Cmd = userMenu + Find_ret : pers_find.pushbutton()
        case menu_Cmd = userMenu + FindLic  : Find_lic.pushbutton()
        case menu_Cmd = userMenu + FindName : surname_find.pushbutton()
        case menu_cmd = usermenu + file_print_setup: active.menuAction(menuFilePrinterSetup)
        case menu_Cmd = userMenu + Ends_Prt : printBtn.pushButton()
        case menu_Cmd = userMenu + EndExit  : exitbutton.pushButton()
        case menu_Cmd = userMenu + Edit_cut :
        active.menuAction(menuEditCut)
        setMenuChoiceAttributeByID(userMenu + editPast, menuEnabled)
        case menu_Cmd = userMenu + EditCopy :
        active.menuAction(menuEditCopy)
        setMenuChoiceAttributeByID(userMenu + editPast, menuEnabled)
        case menu_Cmd = userMenu + EditPast : active.menuAction(menuEditPaste)
        case menu_Cmd = userMenu + RecFirst : active.postAction(dataBegin)
        case menu_Cmd = userMenu + Rec_Prev : active.postAction(dataPriorRecord)
        case menu_Cmd = userMenu + Rec_Next : active.postAction(dataNextRecord)
        case menu_Cmd = userMenu + Rec_Last : active.postAction(dataEnd)
        case menu_Cmd = userMenu + Rec_Plus : Action(dataInsertRecord)
        case menu_Cmd = userMenu + Rec_Nuke : active.postAction(dataDeleteRecord)
        case menu_Cmd = userMenu + Rec_Canc : active.postAction(dataCancelRecord)
        case menu_Cmd = userMenu + Rec_Look : active.postAction(dataLookup)
        case menu_Cmd = userMenu + Endments  : Endorse.pushbutton()
        case menu_Cmd = userMenu + Floraret   : Floraretbutton.pushbutton()
        case menu_Cmd = userMenu + Findret   : Findretbttn.pushbutton()
        case menu_Cmd = userMenu + Stats     : processButton.pushbutton()
        case menu_Cmd = userMenu + Reports   : reportprocessbutton.pushbutton()
        case menu_Cmd = userMenu + Exit     : exitButton.pushbutton()
        case menu_Cmd = userMenu + HelpHelp : helpOnHelp()
        case menu_Cmd = userMenu + HelpList : helpShowContext(helpFile,1000)
        case menu_Cmd = userMenu + Help_Use : helpShowContext(helpFile, headerHelp)
```

```
        case menu_Cmd = userMenu + Prog_help  : helpShowindex(progHelp)
endswitch
endmethod
```

Object : mainpage.processButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(statsFM) then
        statsFM.bringtoTop()
        statsFM.wait()  ; put statsFM into wait state
        statsFM.hide()
        maximize()
        mainpage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if statsFM.Open("Stats.fdl", WinStyleMaximize) then
                statsFM.Wait()  ; put statsFM into wait state
                statsFM.hide()
                maximize()
                mainpage.moveTo()
        else
                msgInfo("Status", "Sorry, the Statistics form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.surname_find

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        surname         String
        person          UIObject
        userchoice      String
        numfound        smallint
        olesound        OLE
endvar
ignorecaseinlocate(yes)
surname  = ""
surname.view ("Enter a Surname")
person.attach(PERSONS)
if surname <> "" then
        if person.locate("Surname",surname) then
                beep()
                numfound = 1
                message ("")
                userchoice = msgQuestion ("Choose", "Do you wish to see next " + String(surname))
                while userchoice = "Yes"
                        person.locatenext("Surname",surname)
                        beep()
                        numfound = numfound + 1
                        message ("")
                        userchoice = msgQuestion ("Choose",
                          "Do you wish to see next " + String(surname))
                endwhile
                return
                if not person.locatenext ("Surname", surname) then
                        msgInfo("That's all folks for " + surname,"No further records")
                endif
                msgInfo("Found " + surname, strval(numFound) + " times.")
                return
        else
                msgInfo("No record found for " + surname, "Check spelling")
        endif
endif
endmethod
```

Object : mainpage.surname_find

MethodName : make_music

Source :

```
method make_music()

var
        quarterNote, octave, note       LongInt
        power                           Number
        olevar                          OLE
endVar

const
 noteA1  = 110
 noteA#1 = 116
 noteB1  = 123
 noteC1  = 130
 noteC#1 = 138
 noteD1  = 146
 noteD#1 = 155
 noteE1  = 164
 noteF1  = 174
 noteF#1 = 184
 noteG1  = 195
 noteG#1 = 207
 noteA2  = 220
 noteA#2 = 234
 noteB2  = 249
 noteC2  = 265
 noteC#2 = 282
 noteD2  = 300
endConst

sound(noteA1, 200)
sound(noteD1, 150)
sound(noteF#1, 50)
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteA2, 150)
sound(noteF#1, 50)
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteC#2, 150)
sound(noteD2, 50)
sound(noteA2, 100)
sound(noteF#1, 100)
sound(noteD1, 100)
sleep(1000)
endMethod
```

Object : mainpage.reportprocessbutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(reportsFm) then
        reportsFm.bringtoTop()
        reportsFm.wait()  ; put reportsFm into wait state
        reportsFm.hide()
```

```
        maximize()
        reportsFM.moveTo()
else
        fidmslib.setIsCalledTrue()
        if reportsFm.Open("report.fdl", WinStyleMaximize) then
                reportsFm.Wait()  ; put reportsFm into wait state
                reportsFm.hide()
                maximize()
                reportsFM.moveTo()
        else
                msgInfo("Status", "Sorry, the Reports form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.floraretbutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(floraretFm) then
        floraretFm.bringtoTop()
        floraretFm.wait()  ; put floraretFm into wait state
        floraretFm.hide()
        maximize()
        FloraretFM.moveTo()
else
        fidmslib.setIsCalledTrue()
        if floraretFm.Open("floraret.fdl", WinStyleMaximize) then
                floraretFm.Wait()  ; put floraretFm into wait state
                floraretFm.hide()
                maximize()
                FloraretFM.moveTo()
        else
                msgInfo("Status", "Sorry, the floraret form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.pers_find

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        persNo LongInt
        person UIObject
endVar

person.attach(PERSONS)
persNo  = 0
persNo.view ("Enter a person number")

if persNo <> 0 then
        if not person.locate("Person_No",persNo) then
                beep()
                msginfo ("Search unsuccessful", "Couldn't locate Person No " + String(PersNo))
        endif
endif
endmethod
```

Object : mainpage.Find_lic

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        ret_stattv Tableview
        ret_qry Query
        retstatTC TCursor
        retstatdlg Form
        theChoice String
        licence_no,dlgval String
        useritem String
endvar
licence_no = ""
licence_no.view ("Enter a licence number")
if licence_no <> " " then
        myitem.value = licence_no
endif
useritem = myitem.value

ret_qry = query

:FIDMS:Persons.db       |Person_no  |Surname   |Initials |Forename  |Address1   |Address2  |Address3
|Postcode   |Home_Phone_No|Work_Phone_No |
                |Check _abcd|Check    |Check   |Check    |Check     |Check     |Check     |Check
|Check      |Check       |

:Fidms:licences.db      |Person_no |licence_no              |Valid_from|Expiry_date|
                        |_abcd      |Check ~useritem,_XYZ |Check     |Check     |

:FIDMS:LOC.DB     |LICENCE_NO   |LOCATION   |
                  |_XYZ        |CHECK      |

endQuery

message ("The query is processing")
sleep (1000)
if not executeqbe(ret_qry,":FIDMS:lic_stat.db") then
        errorshow()
endif
theChoice = msgYesNoCancel ("Reports","Do you wish to view the licence details?")
switch
        case theChoice = "Yes" :
        if retstatDlg.open("Gen_lic.fdl") then
                dlgVal =  retstatDlg.wait()
        endif
        otherwise                   :return
endswitch
endmethod
```

Object : mainpage.endorse

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(endorseFM) then
        endorseFM.bringtoTop()
        endorseFm.wait()  ; put endorseFm into wait state
        endorseFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if endorseFm.Open("endorsfm.fdl", WinStyleMaximize) then
                endorseFm.Wait()  ; put endorseFm into wait state
                endorseFm.hide()
                maximize()
                mainPage.moveTo()
```

```
        else
                msgInfo("Status", "Sorry, the Endorsement form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.printBtn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        lictv Tableview
        licqry Query
        userin String
        floralic Report
        repInfo ReportPrintInfo
        theChoice String
endvar

userin = ""
userin.view ("Enter Licence number to print")
if userin <> "0" then
        myitem.value = userin
endif

licqry = query

persons.db  |person_no   |Surname|Title  |FORENAME  |ADDRESS1|ADDRESS2|ADDRESS3|Postcode|
            |_abcd      |Check  |Check  |Check     |Check   |Check   |Check   |Check   |

licences.db  |person_no     |licence_no     |Valid_from   |Expiry_date|
             |_abcd         |Check ~userin  |Check        |Check      |

endQuery

if not executeqbe(licqry,"licprn.db") then
        errorshow()
endif

theChoice = msgYesNoCancel ("Reports","Do you wish to print licences?")
switch
        case theChoice = "Yes" : floralic.open("floralic.rdl")
        repInfo.nCopies = 1
        floralic.print(repInfo)
        otherwise              :return
endswitch
endmethod
```

Object : mainpage.exitButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
choice = msgQuestion("Quit F*I*D*M*S", "Do you want to quit " +
                "the module?")
if choice= "Yes" then
        okToExit = True
        close()
else
        eventInfo.setErrorCode(cannotDepart)
endif
endmethod
```

Object : mainpage.helpButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        helpShowindex (helpFile)
endmethod
```

Object :	mainpage.Findretbttn
MethodName :	pushButton

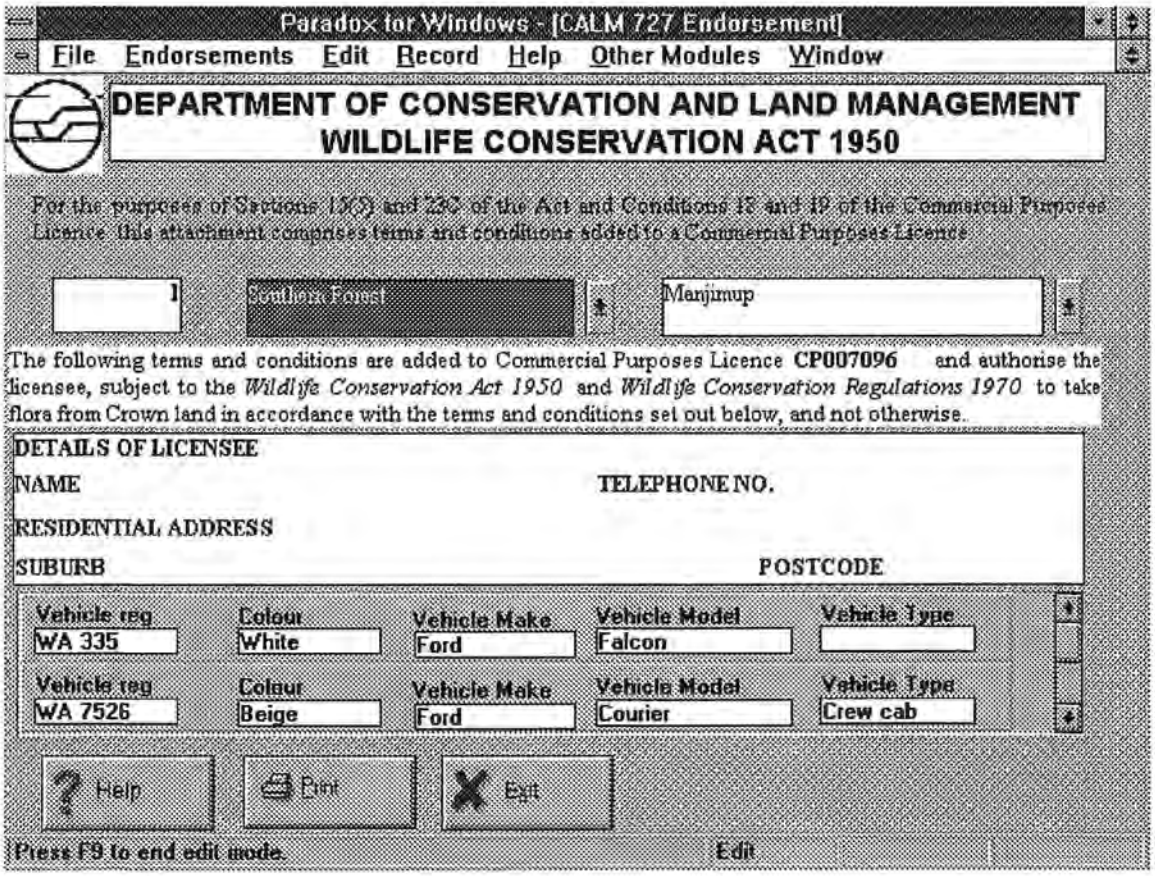
Source :

```
method pushButton(var eventInfo Event)
var
        ret_stattv Tableview
        ret_qry Query
        retstatTC TCursor
        retstatdlg Form
        theChoice String
        licence_no,dlgval String
        useritem String
endvar

licence_no = ""
licence_no.view ("Enter a licence number")
if licence_no <> " " then
        myitem.value = licence_no
endif
useritem = myitem.value

  ret_qry = query

:Fidms:persons.db   |Person_no |Surname   |Title|Forename|
                    |_efgh     |Check     |Check|Check   |

:Fidms:licences.db  |Person_no|licence_no       |
                    |_efgh    |Check_abcd,~useritem|

:Fidms:retnum.db    |Licence_number|Collecting_date|
                    |_abcd         |Check          |

endQuery

message ("The query is processing")
sleep (1000)
if not executeqbe(ret_qry,":priv:ret_stat.db") then
        errorshow()
endif
theChoice = msgYesNoCancel ("Reports","Do you wish to view the return status?")
switch
        case theChoice = "Yes" :
        if retstatDlg.open("retstat.fdl") then
                dlgVal =  retstatDlg.wait()
        endif
        otherwise                   :return
endswitch
endmethod
```

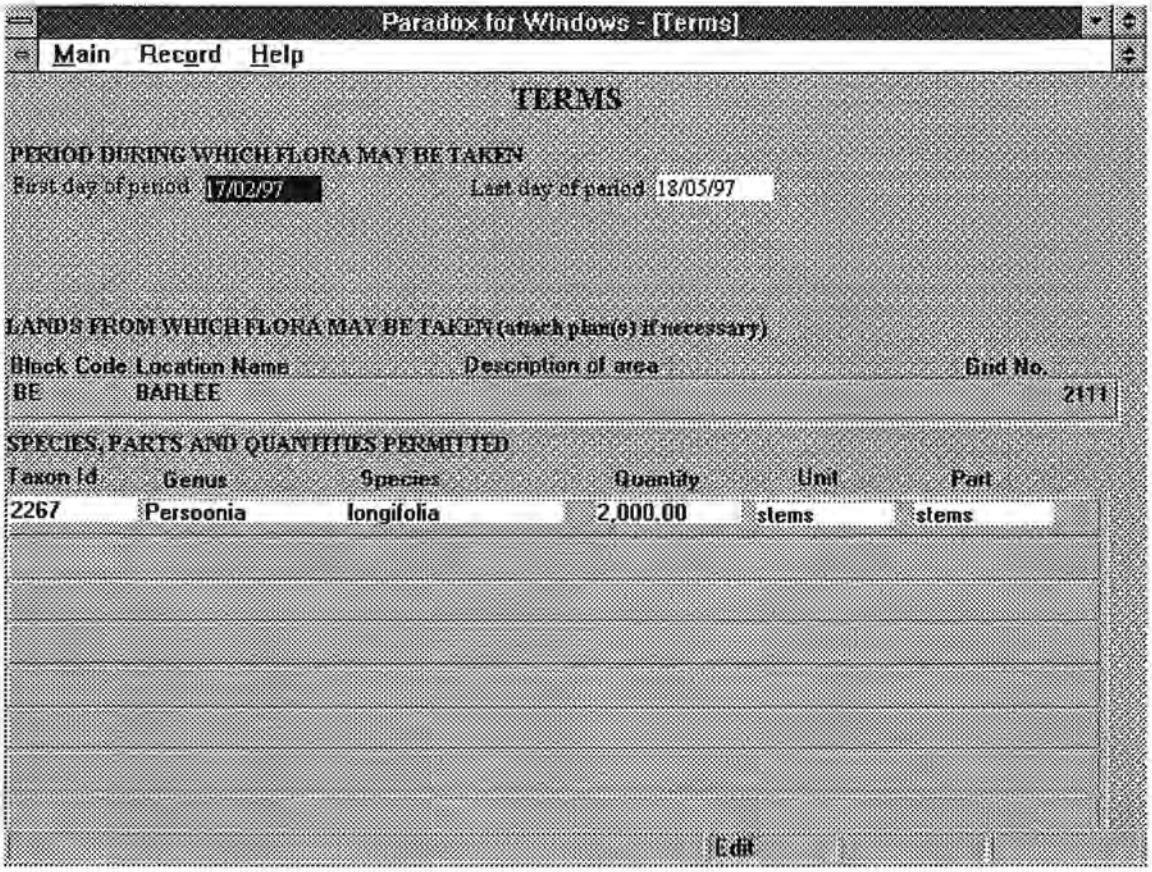
CALM Endorsements Form (EndorsFM.FSL)

The CALM Endorsements form consists of three pages which allow the user to issue and print a CALM endorsement ("727"):

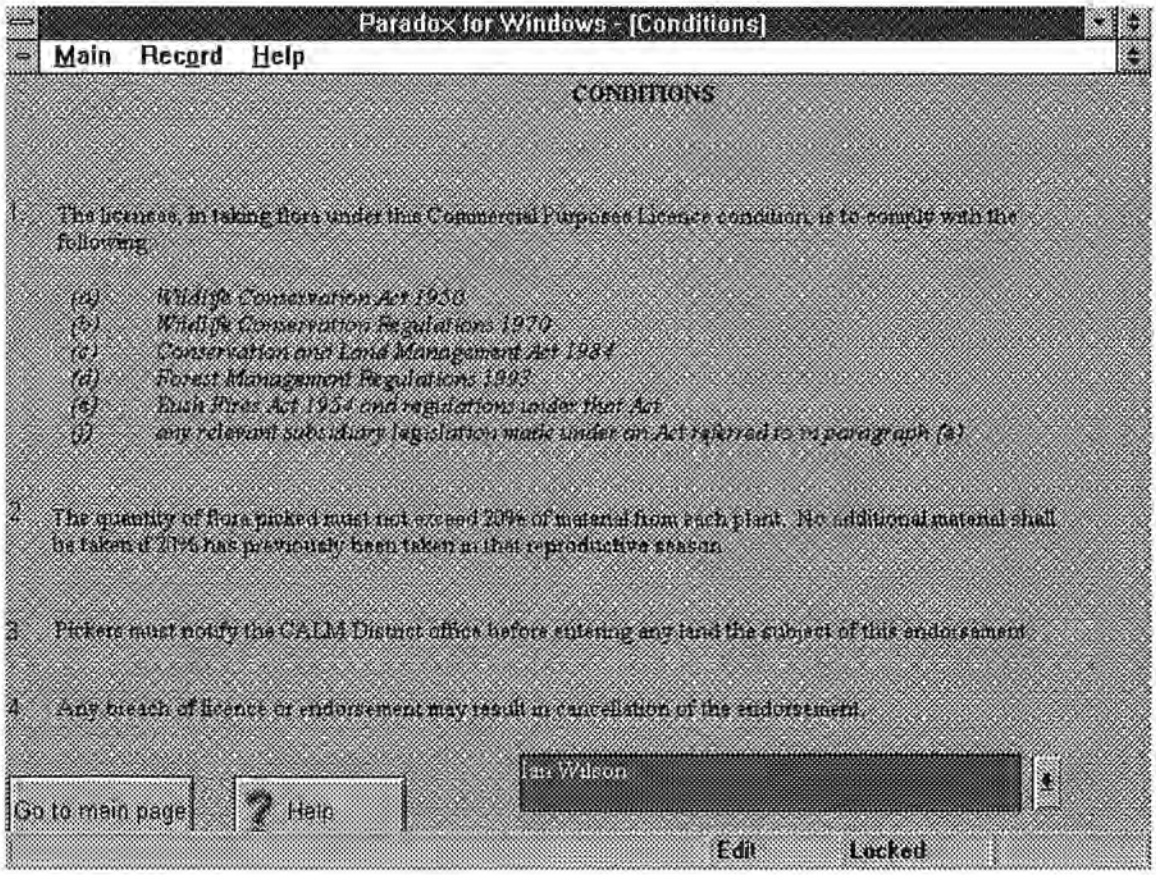
- mainPage is the main page of the Endorsement Form). Controls on this page let the user automatically create a new endorsement, locate any endorsement already issued, or jump to the other forms. The form also lets users view, enter and edit details on an endorsee. The first page automatically shows contact data such as name, address, and telephone numbers when a valid licence number is entered. It also has fields that allow the user to enter data such as vehicle details, Region and District. The page appears as shown below (but in colour).



- termPage enters, stores and retrieves information about the terms of an endorsement, including applicable land areas and species. The page appears as shown below (but in colour).



- condPage shows a standard set of endorsement licence conditions and allows the user to enter his/her name as the issuing officer. The page form appears as shown below (but in colour).



ENDORSFM.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Methods (ENDORSFM.FSL)
Uses
Var
Const

close
keyPhysical
menuAction
mouseEnter
mouseExit
mouseRightUp
renew

Objects
mainPage
termPage
condPage

Methods (mainPage)
Uses
Var
Const

action
arrive
close
district_find
end_find
licence_find
make_music
menuAction
mouseExit
new_endorse
open
region_find

Objects (mainPage)
exitbutton
floraretbutton
helpbutton
licence_button
printBtn
processButton
reportprocessbutton

Fields (mainPage)
Licence No
District
Region

Methods (District)
removeFocus
setFocus

Methods (Licence No)
depart
removeFocus
setFocus

Methods (Region)
removeFocus
setFocus

Methods (termPage)
arrive
menuAction
pageMenu

Objects (termPage)
exitbutton
helpButton
mainButton
printBtn

Fields (termPage)
Block Code
Endorse From
Endorse To
Quantity
Species
Species Code

Methods (Block Code)
changeValue
newValue

Methods (Endorse From)
depart

Methods (Endorse To)
depart

Methods (Quantity)
canDepart

Methods (Species)
depart

Methods (Species Code)
depart

Methods (condPage)
arrive
menuAction
pageMenu

Objects (condPage)
exitButton
helpButton
mainButton
printBtn

Fields (condPage)
Issue_officer

Methods (Issue_officer)
removeFocus
setFocus

Full listings of the code follows.

Object : endorsFM

MethodName : Uses

Source :
```
Uses ObjectPal
        showMe(const theTitle String, const listDataSource String)
        clearPageValues(formName String, pageName String)
        createObjMenu(var formName Form, pageName String, uio UIObject)
        fidmsObjMsg(formName String, pageName String, objID String)
        pageNameOf(ui UIObject) String
        returnIsCalled() Logical
        setIsCalledTrue()
        setIsCalledFalse()
        formCanClose()
        clearPageValues(formName String, pageName String)
        createObjMenu(var formName Form, pageName String, uio UIObject)
        setPageValues(formName String, pageName String) Logical
        pageNameOf(ui UIObject) String
        returnIsCalled() Logical
endUses
```

Object : endorsFM

MethodName : Var

Source :
```
Var
        uioName, choice String
        uio UIObject
        formName, floradbFm, floraretFm, licenceFm, endorsFM, reportsFM, aboutFm, statsFM Form
        fidmslib Library
        closeVCRLg Logical
        okToExit Logical
        myBar Form
        wasCalled Logical
        newVal String
        vehTC    tCursor
        renvehTC  tCursor
        endlocTC  tCursor
        endrenTC  tCursor
        endnoTC   tCursor
endVar
```

Object : endorsFM

MethodName : Const

Source :
```
Const
        helpFile   = ":fidms:floradb.hlp" ; path to Help file
        headerHelp = LongInt(3)
        detailHelp = LongInt(20007)
        proghelp   = ":fidms:fidprog.hlp" ; path to programmer help file
endConst
```

Object : endorsFM

MethodName : close

Source :
```
method close(var eventInfo Event)
if eventInfo.isPrefilter() then
        doDefault
else
        if floradbFm.isAssigned() then
                floradbFm.close()
        endif
endif
endmethod
```

Object : endorsFM

MethodName : mouseEnter

Source :
```
method mouseEnter(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
```

```
        uioName = uio.Name
        if uio.name <> self.name AND  (uio.class = "Field"
          OR uio.class = "Multirecord" OR uio.class = "Button") then
                fidmslib.FIDMSObjMsg(formName.name, fidmslib.pageNameOf(uio), uioName)
        endif
endif
endmethod
```

Object : endorsFM

MethodName : mouseExit

Source :

```
method mouseExit(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        message("")
else
endIf
endmethod
```

Object : endorsFM

MethodName : mouseRightUp

Source :

```
method mouseRightUp(var eventInfo MouseEvent)
var
        uiTarget uiObject ; The object that was right-clicked
        thisForm Form     ; Form variable for fidmsLIB.createObjMenu()
        Obj_Name String   ; Used to make sure we search for the
endVar

if eventInfo.isPreFilter() then
        eventInfo.getTarget(uiTarget)
        obj_Name = uiTarget.Name
        while obj_Name.subStr(1, 1) = "#" and uiTarget.containerName <> ""
                uiTarget.attach(uiTarget.containerName)
                obj_Name = uiTarget.Name
        endWhile
        thisForm.attach()
        fidmsLib.createObjMenu(thisForm, fidmslib.pageNameOf(uio), uiTarget)
        if uiTarget.class = "Field" then
                if uiTarget.tabStop then
                        self.postAction(uiTarget.moveTo())
                endIf
        endIf
else
endIf
endMethod
```

Object : endorsFM

MethodName : keyPhysical

Source :

```
method keyPhysical(var eventInfo KeyEvent)
var
        theKey String
        thePage String
endvar
if eventInfo.isPreFilter() then
        theKey = eventInfo.vChar()  ; tell me what key was pressed
        if eventInfo.isAltKeyDown() then
                switch
                        case theKey = "E" or theKey = "e" : disableDefault
                        mainpage.exitButton.pushButton()
                        case theKey = "N" or theKey = "n" : disableDefault
                        mainpage.new_endorse.pushbutton()
                        case theKey = "R" or theKey = "r" : disableDefault
                        mainpage.reportprocessbutton.pushbutton()
                        case theKey = "X" or theKey = "x" : disableDefault
                        mainpage.exitButton.pushButton()
                        otherwise: doDefault
                endswitch
        else
                switch
```

```
                        case theKey = "VK_F1" : disableDefault
                        helpShowContext(helpfile, detailhelp)
                        otherwise : doDefault
                endSwitch
        endif
endif
endmethod
```

Object : endorsFM

MethodName : action

Source :

```
method action(var eventInfo ActionEvent)
        setTitle("CALM 727 Endorsement")
endmethod
```

Object : endorsFM

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
if not eventInfo.isPreFilter() then
        if eventInfo.id() = MenuControlMaximize or eventInfo.id() = MenuControlMinimize then
        disableDefault
        beep()
        message("Form cannot be resized.")
        else
                if eventInfo.reason() = menuNormal and
                  eventInfo.id() = MenuCanClose and not okToExit then
                        eventInfo.setErrorCode(CanNotDepart)
                        disableDefault
                        message("Use the Exit button to exit ENDORSE.")
                endIf
        endIf
endIf
endmethod
```

Object : endorsFM

MethodName : renew

Source :

```
method renew()
var
        endorseTC tCursor
        end_val   LongInt
        tvRenew   TableView
        vehTC     tCursor
        endlocTC  tCursor
        endspecTC tCursor
        endrenTC  tCursor
        detailuio,
        end_locuio,
        end_specuio
        uiObject
endVar
end_val = 0
end_val.view ("Enter endorsement number to renew")
myitem.value = end_val
endorseTC.open("endorse.db")
endorseTC.edit()
scan endorseTC for endorseTC."Endorsement_no" = myitem.value :
        new_endorse.pushbutton()
        region.value = endorseTC.region
        district.value = endorseTC.district
        licence_no.value = endorseTC.licence_no
        Endorse_From.value = endorseTC."Endorse From"
        Endorse_To.value = endorseTC."Endorse To"
        condpage.issue_officer.value = endorseTC.issue_officer
        postRecord()
endScan

detailuio.attach(DETAIL)
vehTC.open("Vehicle.db")
```

```
vehTC.edit()
scan vehTC for vehTC."End_no" = myitem.value :
        mainPage.DETAIL.insertrecord()
        mainPage.DETAIL.vehicle_reg.value = vehTC.vehicle_reg
        mainPage.DETAIL.Vehicle_Make.value = vehTC.Vehicle_make
        mainPage.DETAIL.colour.value = vehTC.colour
        mainPage.DETAIL.Vehicle_Model.value = vehTC.Vehicle_Model
        mainPage.DETAIL.Vehicle_Type = vehTC.Vehicle_Type
endScan

end_locuio.attach(END_LOC)
endlocTC.open("end_loc.db")
endlocTC.edit()
endspecTC.open("end_spec.db")
endspecTC.edit()
end_specuio.attach(END_SPEC)
scan endlocTC for endlocTC.End_no = myitem.value :
        termPage.END_LOC.insertrecord()
        termpage.END_LOC.Block_code.value = endlocTC.Block_code
        termpage.END_LOC.Location_Name = endlocTC.Location_Name
        termpage.END_LOC.Location_Description.value = endlocTC.Location_Description
        termpage.END_LOC.GridNo.value = endlocTC.GridNo
        scan endspecTC for endspecTC.end_no = myitem.value
          AND endspecTC.block_code = endlocTC.block_code :
                endspecTC.block_code = termpage.END_LOC.Block_code.value
                termpage.END_SPEC.species_code.value = endspecTC.species_code
                termpage.END_SPEC.genus.value = endspecTC.genus
                termpage.END_SPEC.species.value = endspecTC.species
                termpage.END_SPEC.infracat.value = endspecTC.infracat
                termpage.END_SPEC.infraspecies.value = endspecTC.infraspecies
                termpage.END_SPEC.quantity.value = endspecTC.quantity
                termpage.END_SPEC.unit.value = endspecTC.unit
                termpage.END_SPEC.part.value = endspecTC.part
                termPage.END_SPEC.insertrecord()
        endscan
endScan
endmethod
```

Object : condpage

MethodName : arrive

Source :

```
method arrive(var eventInfo MoveEvent)
        setTitle("Conditions")
        pageMenu()
endmethod
```

Object : condpage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
var
        mc String
endvar
mc = eventInfo.menuChoice()
switch
        case mc = "&Print"                  : printBtn.pushbutton()
        case mc = "E&xit"                   : exitButton.pushbutton()
        case mc = "&Main"                   : mainbutton.pushbutton()
        case mc = "&First\tCtrl+F11"        : active.action(DataBegin)
        case mc = "&Last\tCtrl+F12"     : active.action(DataEnd)
        case mc = "&Next\tF12"              : active.action(DataNextRecord)
        case mc = "Next &Section\tF4"       : active.action(FieldGroupForward)
        case mc = "&Previous\tF11"      : active.action(DataPriorRecord)
        case mc = "&Insert\tIns"            : active.action(DataInsertRecord)
        case mc = "&Delete\tCtrl+Del"   : active.action(DataDeleteRecord)
        case mc = "C&ancel Changes\tAlt+Bksp" : active.action(DataCancelRecord)
        case mc = "U&sing this Form"        : helpShowContext(helpFile, headerHelp)
        case mc = "&Using Help"             : helpOnHelp()
endswitch
```

endmethod

Object : condpage

MethodName : proc

Source :

```
proc pageMenu()
Var
        resultsMenu Menu
        dropMenu1, dropMenu2, dropMenu3 popUpMenu
endVar
resultsMenu.addText("&Main")
dropMenu1.addText("&Print")
dropMenu1.addText("E&xit")
dropMenu2.addtext("&First\tCtrl+F11")
dropMenu2.addtext("&Last\tCtrl+F12")
dropMenu2.addtext("&Next\tF12")
dropMenu2.addtext("Next &Section\tF4")
dropMenu2.addtext("&Previous\tF11")
dropMenu2.addSeparator()
dropMenu2.addtext("&Insert\tIns")
dropMenu2.addtext("&Delete\tCtrl+Del")
dropMenu2.addtext("C&ancel Changes\tAlt+Bksp")
resultsMenu.addPopUp("Rec&ord", dropMenu2)
dropMenu3.addText("U&sing this Form")
dropMenu3.addText("&Using Help")
resultsMenu.addPopUp("&Help", dropMenu3)
resultsMenu.show()
endproc
```

Object : condpage.printBtn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
  endtv Tableview
  endqry Query
  userin String
  endorse Report
  repInfo ReportPrintInfo
  theChoice,useritem String
endvar

userin = ""
userin.view ("Enter licence no, blank is current endorsement")
if userin <> "" then
        myitem.value = userin
        useritem = myitem.value
        endqry = query

        persons.db      |person_no    |Surname|Title|forename
|Address1|Address2|Address3|Postcode|
                        |_abcd       |Check  |Check|Check    |Check   |Check   |Check   |Check   |

        licences.db  |Licence_no   |person_no|
            |~useritem, _bcd|_abcd   |

        Endorse.db  |Licence_no   |Endorsement_no|Issue_officer|Region|District|Endorse
from|Endorse to|
            |Check _bcd   |Check       |Check       |Check |Check   |Check      |Check    |

        endQuery
else
        if userin = "" then
                useritem = endorsement_no.value
                endqry = query

                persons.db  |person_no |Surname|Title|forename
|Address1|Address2|Address3|Postcode|
                        |_abcd       |Check  |Check|Check    |Check   |Check   |Check   |Check   |
```

```
                    licences.db |person_no|Licence_no|
                                |_abcd   |_lic     |

                    Endorse.db |Licence_no|Endorsement_no|Issue_officer|Region|District|Endorse
from|Endorse to|
                                |Check _lic  |Check ~useritem|Check       |Check |Check |Check    |Check    |

                    endQuery
          endif
endif
if not executeqbe(endqry,"endprn.db") then
          errorshow()
endif
theChoice = msgYesNoCancel ("Reports","Do you wish to print endorsement?")
switch
          case theChoice = "Yes" : endorse.open("endorse.rdl")
          repInfo.nCopies = 2
          repInfo.makecopies = true
          endorse.print(repInfo)
          otherwise              :return
endswitch
endmethod
```

Object : condpage.helpButton

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
          helpShowContext(helpFile, 20006)
endmethod
```

Object : condpage.exitButton

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
choice = msgQuestion("Quit F*I*D*M*S", "Do you want to quit " + "the module?")
if choice= "Yes" then
          kToExit = True
          lose()
else
          ventInfo.setErrorCode(cannotDepart)
endif
endmethod
```

Object : condpage.mainbutton

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
          mainpage.moveto()
endmethod
```

Object : condpage.Issue_officer

MethodName : setFocus

Source :
```
method setFocus(var eventInfo Event)
          self.color = red
endmethod
```

Object : condpage.Issue_officer

MethodName : removeFocus

Source :
```
method removeFocus(var eventInfo Event)
          self.color = white
endmethod
```

Object : termpage

MethodName : arrive

Source : method arrive(var eventInfo MoveEvent)

```
        pageMenu()
endmethod
```

Object : termpage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
var
        mc String
endvar

mc = eventInfo.menuChoice()
switch
        case mc = "&Print"                  : printBtn.pushbutton()
        case mc = "E&xit"                   : exitButton.pushbutton()
        case mc = "&Main"                   : mainbutton.pushbutton()
        case mc = "&First\tCtrl+F11"        : active.action(DataBegin)
        case mc = "&Last\tCtrl+F12"         : active.action(DataEnd)
        case mc = "&Next\tF12"              : active.action(DataNextRecord)
        case mc = "Next &Section\tF4"       : active.action(FieldGroupForward)
        case mc = "&Previous\tF11"          : active.action(DataPriorRecord)
        case mc = "&Insert\tIns"            : active.action(DataInsertRecord)
        case mc = "&Delete\tCtrl+Del"       : active.action(DataDeleteRecord)
        case mc = "C&ancel Changes\tAlt+Bksp" : active.action(DataCancelRecord)
        case mc = "U&sing this Form"        : helpShowContext(helpFile, headerHelp)
        case mc = "&Using Help"             : helpOnHelp()
endswitch
endmethod
```

Object : termpage

MethodName : proc

Source :

```
proc pageMenu()
Var
        resultsMenu Menu
        dropMenu1, dropMenu2, dropMenu3 popUpMenu
endVar
resultsMenu.addText("&Main")
dropMenu1.addText("&Print")
dropMenu1.addText("E&xit")
dropMenu2.addtext("&First\tCtrl+F11")
dropMenu2.addtext("&Last\tCtrl+F12")
dropMenu2.addtext("&Next\tF12")
dropMenu2.addtext("Next &Section\tCtrl+F4")
dropMenu2.addtext("&Previous\tF11")
dropMenu2.addSeparator()
dropMenu2.addtext("&Insert\tIns")
dropMenu2.addtext("&Delete\tCtrl+Del")
dropMenu2.addtext("C&ancel Changes\tAlt+Bksp")
resultsMenu.addPopUp("Rec&ord", dropMenu2)
dropMenu3.addText("U&sing this Form")
dropMenu3.addText("&Using Help")
resultsMenu.addPopUp("&Help", dropMenu3)
resultsMenu.show()
endproc
```

Object : termpage.printBtn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        endtv Tableview
        endqry Query
        userin String
        endorse Report
        repInfo ReportPrintInfo
        theChoice,useritem String
endvar
userin = ""
userin.view ("Enter licence no, blank is current endorsement")
```

```
if userin <> "" then
        myitem.value = userin
        useritem = myitem.value
        endqry = query

        persons.db   |person_no    |Surname|Title|forename  |Address1|Address2|Address3|Postcode|
                     |_abcd       |Check  |Check|Check     |Check   |Check   |Check   |Check   |

        licences.db  |Licence_no   |person_no|
                     |~useritem, _bcd|_abcd   |

        Endorse.db   |Licence_no   |Endorsement_no|Issue_officer|Region|District|Endorse
from|Endorse to|
                     |Check _bcd   |Check       |Check       |Check |Check   |Check      |Check    |

        endQuery
else
        if userin = "" then
                useritem = endorsement_no.value
                endqry = query

                persons.db   |person_no |Surname|Title|forename
|Address1|Address2|Address3|Postcode|
                             |_abcd       |Check  |Check|Check     |Check   |Check   |Check   |Check   |

                licences.db  |person_no|Licence_no|
                             |_abcd   |_lic     |

                Endorse.db   |Licence_no|Endorsement_no|Issue_officer|Region|District|Endorse
from|Endorse to|
                            |Check _lic  |Check ~useritem   |Check     |Check |Check   |Check   |Check   |
                endQuery
        endif
endif
if not executeqbe(endqry,"endprn.db") then
        errorshow()
endif
theChoice = msgYesNoCancel ("Reports","Do you wish to print endorsement?")
switch
        case theChoice = "Yes" : endorse.open("endorse.rdl")
        replnfo.nCopies = 2
        replnfo.makecopies = true
        endorse.print(replnfo)
        otherwise              :return
endswitch
endmethod
```

Object : termpage.helpButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        helpShowContext(helpFile, 20006)
endmethod
```

Object : termpage.exitButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
choice = msgQuestion("Quit F*I*D*M*S", "Do you want to quit " + "the module?")
if choice= "Yes" then
        okToExit = True
        close()
else
        eventInfo.setErrorCode(cannotDepart)
endif
endmethod
```

Object : termpage.mainbutton

MethodName : pushButton

```
method pushButton(var eventInfo Event)
```

Source :

```
            mainpage.moveto()
endmethod
```

Object : termpage.Endorse_From

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
var
            LicTC  tcursor
            licdlg   Form
            dlgVal,
            pers_no  String
            licloc   query
endVar
if not isblank(self.value) then
            licTC.open ("licences.db")
            licTC.locate("licence_no",licence_no.value)
            if endorse_from.value > licTC.expiry_date or endorse_from.value <licTC.valid_from then
                        eventinfo.seterrorcode(cannotdepart)
                        message ("The valid from date is not valid for this licence ", "Checking Licence
Number")
                        sleep (3000)
                        pers_no = personvalue.value
                        licloc = query

                        :fidms:licences.db  |Person_no        |Licence_no   |Valid_from      |Expiry_date  |
                                                |Check ~pers_no  |Check         |Check           |Check        |

                        endQuery

                        if not executeqbe(licloc, "licloc.db") then
                                    errorshow()
                        endif
                        if licDlg.open("licloc.fdl") then
                                    dlgVal =  licDlg.wait()
                                    if dlgVal= "OK" then
                                                licence_no.value = licDlg.licence_no
                                    endif
                        endif
            endif
endif
if not endorse_from.isblank() then
        endorse_to.value = endorse_from.value + 90
endif
endmethod
```

Object : termpage.Endorse_To

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
var
            LicTC tcursor
            licdlg Form
            dlgVal,pers_no String
            licloc query
endVar
if not isblank(self.value) then
            licTC.open ("licences.db")
            licTC.locate("licence_no",licence_no.value)
            if endorse_to.value > licTC.expiry_date or endorse_to.value <licTC.valid_from then
                        eventinfo.seterrorcode(cannotdepart)
                        message ("The valid from date is not valid for this licence ", "Checking Licence
Number")
                        sleep (3000)
                        pers_no = personvalue.value
                        licloc = query

                        :fidms:licences.db  |Person_no        |Licence_no   |Valid_from      |Expiry_date  |
                                                |Check ~pers_no    |Check         |Check           |Check        |
```

```
                endQuery
                if not executeqbe(licloc, "licloc.db") then
                        errorshow()
                endif
                if licDlg.open("licloc.fdl") then
                        dlgVal =  licDlg.wait()
                        if dlgVal= "OK" then
                                licence_no.value = licDlg.licence_no
                        endif
                endif
        endif
endif
endmethod
```

Object : termpage.END_LOC.#Record65.Block_code

MethodName : changeValue

Source :

```
method changeValue(var eventInfo ValueEvent)
        location_description.moveto()
endmethod
```

Object : termpage.END_SPEC.#Record163.Quantity

MethodName : canDepart

Source :

```
method canDepart(var eventInfo MoveEvent)
var
        pritabtc tcursor
        pritab   Table
endvar
pritab.attach (":pri:pri.db")
addpassword ("fidms")
pritabTC.open (":pri:pri.db")
if PritabTC.locate("TaxonID",species_code.value) then
        if not isblank(species_code.value) then
                msgInfo ("Warning - Do not issue", "This species is rare or restricted")
                eventinfo.seterrorcode(cannotdepart)
                species_code.color=red
                Genus.color=red
                Species.color=red
                beep()
        else
                species_code.color=White
                Genus.color=White
                Species.color=White
                dodefault
        endif
endif
endmethod
```

Object : termpage.END_SPEC.#Record163.Species

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
var
        SPMAS Tcursor
        HBTTRANSTC Tcursor
endvar
if isblank(species_code.value) then
        SPMAS.open (":fidms:hbttaxon.db")
        if spmas.locate ("genus",genus.value,"species",species.value) then
                if SPMAS.curnt = "Y" then
                        species_code.value=SPMAS.taxonId
                        genus.value = SPMAS.genus
                        species.value = SPMAS.species
                else
                        if SPMAS.curnt = "N" then
                                hbttransTC.open (":fidms:hbttrans.db")
                                if hbttransTC.locate ("oldId", spmas.taxonid) then
                                species_code.value = hbttranstc.newid
```

```
                                        genus.value = SPMAS.genus
                                        species.value = SPMAS.species
                                        infracat.value = SPMAS.infrasp_Rank
                                        infraspecies.value = SPMAS.infraSp_Name
                                msginfo ("Species Name Updated", "Revision of Taxonomy")
                        else
                                        msginfo ("Cannot find new taxon Id", "
                                          Check with Flora Industry Botanist")
                                endif
                        else
                                msginfo ("Species not valid", "Please Check in |Locate|Unknown
Taxon Id")
                        endif
                endif
        endif
endif
endmethod
```

Object : termpage.END_SPEC.#Record163.Species_code

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
var
        spMasTC TCursor
        hbttransTC TCursor
endvar
if not isblank(species_code.value) then
        spMasTC.open (":fidms:hbttaxon.db")
        spMasTC.locate ("taxonid",species_code)
        if spMasTC.curnt = "N" then
                message ("This taxon number is not current", "Program will update - please wait")
                sleep (3000)
                hbttransTC.open (":fidms:hbttrans.db")
                hbttransTC.locate("oldid",species_code)
                species_code.value = hbttranstc.newid
                spMasTC.locate ("taxonid",species_code)
                genus.value = spmasTC.genus
                species.value = spmasTC.species
                Infracat.value = spmasTC.infraSp_Rank
                Infraspecies.value = spmasTC.infrasp_Name
        else
                genus.value = spmasTC.genus
                species.value = spmasTC.species
                Infracat.value = spmasTC.infraSp_Rank
                Infraspecies.value = spmasTC.infrasp_Name
        endif
endif
endmethod
```

Object : mainpage

MethodName : Uses

Source :

```
Uses ObjectPal
        setCallFormName(const CallerTitle String, xPos LongInt, yPos LongInt)
        VCRisOpen()
        VCRisClosed()
        isVCROpen() Logical
        vcrName() String
        listPageObjects(formName String, pageName String) Logical
endUses
```

Object : mainpage

MethodName : Var

Source :

```
Var
        licTC,
        persTC,
        EndorseTC,
        floraretTC,
        endnoTC,
```

```
        TaxonIdTC  tCursor
        wtext,maintext array[12] String
endVar
```

Object : mainpage

MethodName : Const

Source :

```
Const
        Ends_New = 101 ; Endorsements | New Endorsement command
        Ends_Prt = 102 ; Endorsements | Print command
        EndExit  = 103 ; Endorsements | Exit command
        Ends_Renew = 104 ; Endorsements | Renew Endorsement command

        Edit_Cut = 201 ; Edit | Cut command
        EditCopy = 202 ; Edit | Copy command
        EditPast = 203 ; Edit | Paste command

        RecFirst = 301 ; Record | First command
        Rec_Prev = 302 ; Record | Previous command
        Rec_Next = 303 ; Record | Nextcommand
        Sect_Next = 309; Record | Next section
        Rec_Last = 304 ; Record | Last command
        Rec_Plus = 305 ; Record | Insert command
        Rec_Nuke = 306 ; Record | Delete command
        Rec_Canc = 307 ; Record | Cancel Changes command
        Rec_Look = 308 ; Record | LookUp Help command

        HelpHelp = 401 ; Help | Using Help command
        HelpList = 402 ; Help | Help Index command
        Help_Use = 403 ; Help | System Help command
        Prog_help = 404 ; Help | Programmer Help

        Florarets = 501 ; Other Modules | Flora Returns command
        Licences = 502 ; Other Modules | Licences command
        Statistics= 504 ; Other Modules | Statistics command
        Reports  = 503 ; Other Modules | Reports command

        Find_endorse = 601 ; Endorsements | Locate | Endorsement No
        FindDist = 602 ; Endorsements | Locate | District
        FindLic  = 603 ; Endorsements | Locate | Licence Number
        FindReg  = 604
        FindTaxon= 607 ; Endorsements | Locate | Taxon Id
        Find_Qty = 608 ; Endorsements | Locate | Quantity

        file_print = 800
        Exit    = 801 ; Exit command
        file_print_setup = 802

        #IDH_MENUS_WINDOW_CURRENTWINDOW = "Current Open Windows"
        #IDH_MENUS_WINDOWS = "&Window"
endConst
```

Object : mainpage

MethodName : open

Source :

```
method open(var eventInfo Event)
Message("Loading ENDORSFM.  One moment, please...")
delayScreenUpdates(Yes)
if not fidmslib.open("fidmslib.ldl", globalToDesktop) then
        msgStop("Failure", "fidmslib could not be opened")
endif
formName.attach()
hideSpeedBar()
maximize()
okToExit = False
delayScreenUpdates(No)
if not isFile("EndorsFM.fdl") then
        disableDefault
        beep()
        msgStop("Directory Error", "The Paradox for Windows " +
```

```
            "working directory must be set to the directory " +
            "containing this form, for example: " +
            "C:\\PDOXWIN5\\FLORADB.")
            close()
else
            blankAsZero(No)
            endnoTC.open("end_no.db")
            endnoTC.edit()
            fidmslib.open("fidmslib", globalToDesktop)
            okToExit = False
          doDefault
            wText[1]=#IDH_MENUS_WINDOW_CURRENTWINDOW
            mainText[5]=#IDH_MENUS_WINDOWS
            endorseTC.open(":fidms:endorse.db")
            endorseTC.edit()
            persTC.open (":FIDMS:persons.db")
            LicTC.open(":fidms:LICENCEs.DB")
            FloraretTC.open(":FIDMS:Floraret.DB")
            TaxonidTC.open(":FIDMS:HBTTAXON.DB")
            self.postAction(dataBeginEdit)
            delayScreenUpdates(No)
endIf
endmethod
```

Object : mainpage

MethodName : close

Source :

```
method close(var eventInfo Event)
if fidmslib.isVCROpen() then
            myBar.close()
            fidmslib.VCRisClosed()
endif
endmethod
```

Object : mainpage

MethodName : arrive

Source :

```
method arrive(var eventInfo MoveEvent)
var
            mainMenu        Menu    ; The main menu
            EndorsementMenu,        ; The Orders popup menu
            EditMenu,       ; The Edit popup menu
            Rec_Menu,               ; The Record menu
            filemenu,
            ModuleMenu,             ; The Module menu
            HelpMenu,       ; The Help menu
            windowmenu,
            FindMenu        popupMenu  ; The Endorsement | Locate menu

endVar

 fileMenu.addtext ("&Print", Menuenabled, Usermenu + file_print)
 fileMenu.addtext ("Printer &Setup", MenuEnabled, Usermenu + File_print_setup)
 filemenu.addtext ("E&xit", menuEnabled, userMenu + exit)
 mainmenu.addpopup ("&File", filemenu)

 FindMenu.addText("&Endorsement Number", menuEnabled, userMenu + find_endorse)
 findMenu.addtext("&Region", menuEnabled, userMenu + findReg)
 FindMenu.addText("&District", menuEnabled, userMenu + findDist)
 FindMenu.addSeparator()
 FindMenu.addText("&Licence Number", menuEnabled, userMenu + findLic)

 EndorsementMenu.addPopUp("&Locate", FindMenu)
 EndorsementMenu.addSeparator()
 EndorsementMenu.addText("&New Endorsement", menuEnabled, userMenu + Ends_new)
 EndorsementMenu.addText("&Renew Endorsement", menuEnabled, userMenu+ Ends_renew)
 MainMenu.addPopUp("&Endorsements", EndorsementMenu)

 EditMenu.addText("Cu&t\tShift+Del", menuDisabled + menuGrayed, userMenu + edit_Cut)
 EditMenu.addText("&Copy\tCtrl+Ins", menuDisabled + menuGrayed, userMenu + editCopy)
```

```
EditMenu.addText("&Paste\tShift+Ins", menuDisabled + menuGrayed, userMenu + editPast)
MainMenu.addPopUp("&Edit", EditMenu)

Rec_Menu.addText("&First\tCtrl+F11", menuEnabled, userMenu + recFirst)
Rec_Menu.addText("&Previous\tF11", menuEnabled, userMenu + rec_Prev)
Rec_Menu.addText("&Next\tF12", menuEnabled, userMenu + rec_Next)
Rec_Menu.addText("Next &Section\tF4", menuEnabled, userMenu + Sect_next)
Rec_Menu.addText("&Last\tCtrl+F12", menuEnabled, userMenu + rec_Last)
Rec_Menu.addSeparator()
Rec_Menu.addText("Lookup &Help\tCtrl+Space", menuEnabled, userMenu + rec_Look)
Rec_Menu.addText("&Insert\tInsert", menuEnabled, userMenu + rec_Plus)
Rec_Menu.addText("&Delete\tCtrl+Del", menuEnabled, userMenu + rec_Nuke)
Rec_Menu.addSeparator()
Rec_Menu.addText("&Cancel Changes\tAlt+BkSp", menuEnabled, userMenu + rec_Canc)
MainMenu.addPopUp("&Record", Rec_Menu)

HelpMenu.addText("Using &Help\tCtrl+F1", menuEnabled, userMenu + helpHelp)
HelpMenu.addText("Help &Index\tShift+F1", menuEnabled, userMenu + helpList)
HelpMenu.addText("&Using this Form\tF1", menuEnabled, userMenu + help_Use)
HelpMenu.addText("&Programmer Help", menuEnabled, userMenu + Prog_help)
MainMenu.addPopUp("&Help", HelpMenu)

ModuleMenu.addtext ("&Licences", menuEnabled, userMenu + Licences)
ModuleMenu.addtext ("&Flora Returns", menuEnabled, userMenu + Florarets)
ModuleMenu.addtext ("&Reports", menuEnabled, userMenu + Reports)
ModuleMenu.addtext ("&Statistics", menuEnabled, userMenu + Statistics)

MainMenu.addPopUp("&Other Modules", ModuleMenu)

windowmenu.addStaticText(wText[1])
mainMenu.addPopUp(mainText[5],windowmenu)

mainMenu.show()
endmethod
```

Object : mainpage

MethodName : mouseExit

Source :

```
method mouseExit(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio) ; what is the object?
        if uio.class <> "Text" and uio.class <> "Bitmap" then
                message("")
        endif
endif
endmethod
```

Object : mainpage

MethodName : action

Source :

```
method action(var eventInfo ActionEvent)
        setTitle("CALM 727 Endorsement")
endmethod
```

Object : mainpage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
var
        menu_Cmd        smallInt ; The constant returned by a menu command
        editTest        Logical  ; Flag variable used to test the type of
                                 ; object that is active
        findTest        String   ; Flag variable that holds the results of
                                 ; a search attempt
        calledBy        Form     ; Holds the handle of the form that opened
                                 ; this form
endVar
menu_Cmd = eventInfo.id()
switch
        case menu_Cmd = menuInit :
```

```
        try
        editTest = active.Editing
        onFail
        editTest = False
        endTry
        if editTest then
        if active.selectedText <> "" then
                        setMenuChoiceAttributeByID(userMenu + edit_Cut, menuEnabled)
                        setMenuChoiceAttributeByID(userMenu + editCopy, menuEnabled)
        else
                        setMenuChoiceAttributeByID(userMenu + edit_Cut, menuDisabled +
menuGrayed)
                        setMenuChoiceAttributeByID(userMenu + editCopy, menuDisabled +
menuGrayed)
        endIf
        endIf
        case menu_Cmd = MenuControlClose :
        disableDefault
        exitbutton.pushButton()
        case menu_Cmd = MenuControlMaximize or
        menu_Cmd = MenuControlMinimize :
        If formcaller(calledBy) then
        disableDefault
        beep()
        msgInfo("Oops!", "Because this form was opened by the " +
                                calledBy.getTitle() + " form, the Maximize and " +
                                "Minimize buttons are disabled.  Sorry...")
        endIf
        case menu_Cmd = userMenu + Find_endorse : end_find.pushbutton()
        case menu_Cmd = userMenu + FindLic  : licence_find.pushbutton()
        case menu_Cmd = userMenu + Ends_Prt : printBtn.pushButton()
        case menu_Cmd = userMenu + Exit  : exitbutton.pushButton()
        case menu_cmd = usermenu + file_print_setup: active.menuAction(menuFilePrinterSetup)
        case menu_Cmd = userMenu + Edit_Cut :
        active.menuAction(menuEditCut)
        setMenuChoiceAttributeByID(userMenu + editPast, menuEnabled)
        case menu_Cmd = userMenu + EditCopy :
        active.menuAction(menuEditCopy)
        setMenuChoiceAttributeByID(userMenu + editPast, menuEnabled)
        case menu_Cmd = userMenu + EditPast : active.menuAction(menuEditPaste)
        case menu_Cmd = userMenu + RecFirst : active.postAction(dataBegin)
        case menu_Cmd = userMenu + Rec_Prev : active.postAction(dataPriorRecord)
        case menu_Cmd = userMenu + Rec_Next : active.postAction(dataNextRecord)
        case menu_Cmd = userMenu + Rec_Last : active.postAction(dataEnd)
        case menu_Cmd = userMenu + Rec_Plus : Action(dataInsertRecord)
        case menu_Cmd = userMenu + Rec_Nuke : active.postAction(dataDeleteRecord)
        case menu_Cmd = userMenu + Rec_Canc : active.postAction(dataCancelRecord)
        case menu_Cmd = userMenu + Rec_Look : active.postAction(dataLookup)
        case menu_Cmd = userMenu + Florarets  : Floraret.pushbutton()
        case menu_Cmd = userMenu + Licences   : Licencebutton.pushbutton()
        case menu_Cmd = userMenu + Findlic   : licence_find.pushbutton()
        case menu_Cmd = userMenu + Statistics : processButton.pushbutton()
        case menu_Cmd = userMenu + Reports   : reportprocessbutton.pushbutton()
        case menu_Cmd = userMenu + ends_new  : new_endorse.pushbutton()
        case menu_Cmd = userMenu + ends_renew : renew()
        case menu_Cmd = userMenu + HelpHelp  : helpOnHelp()
        case menu_Cmd = userMenu + HelpList : helpShowContext(helpFile,1000)
        case menu_Cmd = userMenu + Prog_help  : helpShowindex(progHelp)
        case menu_Cmd = userMenu + Help_Use : helpShowContext(helpFile, headerHelp)
endswitch
endmethod
```

Object : mainpage.processButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(statsFM) then
        statsFM.bringtoTop()
        statsFM.wait()  ; put statsFM into wait state
        statsFM.hide()
```

```
        mainpage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if statsFM.Open("Stats.fdl", WinStyleMaximize) then
                statsFM.Wait()  ; put statsFM into wait state
                statsFM.hide()
                maximize()
                mainpage.moveTo()
        else
                msgInfo("Status", "Sorry, the Statistics form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.end_find

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        endNo Number
        enduio UIObject
endVar
endNo  = 0
endNo.view ("Enter an Endorsement Number")
enduio.attach(Endorsement_no)
if endNo <> 0 then
        if enduio.locate("Endorsement_No",endNo) then
                beep()
        else
                msginfo ("Search unsuccessful", "Couldn't locate Endorsement No " + String(EndNo))
        endif
endif
endmethod
```

Object : mainpage.district_find

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        dis String
        disuio uiobject
        userchoice String
        numfound smallint
        olesound OLE
endvar
ignorecaseinlocate(yes)
dis  = ""
dis.view ("Enter a CALM district")
disuio.attach(district)
if dis <> "" then
        if disuio.locate("District",dis) then
                beep()
                numfound = 1
                message ("")
                userchoice = msgYesNoCancel ("Choose", "Do you wish to see next " + String(dis))
                switch
                case userChoice = "Yes" :
                        while disuio.locatenext("District",dis)
                        beep()
                        numfound = numfound + 1
                        userchoice = msgYesNoCancel ("Choose",
                                "Do you wish to see next " + String(dis))
                        switch
                                        case userChoice = "Yes" :
                                        message ("Searching")
                                        sleep (1000)
                                        if not disuio.locatenext("District",dis) then
                                                make_music()
                                                msginfo ("No further records - Found " + dis,
```

```
                                                    strval(Numfound) + " times.")
                                                    return
                                            endif
                                            loop
                                            otherwise : msgInfo("Found " + dis, strval(numFound) + "
times.")
                                    return
                            endswitch
                            endwhile
                            otherwise : msgInfo("Found " + dis, strval(numFound) + " times.")
                    endswitch
            else
                    msgInfo("No record found for " + dis, "Check spelling")
            endif
endif
endmethod
```

Object : mainpage.district_find

MethodName : make_music

Source :

```
method make_music()
var
        quarterNote, octave, note LongInt
        power               Number
        olevar OLE
endVar
const
 noteA1  = 110
 noteA#1 = 116
 noteB1  = 123
 noteC1  = 130
 noteC#1 = 138
 noteD1  = 146
 noteD#1 = 155
 noteE1  = 164
 noteF1  = 174
 noteF#1 = 184
 noteG1  = 195
 noteG#1 = 207
 noteA2  = 220
 noteA#2 = 234
 noteB2  = 249
 noteC2  = 265
 noteC#2 = 282
 noteD2  = 300
endConst

sound(noteA1, 200)
sound(noteD1, 150)
sound(noteF#1, 50)
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteA2, 150)
sound(noteF#1, 50)
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteC#2, 150)
sound(noteD2, 50)
sound(noteA2, 100)
sound(noteF#1, 100)
sound(noteD1, 100)
sleep(1000)
endMethod
```

Object : mainpage.reg_find

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        reg String
```

```
        reguio uiobject
        userchoice String
        numfound smallint
        olesound OLE
endvar
ignorecaseinlocate(yes)
reg  = ""
reg.view ("Enter a CALM Region")
reguio.attach(region)
if reg <> "" then
        if reguio.locate("Region",reg) then
                beep()
                numfound = 1
                message ("")
                userchoice = msgYesNoCancel ("Choose", "Do you wish to see next " + String(reg))
                switch
                case userChoice = "Yes" :
                        while reguio.locatenext("Region",reg)
                        beep()
                        numfound = numfound + 1
                        userchoice = msgYesNoCancel ("Choose",
                                "Do you wish to see next " + String(reg))
                        switch
                                        case userChoice = "Yes" :
                                        message ("Searching")
                                        sleep (1000)
                                        if not reguio.locatenext("Region",reg) then
                                                make_music()
                                                msginfo ("No further records - Found " + reg,
                                                  strval(Numfound) + " times.")
                                                return
                                        endif
                                        loop
                                        otherwise : msgInfo("Found " + reg, strval(numFound) + "
times.")
                                return
                        endswitch
                        endwhile
                        otherwise : msgInfo("Found " + reg, strval(numFound) + " times.")
                endswitch
        else
                msgInfo("No record found for " + reg, "Check spelling")
        endif
endif
endmethod
```

Object : mainpage.reg_find

MethodName : make_music

Source :

```
method make_music()
var
        quarterNote, octave, note LongInt
        power               Number
        olevar OLE
endVar
const
 noteA1  = 110
 noteA#1 = 116
 noteB1  = 123
 noteC1  = 130
 noteC#1 = 138
 noteD1  = 146
 noteD#1 = 155
 noteE1  = 164
 noteF1  = 174
 noteF#1 = 184
 noteG1  = 195
 noteG#1 = 207
 noteA2  = 220
 noteA#2 = 234
```

```
 noteC2  = 265
 noteC#2 = 282
 noteD2  = 300
endConst

sound(noteA1, 200)
sound(noteD1, 150)
sound(noteF#1, 50)
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteA2, 150)
sound(noteF#1, 50)
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteC#2, 150)
sound(noteD2, 50)
sound(noteA2, 100)
sound(noteF#1, 100)
sound(noteD1, 100)
sleep(1000)
endMethod
```

Object : mainpage.licence_find

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        lic String
        lic_num uiobject
        userchoice String
        numfound smallint
        olesound OLE
endvar
ignorecaseinlocate(yes)
lic  = ""
lic.view ("Enter a licence Number")
lic_num.attach(licence_no)
if lic <> "" then
        if lic_num.locate("Licence_no",lic) then
                beep()
                numfound = 1
                message ("")
                userchoice = msgYesNoCancel ("Choose", "Do you wish to see next " + String(lic))
                switch
                case userChoice = "Yes" :
                        while lic_num.locatenext("Licence_no",lic)
                        beep()
                        numfound = numfound + 1
                        userchoice = msgYesNoCancel ("Choose",
                                "Do you wish to see next " + String(lic))
                        switch
                                        case userChoice = "Yes" :
                                        message ("Searching")
                                        sleep (1000)
                                        if not lic_num.locatenext("Licence_no",lic) then
                                                make_music()
                                                msginfo ("No further records - Found " + lic,
                                                  strval(Numfound) + " times.")
                                                return
                                        endif
                                        loop
                                        otherwise : msgInfo("Found " + lic, strval(numFound) + "
times.")
                                return
                        endswitch
                        endwhile
                        otherwise : msgInfo("Found " + lic, strval(numFound) + " times.")
                endswitch
```

```
                msgInfo("No record found for " + lic, "Check spelling")
        endif
endif
endmethod
```

Object : mainpage.licence_find

MethodName : make_music

Source :

```
method make_music()
var
        quarterNote, octave, note LongInt
        power               Number
        olevar OLE
endVar

const
 noteA1  = 110
 noteA#1 = 116
 noteB1  = 123
 noteC1  = 130
 noteC#1 = 138
 noteD1  = 146
 noteD#1 = 155
 noteE1  = 164
 noteF1  = 174
 noteF#1 = 184
 noteG1  = 195
 noteG#1 = 207
 noteA2  = 220
 noteA#2 = 234
 noteB2  = 249
 noteC2  = 265
 noteC#2 = 282
 noteD2  = 300
endConst

sound(noteA1, 200)
sound(noteD1, 150)
sound(noteF#1, 50)
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteA2, 150)
sound(noteF#1, 50)
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteC#2, 150)
sound(noteD2, 50)
sound(noteA2, 100)
sound(noteF#1, 100)
sound(noteD1, 100)
sleep(1000)
endMethod
```

Object : mainpage.reportprocessbutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(reportsFm) then
        reportsFm.bringtoTop()
        reportsFm.wait()  ; put reportsFm into wait state
        reportsFm.hide()
        maximize()
        mainpage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if reportsFm.Open("report.fdl", WinStyleMaximize) then
                reportsFm.Wait()  ; put reportsFm into wait state
                reportsFm.hide()
                maximize()
                mainpage.moveTo()
```

```
        else
                msgInfo("Status", "Sorry, the Reports form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.licencebutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(licenceFm) then
        licenceFm.bringtoTop()
        licenceFm.wait()  ; put licenceFm into wait state
        licenceFm.hide()
        maximize()
        mainpage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if licenceFm.Open("licence.fdl", WinStyleMaximize) then
                licenceFm.Wait()  ; put licenceFm into wait state
                licenceFm.hide()
                maximize()
                mainpage.moveTo()
        else
                msgInfo("Status", "Sorry, the licence form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.floraret

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(floraretFm) then
        floraretFm.bringtoTop()
        floraretFm.wait()  ; put floraretFm into wait state
        floraretFm.hide()
        maximize()
        mainpage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if floraretFm.Open("floraret.fdl", WinStyleMaximize) then
                floraretFm.Wait()  ; put floraretFm into wait state
                floraretFm.hide()
                maximize()
                mainpage.moveTo()
        else
                msgInfo("Status", "Sorry, the floraret form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.new_endorse

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
action(datainsertrecord)
if isblank(endorsement_no.value) then
        if endnoTC.lockrecord() then
                endorsement_no.value = endnoTC.cmax("endorse_no") + 1
                endnoTC.endorse_no = endorsement_no.value
                endnoTC.unlockrecord()
                region.moveto()
        else
                msgstop ("Problem", "Couldn't lock the end_no table")
        endif
endif
```

```
endmethod
```

Object : mainpage.exitButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
choice = msgQuestion("Quit F*I*D*M*S", "Do you want to quit " + "the module?")
if choice= "Yes" then
        okToExit = True
        close()
else
        eventInfo.setErrorCode(cannotDepart)
endif
endmethod
```

Object : mainpage.printBtn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        endtv Tableview
        endqry Query
        userin String
        endorse Report
        repInfo ReportPrintInfo
        theChoice,useritem String
endvar
userin = ""
userin.view ("Enter licence no, blank is current endorsement")
if userin <> "" then
        myitem.value = userin
        useritem = myitem.value
        endqry = query

        persons.db   |person_no    |Surname|Title|forename  |Address1|Address2|Address3|Postcode|
                |_abcd       |Check  |Check|Check    |Check   |Check   |Check   |Check   |

        licences.db  |Licence_no   |person_no|
             |~useritem, _bcd|_abcd   |

        Endorse.db  |Licence_no   |Endorsement_no|Issue_officer|Region|District|Endorse
from|Endorse to|
                |Check _bcd   |Check       |Check       |Check |Check   |Check      |Check    |

        endQuery

else
        if userin = "" then
                useritem = endorsement_no.value

                endqry = query

                persons.db   |person_no |Surname|Title|forename
|Address1|Address2|Address3|Postcode|
                        |_abcd       |Check  |Check|Check    |Check   |Check   |Check   |Check   |

                licences.db  |person_no|Licence_no|
                        |_abcd   |_lic    |

                Endorse.db  |Licence_no|Endorsement_no |Issue_officer|Region|District|Endorse
from|Endorse to|
                        |Check _lic  |Check ~useritem |Check       |Check |Check |Check   |Check    |

                endQuery
        endif
endif
if not executeqbe(endqry,"endprn.db") then
        errorshow()
endif
theChoice = msgYesNoCancel ("Reports","Do you wish to print endorsement?")
```

```
switch
        case theChoice = "Yes" : endorse.open("endorse.rdl")
        replnfo.nCopies = 2
        replnfo.makecopies = true
        endorse.print(replnfo)
        otherwise           :return
endswitch
endmethod
```

Object : mainpage.helpButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        helpShowContext(helpFile, 20006)
endmethod
```

Object : mainpage.#Text14.Licence_no

MethodName : setFocus

Source :

```
method setFocus(var eventInfo Event)
        self.color = lightBlue
endmethod
```

Object : mainpage.#Text14.Licence_no

MethodName : removeFocus

Source :

```
method removeFocus(var eventInfo Event)
        self.color = white
endmethod
```

Object : mainpage.#Text14.Licence_no

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
        var licTC tCursor
        persTC Tcursor
endvar
if not isblank(licence_no.value) then
        licTC.open ("licences.db")
        persTC.open ("persons.db")
        if licTC.locate("licence_no",licence_no.value) then
                personvalue.value = licTC.person_no
                persTC.locate("person_no",personvalue.value)
                surname.value = persTC.surname
                forename.value = persTC.forename
                telephone.value = persTC.home_phone_no
                address1.value = persTC.address1
                address2.value = persTC.address2
                address3.value = persTC.address3
                postcode.value = persTC.postcode
        else
                msgstop ("Warning", "No record of any person for this licence")
        endif
endif
endmethod
```

Object : mainpage.Region

MethodName : setFocus

Source :

```
method setFocus(var eventInfo Event)
        self.color = red
endmethod
```

Object : mainpage.Region

MethodName : removeFocus

Source : method removeFocus(var eventInfo Event)

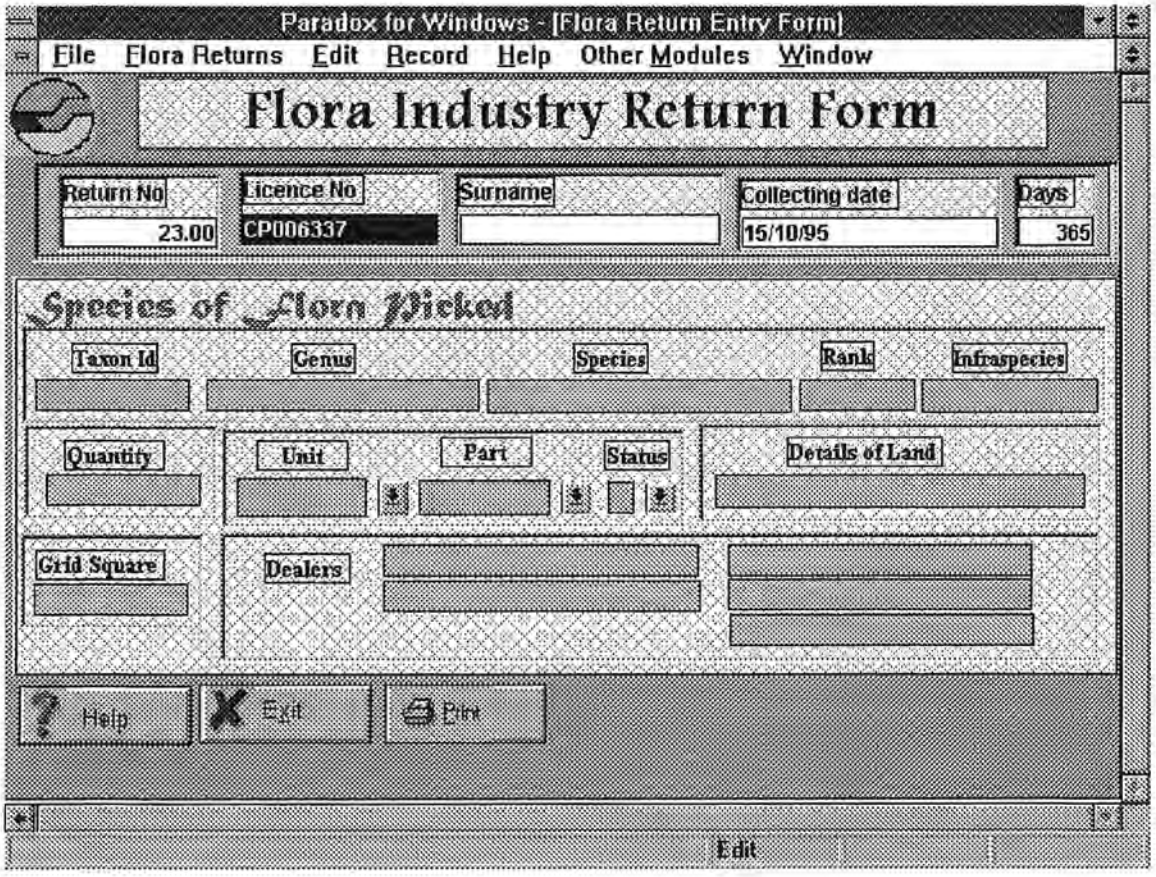
	endmethod

Object :	mainpage.District
MethodName :	setFocus
Source :	method setFocus(var eventInfo Event) self.color = red endmethod

Object :	mainpage.District
MethodName :	removeFocus
Source :	method removeFocus(var eventInfo Event) self.color = white endmethod

Flora Industry Return Form

The Flora Industry Return Form is used to enter, validate and look up data provided by licensed commercial flora pickers on species, quantities and areas of harvest. Shows return details by month for a person and licence and species which were harvested for that return. Allows data entry of these from flora return forms submitted to SOHQ. The form appears as shown below (but in colour).



FLORARET.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Methods (FLORARET.FSL)
Uses
Var
Const

close
keyPhysical
menuAction
mouseEnter
mouseExit
mouseRightUp

Objects
mainPage

Methods (mainPage)
Uses
Var
Const

action
arrive
clearPageObjects
close
findretBttn
licence_find
make_music
menuAction
new_return
newspecies
open
ret_find

Objects (mainPage)
endorseBttn
helpbutton
licence_button
printBtn
processButton
reportprocessButton
exitButton

Fields (mainPage)
Crown-private
collecting_date
Grid Square
infraSpName
licence_number
Taxon Id

Methods (Collecting_date)
depart

Methods (Crown_private)
depart

Methods (Grid_Square)
depart

Methods (infraSpName)
depart

Methods (Licence_number)
depart

Methods (Taxon Id)
depart

Object : floraret

MethodName : Uses

Source :
```
Uses ObjectPal
        clearPageValues(formName String, pageName String)
        createObjMenu(var formName Form, pageName String, uio UIObject)
        FidmsObjMsg(formName String, pageName String, objID String)
        pageNameOf(ui UIObject) String
        setIsCalledTrue()
        setIsCalledFalse()
        formCanClose()
endUses
```

Object : floraret

MethodName : Var

Source :
```
Var
        uioName, choice String
        uio, retnum UIObject
        formName, floradbFm, floraretFm, licenceFm, reportsFM, endorseFM, aboutFm,statsFM Form
        fidmslib Library
        closeVCRLg Logical
        okToExit Logical
        myBar Form
        wasCalled Logical
        newVal String
        ses   Session
endVar
```

Object : floraret

MethodName : Const

Source :
```
Const
        helpFile  = ":fidms:floradb.hlp" ; path to Help file
        proghelp  = ":fidms:fidprog.hlp"
        headerHelp = LongInt(20007)
        detailHelp = LongInt(7)
endConst
```

Object : floraret

MethodName : open

Source :
```
method open(var eventInfo Event)
if eventInfo.isPreFilter() then
        ses.open ()
endif
endmethod
```

Object : floraret

MethodName : close

Source :
```
method close(var eventInfo Event)
if eventInfo.isPrefilter() then
        doDefault
else
        if floradbFm.isAssigned() then
                floradbFm.close()
        endif
endif
endmethod
```

Object : floraret

MethodName : mouseEnter

Source :
```
method mouseEnter(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio)  ; what is the object?
```

```
        uioName = uio.Name
        if uio.name <> self.name AND (uio.class = "Field"
          OR uio.class = "Multirecord" OR uio.class = "Button") then
                fidmslib.FIDMSObjMsg(formName.name, fidmslib.pageNameOf(uio), uioName)
        endif
endif
endmethod
```

Object : floraret

MethodName : mouseExit

Source :

```
method mouseExit(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then  ; Execute for each object on form.
        message("")
else
endIf
endMethod
```

Object : floraret

MethodName : mouseRightUp

Source :

```
method mouseRightUp(var eventInfo MouseEvent)
var
  uiTarget      uiObject; The object that was right-clicked
  thisForm      Form    ; Form variable for fidmsLIB.createObjMenu()
  Obj_Name      String  ; Used to make sure we search for the
                        ; right object when we call createObjMenu.
endVar
  if eventInfo.isPreFilter() then
        eventInfo.getTarget(uiTarget)
        obj_Name = uiTarget.Name
        while obj_Name.subStr(1, 1) = "#" and uiTarget.containerName <> ""
                uiTarget.attach(uiTarget.containerName)
                obj_Name = uiTarget.Name
        endWhile
        thisForm.attach()
        fidmsLib.createObjMenu(thisForm, "mainpage", uiTarget)
        if uiTarget.class = "Field" then
                if uiTarget.tabStop then
                self.postAction(uiTarget.moveTo())
                endIf
        endIf
else
endIf
endMethod
```

Object : floraret

MethodName : keyPhysical

Source :

```
method keyPhysical(var eventInfo KeyEvent)
var
        theKey String
        thePage String
endvar
if eventInfo.isPreFilter() then
        theKey = eventInfo.vChar()  ; tell me what key was pressed
        if eventInfo.isAltKeyDown() then
                switch
                        case theKey = "E" or theKey = "e" : disableDefault
                        mainpage.exitButton.pushButton()
                        case theKey = "N" or theKey = "n" : disableDefault
                        mainpage.new_return.pushbutton()
                        case theKey = "R" or theKey = "r" : disableDefault
                        mainpage.reportprocessbutton.pushbutton()
                        case theKey = "X" or theKey = "x" : disableDefault
                        mainpage.exitButton.pushButton()
                        otherwise: doDefault
                endswitch
        else
                switch
```

```
                        case theKey = "VK_F1" : disableDefault
                        helpShowContext(helpfile, detailhelp)
                        otherwise : doDefault
                endSwitch
        endif
endif
endmethod
```

Object : floraret

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
 if not eventInfo.isPreFilter() then
        if eventInfo.id() = MenuControlMaximize or eventInfo.id() = MenuControlMinimize then
        disableDefault
        beep()
        message("Form cannot be resized.")
        else
                if eventInfo.reason() = menuNormal and eventInfo.id() = MenuCanClose
                  and not okToExit then
                        eventInfo.setErrorCode(CanNotDepart)
                        disableDefault
                        message("Use the Exit button to exit Flora Return Entry Form.")
                endIf
        endIf
endIf
endmethod
```

Object : mainpage

MethodName : Uses

Source :

```
Uses ObjectPal
        setCallFormName(const CallerTitle String, xPos LongInt, yPos LongInt)
        VCRisOpen()
        VCRisClosed()
        isVCROpen() Logical
        vcrName() String
        listPageObjects(formName String, pageName String) Logical
endUses
```

Object : mainpage

MethodName : Var

Source :

```
Var
        spMasTC TCursor
        retnoTC Tcursor
        retnumTC Tcursor
        retexTC Tcursor
        wtext,maintext array[12] String
endVar
```

Object : mainpage

MethodName : Const

Source :

```
Const
  Ends_New = 101 ; Returns | New Return command
  Findret  = 102 ; Returns | Return status
  Ends_Prt = 103 ; Returns | Print command
  file_print_setup = 105
  EndExit  = 104 ; Returns | Exit command

  Edit_Cut = 201 ; Edit | Cut command
  EditCopy = 202 ; Edit | Copy command
  EditPast = 203 ; Edit | Paste command

  RecFirst = 301 ; Record | First command
  Rec_Prev = 302 ; Record | Previous command
  Rec_Next = 303 ; Record | Next command
```

```
Rec_Nuke = 306 ; Record | Delete command
Rec_Canc = 307 ; Record | Cancel Changes command
Rec_Look = 308 ; Record | LookUp Help command

HelpHelp = 401 ; Help | Using Help command
HelpList = 402 ; Help | Help Index command
Help_Use = 403 ; Help | System Help command
Prog_help =404 ; Help | Programmer Help

Endments = 501 ; Other Modules | Endorsements command
Licences = 502 ; Other Modules | Licences command
Reports  = 503 ; Other Modules | Reports command
Stats    = 504 ; Other Modules | Statistics command

Find_ret = 601 ; Returns | Locate | Return No
FindGrid = 602 ; Returns | Locate | Grid Square
 FindLic  = 603 ; Returns | Locate | Licence Number
FindName = 604 ; Returns | Locate | Surname
FindTaxon= 607 ; Returns | Locate | Taxon Id
Find_Qty = 608 ; Returns | Locate | Quantity
get_spec = 609 ; Returns | Locate | Unknown Taxon Id

Exit    = 701 ; Exit command

#IDH_MENUS_WINDOW_CURRENTWINDOW = "Current Open Windows"
#IDH_MENUS_WINDOWS = "&Window"
```

Object : mainpage

MethodName : open

Source :

```
method open(var eventInfo Event)

Message("Loading FLORARET.  One moment, please...")
delayScreenUpdates(Yes)
if not fidmslib.open("fidmslib.lsl", globalToDesktop) then
        msgStop("Failure", "fidmslib could not be opened")
endif
formName.attach()
hideSpeedBar()
maximize()
okToExit = False
delayScreenUpdates(No)
if not isFile("floraret.fsl") then
        disableDefault
        beep()
        msgStop("Directory Error", "The Paradox for Windows " +
          "working directory must be set to the directory " +
          "containing this form, for example: " +
          "C:\\PDOXWIN5\\FLORADB.")
        close()
else
        blankAsZero(No)
        retnoTC.open("retno.db")
        retnoTC.edit()
        fidmslib.open("fidmslib", globalToDesktop)
        okToExit = False
        doDefault
        wText[1]=#IDH_MENUS_WINDOW_CURRENTWINDOW
        mainText[5]=#IDH_MENUS_WINDOWS
        self.postAction(dataBeginEdit)
        taxonid.color = Gray
        Genus.color = Gray
        Species.color = Gray
        delayScreenUpdates(No)
 endIf
endmethod
```

Object : mainpage

MethodName : close

Source :
```
method close(var eventInfo Event)
if fidmslib.isVCROpen() then
        yBar.close()
        idmslib.VCRisClosed()
endif
endmethod
```

Object : mainpage

MethodName : arrive

Source :
```
method arrive(var eventInfo MoveEvent)
var
        mainMenu        Menu    ; The main menu
        FloraRetMenu,           ; The Flora Return popup menu
        EditMenu,       ; The Edit popup menu
        Rec_Menu,               ; The Record menu
        ModuleMenu,             ; The Module menu
        filemenu,               ; The File menu
        HelpMenu,       ; The Help menu
        windowmenu,
        FindMenu popupMenu      ; The Endorsement | Locate menu
endVar

 fileMenu.addText("&Print", menuEnabled, userMenu + ends_Prt)
 fileMenu.addtext ("Printer &Setup", MenuEnabled, Usermenu + File_print_setup)
 filemenu.addtext ("E&xit", menuEnabled, userMenu + exit)
 mainmenu.addpopup ("&File", filemenu)

 FindMenu.addText("&Floraret Number", menuEnabled, userMenu + find_ret)
 FindMenu.addSeparator()
 FindMenu.addText("&Licence Number", menuEnabled, userMenu + findLic)
 FindMenu.addSeparator()
 FindMenu.addText("&Unknown Taxon Id", menuEnabled, userMenu + get_spec)

 FloraRetMenu.addPopUp("&Locate", FindMenu)
 FloraRetMenu.addText("&Return Status",menuEnabled, userMenu + Findret)

 FloraRetMenu.addSeparator()
 FloraRetMenu.addText("&New Return", menuEnabled, userMenu + Ends_new)
 FloraRetMenu.addSeparator()
 FloraRetMenu.addText("E&xit", menuEnabled, userMenu + EndExit)
 MainMenu.addPopUp("&Flora Returns", FloraRetMenu)

 EditMenu.addText("Cu&t\tShift+Del", menuDisabled + menuGrayed, userMenu + edit_Cut)
 EditMenu.addText("&Copy\tCtrl+Ins", menuDisabled + menuGrayed, userMenu + editCopy)
 EditMenu.addText("&Paste\tShift+Ins", menuDisabled + menuGrayed, userMenu + editPast)
 MainMenu.addPopUp("&Edit", EditMenu)

 Rec_Menu.addText("&First\tCtrl+F11", menuEnabled, userMenu + recFirst)
 Rec_Menu.addText("&Previous\tF11", menuEnabled, userMenu + rec_Prev)
 Rec_Menu.addText("&Next\tF12", menuEnabled, userMenu + rec_Next)
 Rec_Menu.addText("&Last\tCtrl+F12", menuEnabled, userMenu + rec_Last)
 Rec_Menu.addSeparator()
 Rec_Menu.addText("Lookup &Help\tCtrl+Space", menuEnabled, userMenu + rec_Look)
 Rec_Menu.addText("&Insert\tInsert", menuEnabled, userMenu + rec_Plus)
 Rec_Menu.addText("&Delete\tCtrl+Del", menuEnabled, userMenu + rec_Nuke)
 Rec_Menu.addSeparator()
 Rec_Menu.addText("&Cancel Changes\tAlt+BkSp", menuEnabled, userMenu + rec_Canc)
 MainMenu.addPopUp("&Record", Rec_Menu)

 HelpMenu.addText("Using &Help\tCtrl+F1", menuEnabled, userMenu + helpHelp)
 HelpMenu.addText("Help &Index\tShift+F1", menuEnabled, userMenu + helpList)
 HelpMenu.addText("&Using this Form\tF1", menuEnabled, userMenu + help_Use)
 HelpMenu.addText("&Programmer Help", menuEnabled, userMenu + Prog_help)
 MainMenu.addPopUp("&Help", HelpMenu)

 ModuleMenu.addtext ("&Endorsements", menuEnabled, userMenu + Endments)
 ModuleMenu.addtext ("&Statistics", menuEnabled, userMenu + Stats)
 ModuleMenu.addtext ("&Reports", menuEnabled, userMenu + Reports)
```

```
ModuleMenu.addtext ("&Licences", menuEnabled, userMenu + Licences)
MainMenu.addpopup ("Other &Modules", Modulemenu)

windowmenu.addStaticText(wText[1])
mainMenu.addPopUp(mainText[5],windowmenu)

mainMenu.show()
endmethod
```

Object : mainpage

MethodName : action

Source :

```
method action(var eventInfo ActionEvent)
        setTitle("Flora Return Entry Form")
endmethod
```

Object : mainpage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
var
        menu_Cmd        smallInt ; The constant returned by a menu command
        editTest        Logical  ; Flag variable used to test the type of
                                 ; object that is active
        findTest        String   ; Flag variable that holds the results of
                                 ; a search attempt
        calledBy        Form     ; Holds the handle of the form that opened
                                 ; this form
endVar
menu_Cmd = eventInfo.id()
switch
        case menu_Cmd = menuInit :
        try
        editTest = active.Editing
        onFail
        editTest = False
        endTry
        if editTest then
        if active.selectedText <> "" then
                        setMenuChoiceAttributeByID(userMenu + edit_Cut, menuEnabled)
                        setMenuChoiceAttributeByID(userMenu + editCopy, menuEnabled)
        else
                        setMenuChoiceAttributeByID(userMenu + edit_Cut, menuDisabled +
menuGrayed)
                        setMenuChoiceAttributeByID(userMenu + editCopy, menuDisabled +
menuGrayed)
        endIf
        endIf
        case menu_Cmd = MenuControlClose :
        disableDefault
        exitbutton.pushButton()
        case menu_Cmd = MenuControlMaximize or
        menu_Cmd = MenuControlMinimize :
        if formcaller(calledBy) then
                disableDefault
                beep()
                msgInfo("Oops!", "Because this form was opened by the " +
                        "calledBy.getTitle() + " form, the Maximize and " +
                        "Minimize buttons are disabled.  Sorry...")
        endIf
        case menu_Cmd = userMenu + Find_ret : ret_find.pushbutton()
        case menu_Cmd = userMenu + FindLic  : licence_find.pushbutton()
        case menu_Cmd = userMenu + Ends_Prt : printBtn.pushButton()
        case menu_Cmd = userMenu + Exit  : exitbutton.pushButton()
        case menu_cmd = usermenu + file_print_setup: active.menuAction(menuFilePrinterSetup)
        case menu_Cmd = userMenu + Edit_Cut :
        active.menuAction(menuEditCut)
        setMenuChoiceAttributeByID(userMenu + editPast, menuEnabled)
        case menu_Cmd = userMenu + EditCopy :
        active.menuAction(menuEditCopy)
```

```
            setMenuChoiceAttributeByID(userMenu + editPast, menuEnabled)
            case menu_Cmd = userMenu + EditPast : active.menuAction(menuEditPaste)
            case menu_Cmd = userMenu + RecFirst : active.postAction(dataBegin)
            case menu_Cmd = userMenu + Rec_Prev : active.postAction(dataPriorRecord)
            case menu_Cmd = userMenu + Rec_Next : active.postAction(dataNextRecord)
            case menu_Cmd = userMenu + Rec_Last : active.postAction(dataEnd)
            case menu_Cmd = userMenu + Rec_Plus : Action(dataInsertRecord)
            case menu_Cmd = userMenu + Rec_Nuke : active.postAction(dataDeleteRecord)
            case menu_Cmd = userMenu + Rec_Canc : active.postAction(dataCancelRecord)
            case menu_Cmd = userMenu + Rec_Look : active.postAction(dataLookup)
            case menu_Cmd = userMenu + Endments  : Endorse.pushbutton()
            case menu_Cmd = userMenu + Licences   : Licencebutton.pushbutton()
            case menu_Cmd = userMenu + Findret   : Findretbttn.pushbutton()
            case menu_Cmd = userMenu + Findlic   : licence_find.pushbutton()
            case menu_Cmd = userMenu + Stats    : processButton.pushbutton()
            case menu_Cmd = userMenu + Reports   : reportprocessbutton.pushbutton()
            case menu_Cmd = userMenu + ends_new  : new_return.pushbutton()
            case menu_Cmd = userMenu + HelpHelp  : helpOnHelp()
            case menu_Cmd = userMenu + HelpList : helpShowContext(helpFile,1000)
            case menu_Cmd = userMenu + Prog_help  : helpShowindex(progHelp)
            case menu_Cmd = userMenu + get_spec : newspecies.pushbutton()
            case menu_Cmd = userMenu + Help_Use : helpShowContext(helpFile, detailHelp)
endswitch
endmethod
```

Object : mainpage

MethodName : clearPageObjects

Source :

```
method clearPageObjects(pageName String) Logical
var
            pageObjTc TCursor
            fieldObj UIObject
endVar
fidmslib.listPageObjects(formName.name, pageName)
pageObjTc.open("Objlist.db")
scan pageObjTc :
            fieldObj.attach(pageObjTc."Object Path")
            if fieldObj  <> "" then
                        fieldObj = ""
            endIf
endScan
fidmslib.clearPageValues(formName.Name, pageName)
RETURN TRUE
endmethod
```

Object : mainpage.processButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(statsFM) then
            statsFM.bringtoTop()
            statsFM.wait()  ; put statsFM into wait state
            statsFM.hide()
            maximize()
            mainpage.moveTo()
else
            fidmslib.setIsCalledTrue()
            if statsFM.Open("Stats.fsl", WinStyleMaximize) then
                        statsFM.Wait()  ; put statsFM into wait state
                        statsFM.hide()
                        maximize()
                        mainpage.moveTo()
            else
                        msgInfo("Status", "Sorry, the Statistics form is not available")
                        fidmslib.setIsCalledFalse()
            endif
endif
endmethod
```

Object : mainpage.licence_find

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        lic String
        ret_num UIObject
        userchoice String
        numfound smallint
        olesound OLE
endvar
ignorecaseinlocate(yes)
lic = ""
lic.view ("Enter a licence Number")
ret_num.attach(RETNUM)
if lic <> "" then
        if ret_num.locate("Licence_number",lic) then
                beep()
                numfound = 1
                message ("")
                userchoice = msgQuestion ("Choose", "Do you wish to see next " + String(lic))
                while userchoice = "Yes"
                        ret_num.locatenext("Licence_number",lic)
                        beep()
                        numfound = numfound + 1
                        message ("")
                        userchoice = msgQuestion ("Choose", "Do you wish to see next " + String(lic))
                endwhile
                return
                if not ret_num.locatenext ("Licence_number", lic) then
                        msgInfo("That's all folks for " + lic,"No further records")
                endif
                msgInfo("Found " + lic, strval(numFound) + " times.")
                return
        else
                msgInfo("No record found for " + lic, "Check spelling")
        endif
endif
endmethod
```

Object : mainpage.licence_find

MethodName : make_music

Source :

```
method make_music()
var
  quarterNote, octave, note LongInt
  power                Number
  olevar OLE
endVar
const
  noteA1  = 110
  noteA#1 = 116
  noteB1  = 123
  noteC1  = 130
  noteC#1 = 138
  noteD1  = 146
  noteD#1 = 155
  noteE1  = 164
  noteF1  = 174
  noteF#1 = 184
  noteG1  = 195
  noteG#1 = 207
  noteA2  = 220
  noteA#2 = 234
  noteB2  = 249
  noteC2  = 265
  noteC#2 = 282
  noteD2  = 300
endConst
sound(noteA1, 200)
sound(noteD1, 150)
```

```
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteA2, 150)
sound(noteF#1, 50)
sound(noteA2, 100)
sound(noteB2, 100)
sound(noteC#2, 150)
sound(noteD2, 50)
sound(noteA2, 100)
sound(noteF#1, 100)
sound(noteD1, 100)
sleep(1000)
endMethod
```

Object : mainpage.RETNUM.#Record27.Licence_number

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
var
        licTC tCursor
        persTC Tcursor
endvar
if not isblank(licence_number.value) then
        licTC.open ("licences.db")
        persTC.open ("persons.db")
        if licTC.locate("licence_no",licence_number.value) then
                personvalue.value = licTC.person_no
                persTC.locate("person_no",personvalue.value)
                lic_name.value = persTC.surname
        else
                message ("No record of correct surname for this licence")
                sleep (3000)
        endif
endif
endmethod
```

Object : mainpage.RETNUM.#Record27.Collecting_date

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
var
        LicTC tcursor
        licdlg Form
        dlgVal,pers_no String
        licloc query
endVar
if not isblank(self.value) then
        licTC.open ("licences.db")
        licTC.locate("licence_no",licence_number.value)
        if collecting_date.value > licTC.expiry_date or collecting_date.value <licTC.valid_from then
                eventinfo.seterrorcode(cannotdepart)
                message ("The return date is not valid for this licence ", "Check Licence Number")
                sleep (3000)
                pers_no = personvalue.value
                licloc = query

                :fidms:licences.db |Person_no      |Licence_no  |Valid_from     |Expiry_date  |
                        |Check ~pers_no   |Check      |Check         |Check       |

                endQuery
                if not executeqbe(licloc, "licloc.db") then
                        errorshow()
                endif
                if licDlg.open("licloc.fsl") then
                        dlgVal =  licDlg.wait()
                        if dlgVal= "OK" then
                                licence_number.value = licDlg.licence_no
                        endif
                endif
```

```
            endif
endif
endmethod
```

Object : mainpage.ret_find

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
            retNo LongInt
            ret UIObject
endVar
ret.attach(RETNUM)
retNo  = 0
retNo.view ("Enter a return number")
if retNo <> 0 then
            if not ret.locate("retnum",retNo) then
                        beep()
                        msginfo ("Search unsuccessful", "Couldn't locate Return No " + String(retNo))
            endif
endif
endmethod
```

Object : mainpage.Findretbttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
            ret_stattv Tableview
            ret_qry Query
            retstatTC TCursor
            retstatdlg Form
            theChoice String
            licence_no,dlgval String
            useritem String
endvar
licence_no = ""
licence_no.view ("Enter a licence number")
if licence_no <> " " then
            myitem.value = licence_no
endif
useritem = myitem.value
ret_qry = query

:Fidms:persons.db   |Person_no  |Surname   |Title|Forename|
                        |_efgh     |Check     |Check|Check   |

:Fidms:licences.db   |Person_no|licence_no        |
                        |_efgh    |Check_abcd,~useritem|

:Fidms:retnum.db    |Licence_number|Collecting_date|
                        |_abcd        |Check         |

endQuery
message ("The query is processing")
sleep (1000)
if not executeqbe(ret_qry,":priv:ret_stat.db") then
            errorshow()
endif
theChoice = msgYesNoCancel ("Reports","Do you wish to view the return status?")
switch
            case theChoice = "Yes" :
            if retstatDlg.open("retstat.fsl") then
                        dlgVal =  retstatDlg.wait()
            endif
            otherwise                   :return
endswitch
endmethod
```

Object : mainpage.new_return

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
action(datainsertrecord)
if isblank(RETNUM.retnum.value) then
        if retnoTC.lockrecord() then
                RETNUM.retnum.value = retnoTC.cmax("retnum") + 1
                retnoTC.retnum = RETNUM.retnum.value
                retnoTC.unlockrecord()
                RETNUM.licence_number.moveto()
        else
                msgstop ("Problem", "Couldn't lock the retno table")
        endif
endif
endmethod
```

Object : mainpage.reportprocessbutton

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
if isAssigned(reportsFm) then
        reportsFm.bringtoTop()
        reportsFm.wait()  ; put reportsFm into wait state
        reportsFm.hide()
        maximize()
        Floraret.moveTo()
else
        fidmslib.setIsCalledTrue()
        if reportsFm.Open("report.fsl", WinStyleMaximize) then
                reportsFm.Wait()  ; put reportsFm into wait state
                reportsFm.hide()
                maximize()
                Floraret.moveTo()
        else
                msgInfo("Status", "Sorry, the Reports form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.endorse

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
if isAssigned(endorseFM) then
        endorseFM.bringtoTop()
        endorseFm.wait()  ; put endorseFm into wait state
        endorseFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if endorseFm.Open("endorsfm.fsl", WinStyleMaximize) then
                endorseFm.Wait()  ; put endorseFm into wait state
                endorseFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Endorsement form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.licencebutton

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
if isAssigned(licencefm) then
```

```
        licencefm.bringtoTop()
        licencefm.wait()
        licencefm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if licencefm.Open("licence.fsl", WinStyleMaximize) then
                licencefm.Wait()  ; put licencefm into wait state
                licencefm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Licence form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.RETEX.#Record66.#Box157.TaxonID

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
var
        spMasTC TCursor
        hbttransTC TCursor
endvar
if not isblank(taxonid.value) then
        spMasTC.open (":fidms:hbttaxon.db")
        spMasTC.locate ("taxonid",taxonId)
        if spMasTC.curnt = "N" then
                message ("This taxon number is not current", "Program will update - please wait")
                sleep (3000)
                hbttransTC.open (":fidms:hbttrans.db")
                hbttransTC.locate("oldid",taxonid)
                taxonid.value = hbttranstc.newid
                spMasTC.locate ("taxonid",taxonID)
                genus.value = spmasTC.genus
                species.value = spmasTC.species
                InfraspRank.value = spmasTC.infraSp_Rank
                InfraspName.value = spmasTC.infrasp_Name
        else
                genus.value = spmasTC.genus
                species.value = spmasTC.species
                InfraspRank.value = spmasTC.infraSp_Rank
                InfraspName.value = spmasTC.infrasp_Name
        endif
endif
endmethod
```

Object : mainpage.RETEX.#Record66.#Box157.InfraspName

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
var
        SPMAS Tcursor
        HBTTRANSTC Tcursor
endvar
if isblank(taxonid.value) then
        SPMAS.open (":fidms:hbttaxon.db")
        if spmas.locate ("genus",genus.value,"species",species.value,"infrasp_rank",
          infrasprank.value,"infrasp_name",infraspname.value) then
                if SPMAS.curnt = "Y" then
                        taxonid.value=SPMAS.taxonId
                else
                        if SPMAS.curnt = "N" then
                                hbttransTC.open (":fidms:hbttrans.db")
                                if hbttransTC.locate ("oldId", spmas.taxonid) then
                                taxonId.value = hbttranstc.newid
                                        taxonid.moveto()
```

```
                                        msginfo ("Species Name Updated", "Revision of taxonomy")
                                    else
                                        msginfo ("Cannot find new taxon Id",
                                          "Check with Flora Industry Botanist")
                                    endif
                                else
                                    msginfo ("Species not valid",
                                      "Please Check in |Locate|Unknown Taxon Id")
                                endif
                            endif
                        endif
                endif
                endmethod
```

Object : mainpage.RETEX.#Record66.#Box76.Crown_private

MethodName : canDepart

Source :

```
method canDepart(var eventInfo MoveEvent)
var
        pritabTC tcursor
        newPass String
        pritab Table
endvar
pritab.attach (":pri:pri.db")
addpassword ("fidms")
pritabTC.open (":pri:pri.db")
if PritabTC.locate("TaxonID",taxonid.value) then
        if (pritabTC.Pr_code = "R" and crown_private.value <> "A") then
                eventinfo.seterrorcode(cannotdepart)
        taxonid.color=red
        Genus.color=red
        Species.color=red
        beep()
        msginfo ("This species is declared rare.",  "Contact AO Flora immediately")
        else
        if (pritabTC.pr_code = "1" or pritabTC.pr_code = "2" or
                pritabTC.pr_code = "3" or pritabTC.pr_code = "4" ) and crown_private = "C" then
                        eventinfo.seterrorcode(cannotdepart)
                        taxonid.color=red
                        Genus.color=red
                        Species.color=red
                msginfo("This is a priority species.",  "Contact AO Flora immediately")
                beep()
        else
                        if (pritabTC.pr_code = "R" and crown_private.value = "A") then
                                msginfo ("This species is declared rare.", "Contact AO Flora to check")
                else
                                if (pritabTC.pr_code = "5" ) and crown_private = "C" then
                                        eventinfo.seterrorcode(cannotdepart)
                                        taxonid.color=red
                                        Genus.color=red
                                        Species.color=red
                                msginfo("This species cannot be taken from Crown land.",
                                        "Contact AO Flora immediately")
                                beep()
                        else
                                        taxonid.color=Gray
                                        Genus.color=Gray
                                        Species.color=Gray
                                dodefault
                                        removepassword ("fidms")
                                endif
                        endif
                endif
        endif
endif
endmethod
```

Object : mainpage.RETEX.#Record66.Grid_Square

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
var
        geoTC tCursor
endvar
if not isblank(grid_square.value) then
        geoTC.open ("geograph.db")
        if geoTC.locate("Grid Square",grid_square.value) then
                dodefault
        else
                msginfo ("Stop", "This is incorrect grid number")
                eventinfo.setErrorCode(CanNotDepart)
        endif
endif
endmethod
```

Object : mainpage.RETEX.#Record66.Quantity

MethodName : depart

Source :

```
method depart(var eventInfo MoveEvent)
if isblank(taxonid.value) then
        msgStop ("Key violation", "You have not entered the taxon id")
else
        dodefault
endif
endmethod
```

Object : mainpage.newspecies

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        specdlg Form
        dlgVal String
        HbttransTC tcursor
        genus String
        species String
        sploc query
        notcurnt query
        splocTC tcursor
endvar
genus = ""
genus.view ("Enter Genus to search (partial is ok)")
species = ""
species.view ("Enter Species to search (partial is ok)")
sploc = query

:fidms:hbttaxon    |TaxonId|Genus          |Species          |InfraSp_Rank  |InfraSp_Name |Curnt|
                   |Check  |Check ..~genus..|Check ..~species..|Check         |Check        |Check|

endQuery

if not executeqbe(sploc,"sploc.db") then
        errorshow()
else
        message ("Executing query")
        sleep(5000)
endif
if specDlg.open("sploc.fsl") then
        dlgVal = specDlg.wait()
        if dlgVal= "OK" and SpecDlg.curnt = "Y" then
                taxonId.value = SpecDlg.taxonId
        else
                if specdlg.curnt = "N" and dlgval = "OK" then
                        getlic.value = SpecDlg.taxonId
                        specdlg.close()
                        Message ("Form is updating to current taxon id - Please wait")
                        sleep (5000)
                        hbttransTC.open (":fidms:hbttrans.db")
                        if hbttransTC.locate ("oldId", getlic.value) then
                        taxonId.value = hbttranstc.newid
```

```
                        else
                                message ("Cannot find new taxon Id")
                        sleep (5000)
                        endif
                endif
        endif
endif
endmethod
```

Object : mainpage.printBtn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        rettv Tableview
        retqry Query
        userin String
        floraret Report
        repInfo ReportPrintInfo
        theChoice String
endvar
userin = ""
userin.view ("Enter licence number to print")
if userin <> "0" then
        myitem.value = userin
endif
retqry = query

persons.db  |person_no   |Surname|Title|FORENAME  |ADDRESS1|ADDRESS2|ADDRESS3|Postcode|
            |_abcd       |Check  |Check|Check     |Check   |Check   |Check   |Check   |

licences.db  |person_no   |licence_no    |
             |_abcd       |_xyz ,~userin |

retnum.db   |retnum      |licence_number|Collecting_Date|
            |_defg       |Check _xyz    |Check          |

retex.db  |retnum       |Taxonid      |Genus     |Species   |Quantity      |Unit       |Part     |Grid square  |
          |_defg        |Check        |Check     |Check     |Check         |Check      |Check    |Check        |

endQuery

if not executeqbe(retqry,"retprn.db") then
        errorshow()
endif
theChoice = msgYesNoCancel ("Reports","Do you wish to print returns?")
switch
        case theChoice = "Yes" : floraret.open("floraret.rsl")
        repInfo.nCopies = 1
        floraret.print(repInfo)
        otherwise               :return
endswitch
endmethod
```

Object : mainpage.exitButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
choice = msgQuestion("Quit Flora Return Module", "Do you want to quit " +
                        "the module?")
if choice= "Yes" then
        okToExit = True
        close()
else
        eventInfo.setErrorCode(cannotDepart)
endif
endmethod
```

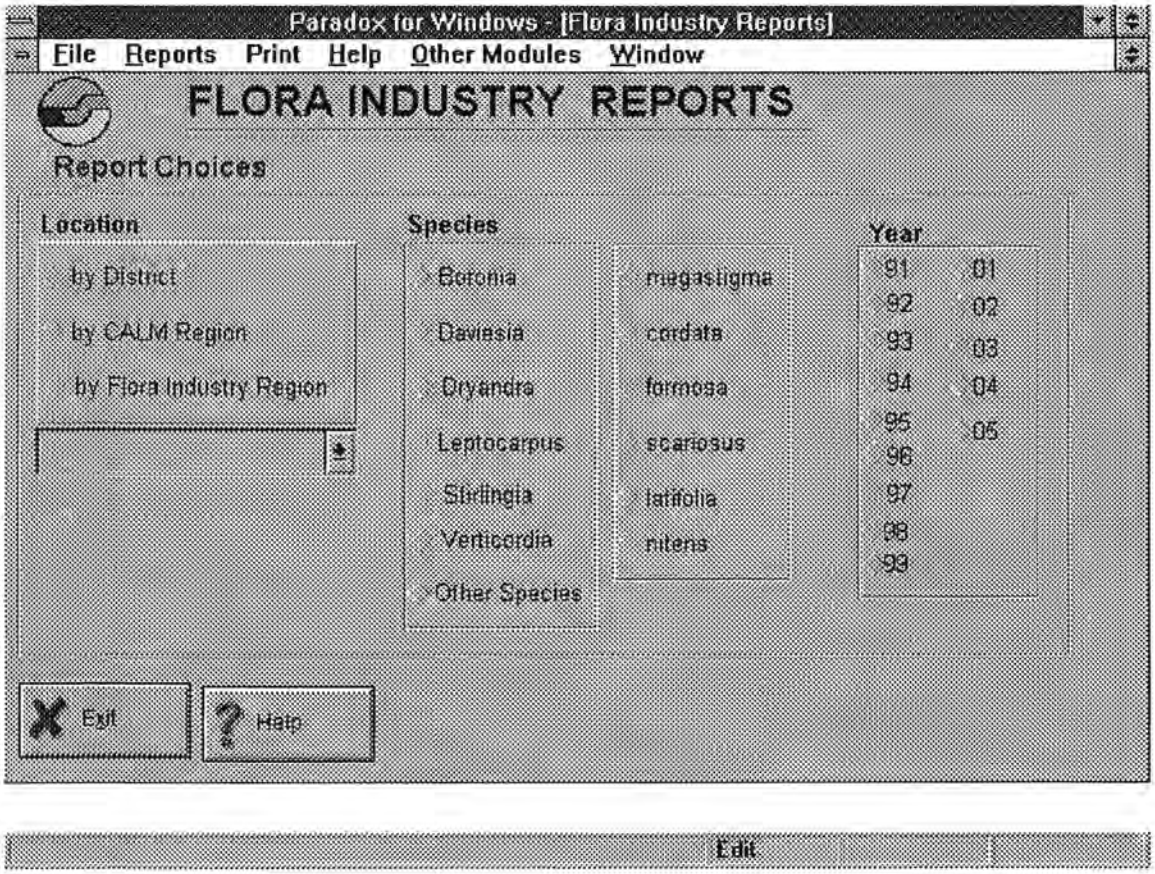
Object : mainpage.helpButton

Source :

```
method pushButton(var eventInfo Event)
        helpShowIndex(helpFile)
endmethod
```

Report Form

The Report Form is used to provide the user with summaries of licence data, endorsement data and return data for Regions, Districts and Flora Industry Regions and for pre-defined or user-defined species. It also performs the necessary queries to produce form letters to licensees depending on their return status. The form appears as shown below (but in colour).



REPORT.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Methods (REPORT.FSL)
Uses
Var
Const

close
keyPhysical
menuAction
mouseEnter
mouseExit
mouseRightUp

Objects
mainPage

Methods (mainPage)
Uses
Var
Const

action
arrive
block_ret_pr
close
cur_post
dis_pcode
end_dis
endspec
hist_pcode
open
lists_print
reg_pcode
renew
ret_dis_button
ret_reg_button
vehicle_dis
Dist_end
dist_ret_pr
end_reg
flora_ind
menuAction
print_lets
Reg_end
reg_ret_pr
ret_block_button
ret_florindret_button
species_block
veh_reg_pr

Objects (mainPage)
endorsebutton
exitButton
helpButton
floraretn
helpButton
licenceButton
processButton

Fields (mainPage)
pickWhich

Methods (pickWhich)
var
close
open
newValue

Object : Reports

MethodName : Uses

Source :
```
Uses ObjectPal
        showMe(const theTitle String, const listDataSource String)
        clearPageValues(formName String, pageName String)
        createObjMenu(var formName Form, pageName String, uio UIObject)
        fidmsObjMsg(formName String, pageName String, objID String)
        pageNameOf(ui UIObject) String
        returnIsCalled() Logical
        setIsCalledTrue()
        print_renew()
        print_lists()
        floraind()
        regret()
        distret()
        blockret()
        block_pick()
        LICENCE_BUTTON()
        floraret_button()
        stats_button()
        endorse_button()
        end_spec()
        dis_spec()
        vehicle_dis()
        vehicle_reg()
        end_reg()
        setIsCalledFalse()
        formCanClose()
        clearPageValues(formName String, pageName String)
        createObjMenu(var formName Form, pageName String, uio UIObject)
        fidmsObjMsg(formName String, pageName String, objID String)
        setPageValues(formName String, pageName String) Logical
        pageNameOf(ui UIObject) String
        returnIsCalled() Logical
endUses
```

Object : Reports

MethodName : Var

Source :
```
Var
        uioName, choice String
        uio UIObject
        formName, floradbFm, floraretFm, licenceFm, endorseFM, reportsFM, statsFM, aboutFm Form
        fidmsLib Library
        replib  Library
        closeVCRLg Logical
        okToExit Logical
        myBar Form
        wasCalled Logical
        newVal String
endVar
```

Object : Reports

MethodName : Const

Source :
```
Const
        helpFile  = ":fidms:floradb.hlp" ; path to Help file
        headerHelp = LongInt(5)
        detailHelp = LongInt(20007)
        proghelp  = ":fidms:fidprog.hlp" ; path to programmer help file
endConst
```

Object : Reports

MethodName : close

Source :
```
method close(var eventInfo Event)
if eventInfo.isPrefilter()  then
        doDefault
```

```
else
        if floradbFm.isAssigned() then
                floradbFm.close()
        endif
endif
endmethod
```

Object : Reports

MethodName : mouseEnter

Source :

```
method mouseEnter(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio)  ; what is the object?
        uioName = uio.Name
        if uio.name <> self.name AND (uio.class = "Field"
          OR uio.class = "Multirecord" OR uio.class = "Button") then
                fidmslib.FIDMSObjMsg(formName.name, fidmslib.pageNameOf(uio), uioName)
        endif
endif
endmethod
```

Object : Reports

MethodName : mouseExit

Source :

```
method mouseExit(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        message("")
else
endIf
endMethod
```

Object : Reports

MethodName : mouseRightUp

Source :

```
method mouseRightUp(var eventInfo MouseEvent)
var
        uiTarget        uiObject ; The object that was right-clicked
        thisForm        Form     ; Form variable for fidmsLIB.createObjMenu()
        Obj_Name        String   ; Used to make sure we search for the
                                 ; right object when we call createObjMenu.
endVar
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uiTarget)
        obj_Name = uiTarget.Name
        while obj_Name.subStr(1, 1) = "#" and uiTarget.containerName <> ""
                uiTarget.attach(uiTarget.containerName)
                obj_Name = uiTarget.Name
        endWhile
        thisForm.attach()
        fidmsLib.createObjMenu(thisForm, "mainpage", uiTarget)
        if uiTarget.class = "Field" then
                if uiTarget.tabStop then
                self.postAction(uiTarget.moveTo())
                endIf
        endIf
else
endIf
endMethod
```

Object : Reports

MethodName : keyPhysical

Source :

```
method keyPhysical(var eventInfo KeyEvent)
var
        theKey String
        thepage String
endvar
if eventInfo.isPreFilter() then
        theKey = eventInfo.vChar()  ; tell me what key was pressed
```

```
                switch
                        case theKey = "X" or theKey = "x" : disableDefault
                        mainpage.exitButton.pushButton()
                        otherwise: doDefault
                endswitch
        else
                switch
                        case theKey = "VK_F1" : disableDefault
                        helpShowContext(helpfile, detailhelp)
                        otherwise : doDefault
                endSwitch
        endif
endif
endmethod
```

Object : Reports

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
if not eventInfo.isPreFilter() then
        if eventInfo.id() = MenuControlMaximize oreventInfo.id() = MenuControlMinimize then
        disableDefault
        beep()
        message("Form cannot be resized.")
        else
                if eventInfo.reason() = menuNormal and eventInfo.id() = MenuCanClose
                  and not okToExit then
                        eventInfo.setErrorCode(CanNotDepart)
                        disableDefault
                        message("Use the Exit button to exit REPORTS.")
                endIf
        endIf
endIf
endmethod
```

Object : Reports

MethodName : FLORA_RET

Source :

```
method FLORA_RET()
        FIDMSLIB.FLORARET_BUTTON()
endmethod
```

Object : Reports

MethodName : stats_mod

Source :

```
method stats_mod()
        fidmslib.stats_button()
endmethod
```

Object : Reports

MethodName : endorsebut

Source :

```
method endorsebut()
        fidmslib.endorse_button()
endmethod
```

Object : mainpage

MethodName : Uses

Source :

```
Uses ObjectPal
        setCallFormName(const CallerTitle String, xPos LongInt, yPos LongInt)
        VCRisOpen()
        VCRisClosed()
        isVCROpen() Logical
        vcrName() String
        listPageObjects(formName String, pageName String) Logical
endUses
```

Object : mainpage

MethodName : Var

Source :
```
Var
        wtext,maintext array[12] String
endVar
```

Object : mainpage

MethodName : Const

Source :
```
Const

  ; Define the menu constants

        #IDH_MENUS_WINDOW_CURRENTWINDOW = "Current Open Windows"
        #IDH_MENUS_WINDOWS = "&Window"

        renew     =     101 ; Reports | Licence Renewal Letters
        cur_postcode =    102 ; Reports | Licences
        returnrep =     103 ; Reports | Returns
        FILE_PRINT_SETUP= 104
        end_region =      105 ; Endorsements | Region
        end_dist  =     106 ; Endorsements | District
        hist_postcode =   107 ; Reports | Licences
        reg_postcode  =   108 ; Reports | Licences
        endexit   =     109 ; Reports | Exit

        ret_reg   =     110 ; Returns|region
        ret_dis   =     111 ; Returns|district
        ret_block =      112 ; Returns|block
        ret_florind =    113 ; Returns|Flora industry region
        dis_postcode  =   114 ; Reports | Licences
        dist_ret     =  115;
        reg_ret   =     116;
        florind_ret =     117;
        block_ret  =    118;
        lets_pr   =     500;
        lists_pr  =     501 ;
        print_reg =      502 ;
        print_dis =     503 ;
        spec_reg  =      504 ;
        pick_reg  =     505 ;
        veh_reg   =     506 ;
        spec_block =      507 ;
        blockpick =     508 ;
        veh_dis   =     509 ;

        HelpHelp = 201 ; Help | Using Help command
        HelpList = 202 ; Help | Help Index command
        Help_Use = 203 ; Help | System Help command
        Prog_help = 204 ; Help | Programmer Help

        Florarets =301 ; Other Modules | Flora Returns command
        Licences = 302 ; Other Modules | Licences command
        Stats    = 302 ; Other Modules | Statistics command
        Endorse  = 303 ; Other Modules | Endorsements command

        Exit    = 401 ; Exit command

endConst
```

Object : mainpage

MethodName : open

Source :
```
method open(var eventInfo Event)
Message("Loading Reports,  One moment, please...")
delayScreenUpdates(Yes)
if not fidmsLib.open("fidmsLib.ldl", globalToDesktop) then
        msgStop("Failure", "fidmslib could not be opened")
endif
if not repLib.open("repLib.ldl", globalToDesktop) then
```

```
endif
formName.attach()
hideSpeedBar()
maximize()
okToExit = False
delayScreenUpdates(No)
if not isFile("report.fdl") then
        disableDefault
        beep()
        msgStop("Directory Error", "The Paradox for Windows " +
          "working directory must be set to the directory " +
          "containing this form")
        close()
else
        if not formatExist("Report") then
                disableDefault
                beep()
                msgStop("Can't Find Date Format", "You must have " +
                        "the Report date format defined " +
                        "before you can use this application.")
                close()
        else
                blankAsZero(No)
                fidmsLib.open("fidmsLIB.ldl", globalToDesktop)
                okToExit = False
                doDefault
                self.postAction(dataBeginEdit)
                delayScreenUpdates(No)
                wText[1]=#IDH_MENUS_WINDOW_CURRENTWINDOW
                mainText[5]=#IDH_MENUS_WINDOWS
        endIf
endif
endmethod
```

Object : mainpage

MethodName : close

Source :

```
method close(var eventInfo Event)
if fidmslib.isVCROpen() then
        myBar.close()
        fidmslib.VCRisClosed()
endif
endmethod
```

Object : mainpage

MethodName : arrive

Source :

```
method arrive(var eventInfo MoveEvent)
var
        mainMenu Menu      ; The main menu
        ReportMenu,        ; The Report popup menu
        endMenu,           ; The Endorsement popup menu
        retMenu,
        licMenu,
        renewsub,
        FILEMENU,
        endprnmenu,
        printMenu,         ; the Print popup menu
        printsub,
        retprn,
        ModuleMenu,        ; The Module menu
        windowmenu,
        HelpMenu           PopupMenu ; The Help menu
endVar
        fileMenu.addtext ("Printer &Setup", MenuEnabled, Usermenu + File_print_setup)
        filemenu.addtext ("E&xit", menuEnabled, userMenu + endexit)
        mainmenu.addpopup ("&File", filemenu)

        EndMenu.addtext ("&Region", menuEnabled, userMenu + end_Region)
        EndMenu.addtext ("&District", menuEnabled, userMenu + end_Dist)
```

```
        ReportMenu.addPopUp("&727 endorsements", EndMenu)
        retmenu.addtext ("&Region", menuEnabled, userMenu + ret_reg)
        retmenu.addtext ("&District", menuEnabled, userMenu + ret_dis)
        retmenu.addtext ("&Flora Industry Region", menuEnabled, userMenu + ret_florind)
        retmenu.addtext ("&Block or other area", menuEnabled, usermenu + ret_block)
        ReportMenu.addPopUp("&Returns", RetMenu)

        licMenu.addtext("Current licences by user-defined postcode",
          menuEnabled, userMenu + cur_postcode)
        licMenu.addtext("Historical licences by user-defined postcode",
          menuEnabled, userMenu + hist_postcode)
        licMenu.addtext("Current licences by Region", menuEnabled,
          userMenu + reg_postcode)
        licMenu.addtext("Current licences by District", menuEnabled,
          userMenu + dis_postcode)
        ReportMenu.addpopup("&Licences", licMenu)
        ReportMenu.addtext("Re&new Letters", menuEnabled, userMenu + Renew)
        mainmenu.addPopUp("&Reports", ReportMenu)

        renewsub.addtext("&Renew letters", menuEnabled, userMenu + lets_pr)
        renewsub.addtext("Renew &lists", menuEnabled, userMenu + lists_pr)
        printMenu.addpopup("&Renew", renewsub)
        endprnmenu.addstatictext("Region")
        endprnmenu.addseparator()
        endprnmenu.addtext ("&Species", menuEnabled, userMenu + spec_reg)
        endprnmenu.addtext ("&Pickers", MenuEnabled, userMenu + pick_reg)
        endprnmenu.addtext ("&Vehicle list", menuEnabled, userMenu + veh_reg)
        endprnmenu.addseparator()
        endprnmenu.addstatictext ("District")
        endprnmenu.addseparator()
        endprnmenu.addtext ("&Species", menuEnabled, userMenu + spec_block)
        endprnmenu.addtext ("&Pickers", MenuEnabled, userMenu + blockpick)
        endprnmenu.addtext ("&Vehicle list", menuEnabled, userMenu + veh_dis)
        printmenu.addpopup("Endorsements", endprnmenu)
        retprn.addtext ("&District", menuEnabled, userMenu + dist_ret)
        retprn.addtext ("&Region", menuEnabled, userMenu + reg_ret)
        retprn.addtext ("&Flora Industry Region", menuEnabled, userMenu + florind_ret)
        retprn.addtext ("&Block or other area", menuEnabled, userMenu + block_ret)
        printmenu.addpopup ("&Returns", retprn)
        MainMenu.addpopup("Print", printmenu)

        HelpMenu.addText("Using &Help\tCtrl+F1", menuEnabled, userMenu + helpHelp)
        HelpMenu.addText("Help &Index\tShift+F1", menuEnabled, userMenu + helpList)
        HelpMenu.addText("&Using this Form\tF1", menuEnabled, userMenu + help_Use)
        HelpMenu.addText("&Programmer Help", menuEnabled, userMenu + Prog_help)
        MainMenu.addPopUp("&Help", HelpMenu)

        ModuleMenu.addtext ("&Endorsements", menuEnabled, userMenu + Endorse)
        ModuleMenu.addtext ("&Statistics", menuEnabled, userMenu + Stats)
        ModuleMenu.addtext ("&Flora Returns", menuEnabled, userMenu + Florarets)
        ModuleMenu.addtext ("&Licences", menuEnabled, userMenu + Licences)
        MainMenu.addPopup ("&Other Modules", ModuleMenu)

        windowmenu.addStaticText(wText[1])
        mainMenu.addPopUp(mainText[5],windowmenu)

        mainMenu.show()

        setTitle("Flora Returns")
endmethod
```

Object : mainpage

MethodName : mouseExit

Source :

```
method mouseExit(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio) ; what is the object?
        if uio.class <> "Text" and uio.class <> "Bitmap" then
                message("")
```

```
endif
endmethod
```

Object : mainpage

MethodName : action

Source :

```
method action(var eventInfo ActionEvent)
        setTitle("Flora Industry Reports")
endmethod
```

Object : mainpage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
var
        menu_Cmd        smallInt ; The constant returned by a menu command
        editTest        Logical  ; Flag variable used to test the type of
                                 ; object that is active
        calledBy        Form     ; Holds the handle of the form that opened
                                 ; this form
        newRetnum       Number
        rettab          Table
endVar
menu_Cmd = eventInfo.id()
switch
        case menu_Cmd = menuInit :
        try
        editTest = active.Editing
        onFail
        editTest = False
        endTry
        if editTest then
        if active.selectedText <> "" then
        else
        endIf
        endIf
        case menu_Cmd = MenuControlClose :
        disableDefault
        exitbutton.pushButton()
        case menu_Cmd = MenuControlMaximize or menu_Cmd = MenuControlMinimize :
        If formcaller(calledBy) then
        disableDefault
        beep()
        msgInfo("Oops!", "Because this form was opened by the " +
                calledBy.getTitle() + " form, the Maximize and " +
                "Minimize buttons are disabled.  Sorry...")
        endIf
        active.menuAction(menuEditCut)
        case menu_cmd = userMenu + end_region : region_end.pushbutton()
        case menu_cmd = userMenu + end_dist   : dist_end.pushbutton()
        case menu_cmd = userMenu + renew      : renewbutton.pushbutton()
        case menu_cmd = userMenu + lets_pr    : print_lets()
        case menu_cmd = userMenu + lists_pr   : lists_print()
        case menu_cmd = userMenu + ret_reg    : ret_reg_button.pushbutton()
        case menu_cmd = userMenu + spec_reg   : endspec()
        case menu_cmd = userMenu + pick_reg   : endreg()
        case menu_cmd = userMenu + veh_reg    : veh_reg_pr()
        case menu_cmd = userMenu + block_ret  : block_ret_pr()
        case menu_cmd = userMenu + spec_block : species_block()
        case menu_cmd = userMenu + blockpick  : block_pick_pr()
        case menu_cmd = userMenu + veh_dis    : veh_dis_pr()
        case menu_cmd = userMenu + ret_dis    : ret_dis_button.pushbutton()
        case menu_cmd = userMenu + ret_florind: ret_florind_button.pushbutton()
        case menu_cmd = userMenu + ret_block  : ret_block_button.pushbutton()
        case menu_cmd = userMenu + cur_postcode  : cur_post.pushbutton()
        case menu_cmd = userMenu + hist_postcode : hist_post.pushbutton()
        case menu_cmd = userMenu + reg_postcode  :  reg_pcode.pushbutton()
        case menu_cmd = userMenu + dis_postcode  :  dis_pcode.pushbutton()
        case menu_cmd = userMenu + dist_ret   : dist_ret_pr()
        case menu_cmd = userMenu + reg_ret    : reg_ret_pr()
```

```
            case menu_cmd = userMenu + florind_ret: flora_ind()
            case menu_Cmd = userMenu + EndExit    : exitbutton.pushButton()
            case menu_Cmd = userMenu + exit       : exitbutton.pushButton()
            case menu_Cmd = userMenu + Florarets  : Flora_RET()
            case menu_Cmd = userMenu + Licences   : LICENCEBUT()
            case menu_Cmd = userMenu + Stats      : stats_mod()
            case menu_Cmd = userMenu + endorse    : endorsebut()
            case menu_Cmd = userMenu + HelpHelp   : helpOnHelp()
            case menu_Cmd = userMenu + HelpList   : helpShowContext(helpFile,1000)
            case menu_Cmd = userMenu + Help_Use   : helpShowContext(helpFile, headerHelp)
            case menu_Cmd = userMenu + Prog_help  : helpShowindex(progHelp)
            case menu_cmd = usermenu + file_print_setup: active.menuAction(menuFilePrinterSetup)
endswitch
endmethod
```

Object :	mainpage
MethodName :	print_lets
Source :	method print_lets() replib.print_renew() endmethod
Object :	mainpage
MethodName :	lists_print
Source :	method lists_print() replib.print_lists() endmethod
Object :	mainpage
MethodName :	flora_ind
Source :	method flora_ind() replib.floraind() endmethod
Object :	mainpage
MethodName :	reg_ret_pr
Source :	method reg_ret_pr() replib.regret() endmethod
Object :	mainpage
MethodName :	dist_ret_pr
Source :	method dist_ret_pr() replib.distret() endmethod
Object :	mainpage
MethodName :	block_ret_pr
Source :	method block_ret_pr() replib.blockret() endmethod
Object :	mainpage
MethodName :	block_pick_pr
Source :	method block_pick_pr() replib.block_pick() endmethod
Object :	mainpage
MethodName :	endspec
Source :	method endspec()

endmethod

Object : mainpage

MethodName : veh_dis_pr

Source :
```
method veh_dis_pr()
        replib.vehicle_dis()
endmethod
```

Object : mainpage

MethodName : species_block

Source :
```
method species_block()
        replib.dis_spec()
endmethod
```

Object : mainpage

MethodName : endreg

Source :
```
method endreg()
        replib.end_reg()
endmethod
```

Object : mainpage

MethodName : veh_reg_pr

Source :
```
method veh_reg_pr()
        replib.vehicle_reg()
endmethod
```

Object : mainpage

MethodName : LICENCEBUT

Source :
```
method LICENCEBUT()
        FIDMSLIB.LICENCE_BUTTON()
endmethod
```

Object : mainpage.reg_pcode

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
var
        currentlicqry Query
        licdlg Form
        licrep report
        repInfo ReportPrintInfo
        theChoice, dlgval String
        locchoice, usrchoice String
endvar
locchoice = ""
locchoice.view ("Enter Region")
currentlicqry=query

:fidms:persons.db  |Person_no  |Surname    |Initials    |Address1  |Address2  |Address3  |Postcode       |
                   |_abcd      |Check      |Check       |Check     |Check     |Check     |Check _reg     |

:fidms:licences.db  |Person_no  |Licence_no  |Valid_from  |Expiry_date  |
                    |_abcd      |Check       |Check <today |Check  >today |

:fidms:postcds.db   |Postcode   |Region     |
                    |_reg       |~locchoice |
endQuery

msginfo ("Please wait", "Query now processing")
if not executeqbe(currentlicqry,"cur_lic.db") then
        errorshow()
else
        message ("Executing query")
```

```
endif
thechoice = msgYesNoCancel ("Results","Do you wish to view licences")
switch
        case theChoice = "Yes" :    licrep.open("pcode.rdl")
        otherwise: return
endswitch
endmethod
```

Object : mainpage.dis_pcode

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        currentlicqry Query
        licdlg Form
        licrep report
        repInfo ReportPrintInfo
        theChoice, dlgval String
        locchoice, usrchoice String
endvar
locchoice = ""
locchoice.view ("Enter District")
currentlicqry=query

:fidms:persons.db |Person_no  |Surname    |Initials    |Address1 |Address2 |Address3  |Postcode      |
                  |_abcd      |Check      |Check       |Check    |Check    |Check     |Check _dis    |

:fidms:licences.db  |Person_no  |Licence_no |Valid_from  |Expiry_date  |
                    |_abcd      |Check      |Check <today |Check  >today |

:fidms:postcds.db   |Postcode   |District       |
                    |_dis       |Check ~locchoice  |
endQuery

msginfo ("Please wait", "Query now processing")
if not executeqbe(currentlicqry,"cur_lic.db") then
        errorshow()
else
        message ("Executing query")
        sleep(5000)
endif
thechoice = msgYesNoCancel ("Results","Do you wish to view licences")
switch
        case theChoice = "Yes" :    licrep.open("pcode.rdl")
        otherwise: return
endswitch
endmethod
```

Object : mainpage.hist_post

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        historylicqry, currentlicqry Query
        licdlg Form
        licrep report
        repInfo ReportPrintInfo
        theChoice, dlgval String
        locchoice, usrchoice String
endvar
locchoice = ""
locchoice.view ("Enter postcode(s)(separate by commas)")
historylicqry=query

:fidms:persons.db |Person_no  |Surname    |Initials    |Address1 |Address2 |Address3  |Postcode      |
                  |_abcd      |Check      |Check       |Check    |Check    |Check     |Check ~locchoice |

:fidms:licences.db  |Person_no  |Licence_no |Valid_from  |Expiry_date  |
                    |_abcd      |Check      |Check       |Check        |
```

```
endQuery
msginfo ("Please wait", "Query now processing")
if not executeqbe(historylicqry,"cur_lic.db") then
        errorshow()
else
        message ("Executing query")
        sleep(5000)
endif
thechoice = msgYesNoCancel ("Results","Do you wish to view licences")
switch
        case theChoice = "Yes" : licrep.open("pcode.rdl")
        otherwise: return
endswitch
endmethod
```

Object : mainpage.cur_post

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        historylicqry, currentlicqry Query
        licdlg Form
        licrep report
        repInfo ReportPrintInfo
        theChoice, dlgval String
        locchoice, usrchoice String
endvar
locchoice = ""
locchoice.view ("Enter postcode(s)(separate by commas)")

currentlicqry=query

:fidms:persons.db |Person_no  |Surname    |Initials   |Address1 |Address2 |Address3 |Postcode      |
                  |_abcd      |Check      |Check      |Check    |Check    |Check    |Check ~locchoice |

:fidms:licences.db  |Person_no  |Licence_no |Valid_from  |Expiry_date  |
                    |_abcd      |Check      |Check <today |Check  >today |

endQuery
msginfo ("Please wait", "Query now processing")
if not executeqbe(currentlicqry,"cur_lic.db") then
        errorshow()
else
        message ("Executing query")
        sleep(5000)
endif
thechoice = msgYesNoCancel ("Results","Do you wish to view licences")
switch
        case theChoice = "Yes" :      licrep.open("pcode.rdl")
        otherwise: return
endswitch
endmethod
```

Object : mainpage.Dist_end

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        endTab Table
        tmpTab Table
        endqry,end2qry Query
        endrep Report
        enddlg Form
        theChoice String
        genus,species String
        repInfo ReportPrintInfo
        dlgval String
        locchoice, usrchoice,yrchoice String
endvar
if querybox.listfield.value <> "" and querybox.pickwhich.districtbutton.value = true
```

```
and querybox.yearchoice <> ""  then
        usrChoice = msgQuestion("Report Choice", "Do you want to view report by species?")
        switch
                case usrChoice = "Yes" :
                if genusfield.value =  "Other Species" then
                        Genus = "Enter Genus here."
                        genus.view ("Which genus?")
                        Species = "Enter Species here."
                        species.view ("Which species?")
                        locchoice = querybox.listfield.value
                        yrchoice = querybox.yearchoice.value
                        endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District      |Endorse From|Endorse To  |
        |_end       |             |check ~locchoice |..~yrchoice |check      |

:fidms:end_loc.db |End_no |block_code |Location_Name |
        |_end  |_loc     |Check      |

:fidms:end_spec.db |block_code |Genus      |Species     |Quantity  |Unit      |Part      |
        |_loc     |Check ~genus |Check ~species|Calc sum  |Check     |Check     |

                        endQuery
                        msginfo ("Please wait", "Query now processing")
                        if not executeqbe(endqry,"endtmpsp.db") then
                                errorshow()
                        else
                                message ("Executing query")
                                sleep(5000)
                        endif
                else
                        if genusfield.value <> "" and genusfield.value <> "Other Species" then
                                genus = genusfield.value
                                species = speciesfield.value
                                locchoice = querybox.listfield.value
                                yrchoice = querybox.yearchoice.value
                                endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District      |Endorse From|Endorse To  |
        |_end       |             |check ~locchoice|..~yrchoice |check      |

:fidms:end_loc.db |End_no |block_code |Location_Name |
        |_end  |Check _loc |Check      |

:fidms:end_spec.db |block_code |Genus      |Species     |Quantity  |Unit      |Part      |
        |_loc     |Check ~genus |Check ~species|Calc sum  |Check     |Check     |

                                endQuery
                                msginfo ("Please wait", "Query now processing")
                                if not executeqbe(endqry,"endtmpsp.db") then
                                        errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                        else
                                if querybox.genusfield.value = "" then
                                        locchoice = querybox.listfield.value
                                        yrchoice = querybox.yearchoice.value
                                        endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District      |Endorse From|Endorse To  |
        |_end       |             |check ~locchoice |~yrchoice  |check      |

:fidms:end_loc.db |End_no |Block_code  |Location_Name |
        |_end  |Check _loc  |Check      |

:fidms:end_spec.db |Block_code  |Genus      |Species     |Quantity  |Unit      |Part      |
        |_loc     |Check      |Check      |Calc sum  |Check     |Check     |

                                        endQuery
```

```
                                        msginfo ("Please wait", "Query now processing")
                                        if not executeqbe(endqry,"endtmpsp.db")then
                                        errorshow()
                                        else
                                        message ("Executing query")
                                        sleep(5000)
                                        endif
                                endif
                        endif
                endif
                theChoice = msgYesNoCancel ("Reports","Do you wish to view by species?")
                switch
                        case theChoice = "Yes" :    endDlg.openasdialog("end_spec.fdl")
                        case theChoice = "No"  :    msginfo ("Print", "Use print menu to print")
                        case theChoice = "Cancel"   :return
                endswitch
                case usrChoice = "No":
                locchoice = querybox.listfield.value
                yrchoice = querybox.yearchoice.value
                endqry = query

:fidms:endorse.db |Licence_no|Endorsement_no|Issue_officer|Region       |District      |Endorse
From|Endorse To |
        |Check   |Check _end   |Check      |Check        |Check ~locchoice |..~yrchoice |check     |

:fidms:end_loc.db |End_no |Location_Name|Location Description|
        |_end  |Check      |Check          |

:fidms:vehicle.db |End_no |Vehicle_reg  |Colour|Vehicle Make|Vehicle Model|Vehicle Type|
        |_end  |Check      |Check |Check     |Check      |Check     |

                endQuery
                if not executeqbe(endqry,"endtmp.db")then
                        errorshow()
                endif
                endqry = query

:fidms:endorse.db |Licence_no|Endorsement_no|Issue_officer|Region |District      |Endorse
From|Endorse To  |
        |Check_lic |Check _end   |Check      |Check  |Check ~locchoice |..~yrchoice |check      |

:fidms:end_loc.db |End_no |Location_Name|Location Description|
        |_end  |Check      |Check          |

                endQuery
                msginfo ("Please wait", "Query now processing")
                if not executeqbe(endqry,":fidms:endrep.db") then
                        errorshow()
                else
                        message ("Executing query")
                        sleep(5000)
                endif
                theChoice = msgYesNoCancel ("Reports","Do you wish to view?")
                switch
                        case theChoice = "Yes" :    endDlg.openasdialog("end_rep.fdl")
                        case theChoice = "No"  :    msginfo ("Print", "Use print menu to print")
                        case theChoice = "Cancel"   :return
                endswitch
        endswitch
else
        if querybox.listfield.value <> "" and querybox.pickwhich.districtbutton.value = true
          and querybox.yearchoice = "" then
                usrChoice = msgQuestion("Report Choice", "Do you want to view report by species?")
                switch
                        case usrChoice = "Yes" :
                        if genusfield.value =  "Other Species" then
                                Genus = "Enter Genus here."
                                genus.view ("Which genus?")
                                Species = "Enter Species here."
                                species.view ("Which species?")
                                locchoice = querybox.listfield.value
```

```
                                endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District      |Endorse From|Endorse To  |
        |_end       |            |check ~locchoice |    <today|Check >today |

:fidms:end_loc.db |End_no |block_code |Location_Name |
        |_end  |_loc     |Check       |

:fidms:end_spec.db |block_code |Genus      |Species     |Quantity  |Unit      |Part      |
        |_loc     |Check ~genus |Check ~species|Calc sum  |Check     |Check     |

                                endQuery
                                msginfo ("Please wait", "Query now processing")
                                if not executeqbe(endqry,"endtmpsp.db") then
                                        errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                        else
                                if genusfield.value <> "" and genusfield.value <> "Other Species" then
                                        genus = genusfield.value
                                        species = speciesfield.value
                                        locchoice = querybox.listfield.value
                                        endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District      |Endorse From|Endorse To  |
        |_end       |            |check ~locchoice|    <today|Check >today |

:fidms:end_loc.db |End_no |block_code |Location_Name |
        |_end  |Check _loc |Check       |

:fidms:end_spec.db |block_code |Genus      |Species     |Quantity  |Unit      |Part      |
        |_loc     |Check ~genus |Check ~species|Calc sum  |Check     |Check     |

                                        endQuery
                                        msginfo ("Please wait", "Query now processing")
                                        if not executeqbe(endqry,"endtmpsp.db") then
                                                errorshow()
                                        else
                                                message ("Executing query")
                                                sleep(5000)
                                        endif
                                else
                                        if querybox.genusfield.value = "" then
                                                locchoice = querybox.listfield.value
                                                endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District      |Endorse From|Endorse To  |
        |_end       |            |check ~locchoice |    <today|Check >today |

:fidms:end_loc.db |End_no |Block_code  |Location_Name |
        |_end  |Check _loc  |Check       |

:fidms:end_spec.db |Block_code  |Genus      |Species     |Quantity  |Unit      |Part      |
        |_loc     |Check       |Check       |Calc sum  |Check     |Check     |

                                                endQuery
                                                msginfo ("Please wait", "Query now processing")
                                                if not executeqbe(endqry,"endtmpsp.db") then
                                                        errorshow()
                                                else
                                                        message ("Executing query")
                                                        sleep(5000)
                                                endif
                                                theChoice = msgYesNoCancel ("Reports",
                                                  "Do you wish to view by species?")
                                                switch
                                                        case theChoice = "Yes" :
                                                        endDlg.openasdialog("end_spec.fdl")
                                                        case theChoice = "No"  :
```

```
                                                    msginfo ("Print", "Use print menu to print")
                                                    case theChoice = "Cancel"     :return
                                                endswitch
                                            endif
                                        endif
                                    endif
                                    case usrChoice = "No":
                                    locchoice = querybox.listfield.value
                                    endqry = query

:fidms:endorse.db |Licence_no|Endorsement_no|Issue_officer|Region        |District       |Endorse
From|Endorse To   |
            |Check_lic |Check _end    |Check       |Check          |Check ~locchoice |     <today|Check >today
|

:fidms:end_loc.db |End_no |Location_Name|Location Description|
            |_end   |Check       |Check             |

:fidms:vehicle.db |End_no |Vehicle_reg  |Colour|Vehicle Make|Vehicle Model|Vehicle Type|
            |_end   |Check       |Check |Check      |Check       |Check      |

                                    endQuery
                                    if not executeqbe(endqry,"endtmp.db") then
                                        errorshow()
                                    endif
                                    endqry = query

:fidms:endorse.db |Licence_no|Endorsement_no|Issue_officer|Region |District       |Endorse
From|Endorse To   |
            |Check_lic |Check _end    |Check       |Check  |Check ~locchoice |     <today|Check >today |

:fidms:end_loc.db |End_no |Location_Name|Location Description|
            |_end   |Check       |Check             |

                                    endQuery
                                    msginfo ("Please wasit", "Query now processing")
                                    if not executeqbe(endqry,":fidms:endrep.db") then
                                        errorshow()
                                    else
                                        message ("Executing query")
                                        sleep(5000)
                                    endif
                                    theChoice = msgYesNoCancel ("Reports","Do you wish to view?")
                                    switch
                                        case theChoice = "Yes" :      endDlg.openasdialog("end_rep,fdk")
                                        case theChoice = "No"  :      msginfo ("Print", "Use print menu to print")
                                        case theChoice = "Cancel"     :return
                                    endswitch
                            endswitch
                    else
                            msgstop ("Stop", "You must choose district")
                    endif
            endif
            querybox.yearchoice.value = ""
            querybox.pickwhich.value = ""
            querybox.listfield.value = ""
            endmethod
```

Object :	mainpage.ret_florind_button

MethodName :	pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        locspec, specqry, otherspecqry Query
        retdlg Form
        retrep report
        genus,species String
        repInfo ReportPrintInfo
        theChoice, dlgval String
        locchoice, usrchoice String
```

```
if querybox.listfield.value <> "" and floraindbutton.value = true and querybox.yearchoice.value <> "" then
        usrChoice = msgQuestion("Report Choice", "Do you want only one species?")
        switch
                case usrChoice = "Yes" :
                if genusfield.value =  "Other Species"  then
                        Genus = "Enter Genus here."
                        genus.view ("Which genus?")
                        Species = "Enter Species here."
                        species.view ("Which species?")
                        locchoice = querybox.listfield.value
                        thechoice = querybox.yearchoice.value
                        locspec=query

 :fidms:retex.db    |Retnum |Genus        |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
               |_abcd  |Check ~genus   |Check ~species   |calc sum |Check|Check|Check _grid|Check
|Check          |

 :fidms:retnum.db    |Retnum |Collecting_date   |
               |_abcd  |Check ..~thechoice |

 :fidms:geograph.db  |Grid Square| Region          |District   |Picking Region   |
               |_grid     | Check           | Check    |Check ~locchoice |

                        endQuery
                        msginfo ("Please wait", "Query now processing")
                        if not executeqbe(locspec,"special.db") then
                                errorshow()
                        else
                                message ("Executing query")
                                sleep(5000)
                        endif
                else
                        if genusfield.value <> "Other Species"  and genusfield.value <> "" then
                                genus = Genusfield.value
                                species = speciesfield.value
                                locchoice = querybox.listfield.value
                                thechoice = querybox.yearchoice.value
                                specqry=query

 :fidms:retex.db    |Retnum |Genus        |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
               |_abcd  |Check ~genus   |Check ~species   |calc sum |Check|Check|Check _grid|Check
|Check          |

 :fidms:retnum.db    |Retnum |Collecting_date   |
               |_abcd  |Check ..~thechoice |

 :fidms:geograph.db  |Grid Square| Region    |District         |Picking Region   |
               |_grid     |         |Check            |Check ~locchoice |

                                endQuery
                                msginfo ("Please wait", "Query now processing")
                                if not executeqbe(specqry,"special.db") then
                                        errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                        else
                                if querybox.genusfield.value = "" then
                                        msgstop ("Error", "You have not chosen species")
                                endif
                        endif
                endif
                thechoice = msgYesNoCancel ("Results","Do you wish to view returns")
                switch
                        case theChoice = "Yes" :     retDlg.openasdialog("ret_rep.fdl")
                        case theChoice = "No"  :     msginfo ("Print", "Use print menu to print results")
                        case theChoice = "Cancel"    :return
                endswitch
```

```
                case usrchoice = "No" :
        locchoice = querybox.listfield.value
        thechoice = querybox.yearchoice.value
                specqry = query

:fidms:retex.db    |Retnum |Genus  |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
            |_abcd  |Check  |Check        |calc sum |Check|Check|Check _grid|Check    |Check     |

:fidms:retnum.db    |Retnum |Collecting_date   |
            |_abcd  |Check ..~thechoice |

:fidms:geograph.db  |Grid Square| Region    |District         |Picking Region  |
            |_grid    |         |Check            |Check ~locchoice |

                endquery
                msginfo ("Please wait", "Query now processing")
                if not executeqbe(specqry,"special.db") then
                        errorshow()
                else
                        message ("Executing query")
                        sleep(5000)
                endif
                thechoice = msgYesNoCancel ("Results","Do you wish to view returns")
                switch
                        case theChoice = "Yes" :      retDlg.openasdialog("ret_rep.fdl")
                        case theChoice = "No"  :      msginfo ("Print", "Use print menu to print results")
                        case theChoice = "Cancel"     :return
                endswitch
        endswitch
else
        if querybox.listfield.value = "" then
                msgstop ("Error", "You have not chosen flora industry region")
        else
                if querybox.listfield.value <> "" and floraindbutton.value = true
                  and querybox.yearchoice.value = False then
                        usrChoice = msgQuestion("Report Choice", "Do you want only one species?")
                        switch
                                case usrChoice = "Yes" :
                                if genusfield.value =  "Other Species" then
                                        Genus = "Enter Genus here."
                                        genus.view ("Which genus?")
                                        Species = "Enter Species here."
                                        species.view ("Which species?")
                                        locchoice = querybox.listfield.value
                                        locspec=query

  :fidms:retex.db    |Retnum |Genus        |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
            |_abcd  |Check ~genus   |Check ~species   |calc sum |Check|Check|Check _grid|Check
|Check         |

  :fidms:retnum.db    |Retnum |Collecting_date   |
            |_abcd  |Check          |

  :fidms:geograph.db  |Grid Square| Region          |District  |Picking Region  |
            |_grid    | Check          | Check    |Check ~locchoice |

  endQuery

                                        if not executeqbe(locspec,"special.db") then
                                                errorshow()
                                        else
                                                message ("Executing query")
                                                sleep(5000)
                                        endif
                                else
                                        if genusfield.value <> "Other Species"
                                          and genusfield.value <> "" then
                                                genus = Genusfield.value
                                                species = speciesfield.value
```

```
                                                    locchoice = querybox.listfield.value
                                                specqry=query

  :fidms:retex.db     |Retnum |Genus          |Species          |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                |_abcd  |Check ~genus    |Check ~species    |calc sum |Check|Check|Check _grid|Check
|Check          |

  :fidms:retnum.db    |Retnum |Collecting_date    |
                |_abcd  |Check            |

  :fidms:geograph.db   |Grid Square| Region     |District          |Picking Region   |
                |_grid     |          |Check             |Check ~locchoice |

                                                    endQuery
                                                    if not executeqbe(specqry,"special.db") then
                                                            errorshow()
                                                    else
                                                            message ("Executing query")
                                                            sleep(5000)
                                                    endif
                                                else
                                                            if genusfield.value = "" then
                                                            msgstop ("Error", "You have not chosen
species")
                                                    endif
                                                endif
                                        endif
                                        case usrchoice = "No" :
                                    locchoice = querybox.listfield.value
                                        qry = query

  :fidms:retex.db     |Retnum |Genus  |Species          |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                |_abcd  |Check  |Check           |calc sum |Check|Check|Check _grid|Check       |Check
|

  :fidms:retnum.db    |Retnum |Collecting_date    |
                |_abcd  |Check            |

  :fidms:geograph.db   |Grid Square| Region     |District          |Picking Region   |
                |_grid     |          |Check             |Check ~locchoice |

                                        query
                                        if not executeqbe(specqry,"special.db") then
                                                errorshow()
                                                message ("Executing query")
                                                sleep(5000)
                                        endif
                                        thechoice = msgYesNoCancel ("Results","Do you wish to view
returns")
                                        switch
                                                case theChoice = "Yes" : retDlg.openasdialog("ret_rep.fdl")
                                                case theChoice = "No"  :
                                                msginfo ("Print", "Use print menu to print results")
                                                case theChoice = "Cancel"     :return
                                        endswitch
                                endswitch
                        else
                                msgstop ("Stop", "You must choose flora industry region")
                        endif
                endif
endif
querybox.yearchoice.value = ""
querybox.pickwhich.value = ""
querybox.listfield.value = ""
endmethod
```

Object :	mainpage.helpButton
MethodName :	pushButton

Source :

```
            helpShowContext(helpFile, 20006)
endmethod
```

Object : mainpage.renewbutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
            rentv Tableview
            renqry Query
            norenqry Query
            userin String
            renew, norenew,renewlst,norenlst Report
            theChoice String
            printmenu popupmenu
endvar
userin = ""
userin.view ("Enter expiry date like >31/_/9_,<1/_/9_")
if userin <> " " then
            myitem.value = userin
endif
renqry = query

  :fidms:persons.db  |person_no|Surname|Title|Forename|Address1|Address2|Address3|Postcode|
                |_efgh   |check  |check|check   |check   |check   |check   |check   |

  :fidms:licences.db |licence_no   |person_no|Expiry_date  |
                |Check _abcd! |_efgh    |check ~userin |

  :fidms:retnum.db   |Licence_number |Collecting_date|
                |_abcd          |Count >6       |

endQuery
norenqry = query

  :fidms:persons.db  |person_no|Surname|Title|Forename|Address1|Address2|Address3|Postcode|
                |_efgh   |check  |check|check   |check   |check   |check   |check   |

  :fidms:licences.db |licence_no   |person_no|Expiry_date  |
                |Check _abcd! |_efgh    |check ~userin |

  :fidms:retnum.db   |Licence_number|Collecting_date|
                |_abcd          |Count <7       |

endQuery
message ("Executing renew letter query - wait")
sleep (5000)
if not executeqbe(renqry,"renew.db") then
            errorshow()
else
            if not executeqbe(norenqry,"norenew.db") then
                        errorshow()
            endif
endif
endmethod
```

Object : mainpage.ret_reg_button

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
            locspec, specqry, otherspecqry Query
            retdlg Form
            retrep report
            genus,species String
            repInfo ReportPrintInfo
            theChoice, dlgval String
            locchoice, usrchoice String
endvar
```

```
if querybox.listfield.value <> "" and regionbutton.value = true and querybox.yearchoice.value <> "" then
        usrChoice = msgQuestion("Report Choice", "Do you want only one species?")
        switch
                case usrChoice = "Yes" :
                genusfield.value =  "Other Species" then
                genus = "Enter Genus here."
                genus.view ("Which genus?")
                Species = "Enter Species here."
                species.view ("Which species?")
                locchoice = querybox.listfield.value
                thechoice = querybox.yearchoice.value
                locspec=query

 :fidms:retex.db     |Retnum |Genus         |Species         |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
              |_abcd  |Check ~genus    |Check ~species    |calc sum |Check|Check|Check _grid|Check
|Check          |

 :fidms:retnum.db    |Retnum |Collecting_date    |
              |_abcd  |Check ..~thechoice |

 :fidms:geograph.db   |Grid Square| Region          |District   |Picking Region   |
              |_grid     | Check ~locchoice   | Check    |Check          |

                endQuery
                msginfo ("Please wait", "Query now processing")
                if not executeqbe(locspec,"special.db") then
                        errorshow()
                else
                        message ("Executing query")
                        sleep(5000)
                endif
        else
                if genusfield.value <> "Other Species"  and genusfield.value <> "" then
                        genus = Genusfield.value
                        species = speciesfield.value
                        locchoice = querybox.listfield.value
                        thechoice = querybox.yearchoice.value
                        specqry=query

 :fidms:retex.db     |Retnum |Genus         |Species         |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
              |_abcd  |Check ~genus    |Check ~species    |calc sum |Check|Check|Check _grid|Check
|Check          |

 :fidms:retnum.db    |Retnum |Collecting_date    |
              |_abcd  |Check ..~thechoice |

 :fidms:geograph.db   |Grid Square| Region          |District          |Picking Region   |
              |_grid     | Check ~locchoice |Check             |Check          |

                        endQuery
                        msginfo ("Please wait", "Query now processing")
                        if not executeqbe(specqry,"special.db") then
                                errorshow()
                        else
                                message ("Executing query")
                                sleep(5000)
                        endif
                else
                        if querybox.genusfield.value = "" then
                                msgstop ("Error", "You have not chosen species")
                        endif
                endif
        endif
        thechoice = msgYesNoCancel ("Results","Do you wish to view returns")
        switch
                case theChoice = "Yes" :     retDlg.openasdialog("ret_rep.fdl")
                case theChoice = "No"  :     msginfo ("Print", "Use print menu to print results")
                case theChoice = "Cancel"    :return
        endswitch
```

```
        case usrchoice = "No" :
      locchoice = querybox.listfield.value
      thechoice = querybox.yearchoice.value
        specqry = query

 :fidms:retex.db    |Retnum |Genus  |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
            |_abcd  |Check  |Check          |calc sum |Check|Check|Check _grid|Check      |Check
|

 :fidms:retnum.db    |Retnum |Collecting_date   |
            |_abcd  |Check ..~thechoice |

 :fidms:geograph.db  |Grid Square| Region        |District         |Picking Region  |
            |_grid    |check  ~locchoice |Check            |Check         |

      endquery
      msginfo ("Please wait", "Query now processing")
      if not executeqbe(specqry,"special.db") then
            errorshow()
      else
            message ("Executing query")
            sleep(5000)
      endif
      thechoice = msgYesNoCancel ("Results","Do you wish to view returns")
      switch
            case theChoice = "Yes" :     retDlg.openasdialog("ret_rep.fdl")
            case theChoice = "No"  :     msginfo ("Print", "Use print menu to print results")
            case theChoice = "Cancel"    :return
      endswitch
endswitch
else
      if querybox.listfield.value = "" then
            msgstop ("Error", "You have not chosen region")
      else
            if querybox.listfield.value <> "" and regionbutton.value = true
              and querybox.yearchoice.value = False then
                  usrChoice = msgQuestion("Report Choice", "Do you want only one species?")
                  switch
                        case usrChoice = "Yes" :
                        if genusfield.value =  "Other Species" then
                              Genus = "Enter Genus here."
                              genus.view ("Which genus?")
                              Species = "Enter Species here."
                              species.view ("Which species?")
                              locchoice = querybox.listfield.value
                              locspec=query

 :fidms:retex.db    |Retnum |Genus        |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
            |_abcd  |Check ~genus   |Check ~species   |calc sum |Check|Check|Check _grid|Check
|Check         |

 :fidms:retnum.db    |Retnum |Collecting_date   |
            |_abcd  |Check          |

 :fidms:geograph.db  |Grid Square| Region          |District  |Picking Region  |
            |_grid    | Check ~locchoice  | Check    |Check         |

                              endQuery
                              if not executeqbe(locspec,"special.db") then
                                    errorshow()
                              else
                                    message ("Executing query")
                                    sleep(5000)
                              endif
                        else
                              if genusfield.value <> "Other Species"
                                and genusfield.value <> "" then
                                    genus = Genusfield.value
                                    species = speciesfield.value
```

```
                                                locchoice = querybox.listfield.value
                                        specqry=query

  :fidms:retex.db     |Retnum |Genus        |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                |_abcd  |Check ~genus   |Check ~species   |calc sum |Check|Check|Check _grid|Check
|Check          |

  :fidms:retnum.db    |Retnum |Collecting_date   |
                |_abcd  |Check          |

  :fidms:geograph.db  |Grid Square| Region    |District         |Picking Region   |
                |_grid     |Check ~locchoice |Check            |Check          |

                                                endQuery
                                                if not executeqbe(specqry,"special.db") then
                                                        errorshow()
                                                else
                                                        message ("Executing query")
                                                        sleep(5000)
                                                endif
                                        else
                                                if genusfield.value = "" then
                                                        msgstop ("Error", "You have not chosen
species")
                                                endif
                                        endif
                                endif
                                case usrchoice = "No" :
                        locchoice = querybox.listfield.value
                                specqry = query

  :fidms:retex.db     |Retnum |Genus  |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                |_abcd  |Check  |Check      |calc sum |Check|Check|Check _grid|Check      |Check      |

  :fidms:retnum.db    |Retnum |Collecting_date   |
                |_abcd  |Check          |

  :fidms:geograph.db  |Grid Square| Region        |District         |Picking Region   |
                |_grid     | Check ~locchoice |Check            |Check          |

                                endquery
                                if not executeqbe(specqry,"special.db") then
                                        errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                                thechoice = msgYesNoCancel ("Results","Do you wish to view
returns")
                                switch
                                        case theChoice = "Yes" :      retDlg.openasdialog("ret_rep.fdl")
                                        case theChoice = "No"  ;
                                        msginfo ("Print", "Use print menu to print results")
                                        case theChoice = "Cancel"     :return
                                endswitch
                        endswitch
                else
                        msgstop ("Stop", "You must choose region")

                endif
        endif
endif
querybox.yearchoice.value = ""
querybox.pickwhich.value = ""
querybox.listfield.value = ""
endmethod
```

Object : mainpage.ret_dis_button

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        locspec, specqry, otherspecqry Query
        retdlg Form
        retrep report
        genus,species String
        repInfo ReportPrintInfo
        theChoice, dlgval String
        locchoice, usrchoice String
endvar
if querybox.listfield.value <> "" and districtbutton.value = true and querybox.yearchoice.value <> "" then
        usrChoice = msgQuestion("Report Choice", "Do you want only one species?")
        switch
                case usrChoice = "Yes" :
                if genusfield.value =  "Other Species" then
                        Genus = "Enter Genus here."
                        genus.view ("Which genus?")
                        Species = "Enter Species here."
                        species.view ("Which species?")
                        locchoice = querybox.listfield.value
                        thechoice = querybox.yearchoice.value
                        locspec=query

  :fidms:retex.db     |Retnum |Genus         |Species         |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                |_abcd  |Check ~genus   |Check ~species   |calc sum |Check|Check|Check _grid|Check
|Check          |

  :fidms:retnum.db    |Retnum |Collecting_date   |
                |_abcd  |Check ..~thechoice |

  :fidms:geograph.db  |Grid Square| Region          |District        |Picking Region   |
                |_grid     | Check           |Check ~locchoice   |Check          |

                        endQuery
                        if not executeqbe(locspec,"special.db") then
                                errorshow()
                        else
                                message ("Executing query")
                                sleep(5000)
                        endif
                else
                        if genusfield.value <> "Other Species"  and genusfield.value <> "" then
                                genus = Genusfield.value
                                species = speciesfield.value
                                locchoice = querybox.listfield.value
                                thechoice = querybox.yearchoice.value
                                specqry=query

  :fidms:retex.db     |Retnum |Genus         |Species         |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                |_abcd  |Check ~genus   |Check ~species   |calc sum |Check|Check|Check _grid|Check
|Check          |

  :fidms:retnum.db    |Retnum |Collecting_date   |
                |_abcd  |Check ..~thechoice |

  :fidms:geograph.db  |Grid Square| Region    |District          |Picking Region   |
                |_grid     |          |Check ~locchoice    |Check          |

                                endQuery
                                if not executeqbe(specqry,"special.db") then
                                        errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                        else
                                if genusfield.value = "" then
```

```
                                        msgstop ("Error", "You have not chosen species")
                                endif
                        endif
                endif
                case usrchoice = "No" :
        locchoice = querybox.listfield.value
        thechoice = querybox.yearchoice.value
                specqry = query

 :fidms:retex.db    |Retnum |Genus  |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
            |_abcd  |Check  |Check          |calc sum |Check|Check|Check _grid|Check   |Check    |

 :fidms:retnum.db   |Retnum |Collecting_date   |
            |_abcd  |Check ..~thechoice |

 :fidms:geograph.db  |Grid Square| Region    |District         |Picking Region  |
            |_grid    |Check      |Check ~locchoice    |Check          |

                endquery
                if not executeqbe(specqry,"special.db") then
                        errorshow()
                else
                        message ("Executing query")
                        sleep(5000)
                endif
                thechoice = msgYesNoCancel ("Results","Do you wish to view returns")
                switch
                        case theChoice = "Yes" :      retDlg.openasdialog("ret_rep.fdl")
                        case theChoice = "No"  :
                        msginfo ("Print", "Use print menu to print results")
                        case theChoice = "Cancel"     :return
                endswitch
        endswitch
else
        if querybox.listfield.value <> "" and querybox.pickwhich.districtbutton.value = true
          and querybox.yearchoice.value = "" then
                usrChoice = msgQuestion("Report Choice", "Do you want only one species?")
                switch
                        case usrChoice = "Yes" :
                        if genusfield.value =  "Other species" then
                                Genus = "Enter Genus here."
                                genus.view ("Which genus?")
                                Species = "Enter Species here."
                                species.view ("Which species?")
                                locchoice = querybox.listfield.value
                                locspec=query

 :fidms:retex.db    |Retnum |Genus          |Species          |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
            |_abcd  |Check ~genus    |Check ~species    |calc sum |Check|Check|Check _grid|Check
|Check          |

 :fidms:retnum.db   |Retnum |Collecting_date   |
            |_abcd  |Check          |

 :fidms:geograph.db  |Grid Square| Region          |District          |Picking Region  |
            |_grid    | Check           | Check ~locchoice    |Check          |

                                endQuery
                                 if not executeqbe(locspec,"special.db") then
                                        errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                        else
                                if genusfield.value <> "Other Species"
                                  and genusfield.value <> "" then
                                        genus = Genusfield.value
                                        species = speciesfield.value
```

```
                                        locchoice = querybox.listfield.value
                                        specqry=query

  :fidms:retex.db     |Retnum |Genus         |Species         |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                |_abcd  |Check ~genus   |Check ~species   |calc sum |Check|Check|Check _grid|Check
|Check          |

  :fidms:retnum.db    |Retnum |Collecting_date   |
                |_abcd  |Check           |

  :fidms:geograph.db  |Grid Square| Region    |District          |Picking Region   |
                |_grid    |         |Check  ~locchoice   |Check          |

                                        endQuery
                                        if not executeqbe(specqry,"special.db") then
                                                errorshow()
                                        else
                                                message ("Executing query")
                                                sleep(5000)
                                        endif
                                else
                                        if genusfield.value = "" then
                                                msgstop ("Error", "You have not chosen species")
                                        endif
                                endif
                        endif
                        case usrchoice = "No" :
                locchoice = querybox.listfield.value
                        specqry = query

  :fidms:retex.db     |Retnum |Genus  |Species         |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                |_abcd  |Check  |Check          |calc sum |Check|Check|Check _grid|Check      |Check
|

  :fidms:retnum.db    |Retnum |Collecting_date   |
                |_abcd  |Check           |

  :fidms:geograph.db  |Grid Square| Region    |District          |Picking Region   |
                |_grid    |         |Check ~locchoice    |Check          |

                        endquery
                        if not executeqbe(specqry,"special.db") then
                                errorshow()
                        else
                                message ("Executing query")
                                sleep(5000)
                        endif
                        thechoice = msgYesNoCancel ("Results","Do you wish to view returns")
                        switch
                                case theChoice = "Yes" :     retDlg.openasdialog("ret_rep.fdl")
                                case theChoice = "No"  :
                                msginfo ("Print", "Use print menu to print results")
                                case theChoice = "Cancel"     :return
                        endswitch
                endswitch
        else
                if querybox.listfield.value = false then
                        msgstop ("Stop", "You must choose district")
                endif
        endif
endif
querybox.yearchoice.value = ""
querybox.pickwhich.value = ""
querybox.listfield.value = ""
endmethod
```

Object :	mainpage.ret_block_button
MethodName :	pushButton

Source :

```
var
        locspec, specqry, otherspecqry Query
        retdlg Form
        retrep report
        genus,species string
        fbk String
        repInfo ReportPrintInfo
        theChoice, dlgval String
        locchoice, usrchoice String
endvar
if querybox.yearchoice.value <> "" then
        usrChoice = msgQuestion("Report Choice", "Do you want only one species?")
        switch
                case usrChoice = "Yes" :
                if genusfield.value = "Other Species" and querybox.listfield.value <> "" then
                        fbk = "Enter Genus here"
                        fbk.view ("Which genus?")
                        Genus = "Enter Genus here"
                        genus.view ("Which genus?")
                        Species = "Enter Species here"
                        species.view ("Which species?")
                        thechoice = querybox.yearchoice.value
                        if fbk <> "" then
                        locspec=query

    :fidms:retex.db     |Retnum |Genus          |Species          |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name   |
                |_abcd  |Check ~genus    |Check ~species    |calc sum |Check|Check|Check _grid|Check
|Check ..~fbk..      |

    :fidms:retnum.db    |Retnum |Collecting_date    |
                |_abcd  |Check ..~thechoice |

    :fidms:geograph.db  |Grid Square| Region          |District   |Picking Region   |
                |_grid     | Check           |Check       |Check          |

                        endQuery
                        if not executeqbe(locspec,"special.db") then
                                errorshow()
                        else
                                message ("Executing query")
                                sleep(5000)
                        endif
                else
                        msgstop ("Error", "You have not chosen block")
                endif
        else
                if genusfield.value <> "Other Species"  and genusfield.value <> "" then
                        fbk = ""
                        fbk.view ("Which block to view")
                        genus = Genusfield.value
                        species = speciesfield.value
                        thechoice = querybox.yearchoice.value
                if fbk <> "" then
                                specqry=query

      :fidms:retex.db     |Retnum |Genus          |Species          |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                |_abcd  |Check ~genus    |Check ~species    |calc sum |Check|Check|Check _grid|Check
|Check ..~fbk..    |

      :fidms:retnum.db    |Retnum |Collecting_date    |
                |_abcd  |Check ..~thechoice |

      :fidms:geograph.db  |Grid Square| Region     |District      |Picking Region   |
                |_grid     | check     |Check        |Check          |

                                endQuery
                                if not executeqbe(specqry,"special.db") then
                                errorshow()
                                else
```

```
                                    message ("Executing query")
                                    sleep(5000)
                                    endif
                            else
                                    msgstop ("Error", "You have not chosen block")
                                    endif
                    else
                                    if genusfield.value = "" then
                                    msgstop ("Error", "You have not chosen species")
                                    endif
                    endif
            endif
            case usrchoice = "No" :
            fbk = ""
            fbk.view ("Which block to view")
            thechoice = querybox.yearchoice.value
            if fbk <> "" then
                    specqry = query

    :fidms:retex.db     |Retnum |Genus  |Species          |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                    |_abcd  |Check  |Check            |calc sum |Check|Check|Check _grid|Check      |Check
..~fbk..   |

    :fidms:retnum.db    |Retnum |Collecting_date   |
                    |_abcd  |Check ..~thechoice |

    :fidms:geograph.db  |Grid Square|Region    |District          |Picking Region  |
                    |_grid     |check     |Check             |Check          |

                    endquery
                    if not executeqbe(specqry,"special.db") then
                            errorshow()
                    else
                            message ("Executing query")
                            sleep(5000)
                    endif
            else
                    msgstop ("Error", "You have not chosen block")
            endif
endswitch
else
            if querybox.yearchoice.value = "" then
                    usrChoice = msgQuestion("Report Choice", "Do you want only one species?")
                    switch
                            case usrChoice = "Yes" :
                            if genusfield.value =  "Other Species" and querybox.listfield.value <> "" then
                            fbk = "Enter Genus here."
                            fbk.view ("Which block to view")
                            Genus = "Enter Genus here."
                            genus.view ("Which genus?")
                            Species = "Enter Species here."
                            species.view ("Which species?")
                            if fbk <> "" then
                                    locspec=query

            :fidms:retex.db     |Retnum |Genus          |Species          |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name  |
                            |_abcd  |Check ~genus    |Check ~species    |calc sum |Check|Check|Check
_grid|Check      |Check ..~fbk..     |

            :fidms:retnum.db    |Retnum |Collecting_date   |
                            |_abcd  |Check            |

            :fidms:geograph.db  |Grid Square| Region          |District          |Picking Region  |
                            |_grid     | Check           |Check             |Check          |

                                    endQuery
                                    if not executeqbe(locspec,"special.db") then
                                                    errorshow()
                                    else
```

```
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                        else
                                        msgstop ("Error", "You have not chosen block")
                        endif
                else
                        if genusfield.value <> "Other Species"  and genusfield.value <> "" then
                                fbk = ""
                                fbk.view ("Which block to view")
                                genus = Genusfield.value
                                species = speciesfield.value
                                if fbk <> "" then
                                        specqry=query

        :fidms:retex.db     |Retnum |Genus        |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                        |_abcd  |Check ~genus   |Check ~species   |calc sum |Check|Check|Check
_grid|Check      |Check ..~fbk..   |

        :fidms:retnum.db    |Retnum |Collecting_date   |
                        |_abcd  |Check           |

        :fidms:geograph.db  |Grid Square| Region    |District         |Picking Region   |
                        |_grid     |Check      |Check            |Check          |

                                        endQuery
                                        if not executeqbe(specqry,"special.db") then
                                                errorshow()
                                        else
                                                message ("Executing query")
                                                sleep(5000)
                                        endif
                                else
                                        msgstop ("Error", "You have not chosen block")
                                endif
                        else
                                if genusfield.value = "" then
                                        msgstop ("Error", "You have not chosen species")
                                endif
                        endif
                        endif
                        case usrchoice = "No" :
                        fbk = ""
                        fbk.view ("Which block to view")
                        if fbk <> "" then
                                specqry = query

        :fidms:retex.db     |Retnum |Genus  |Species        |Quantity |Unit |Part |Grid
Square|Crown_private|Owner/Company_Name|
                        |_abcd  |Check  |Check          |calc sum |Check|Check|Check _grid|Check
|Check ..~fbk..   |

        :fidms:retnum.db    |Retnum |Collecting_date   |
                        |_abcd  |Check           |

        :fidms:geograph.db  |Grid Square| Region    |District         |Picking Region   |
                        |_grid     |         |Check            |Check          |

                                endquery
                                if not executeqbe(specqry,"special.db") then
                                        errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                        else
                                msgstop ("Error", "You have not chosen block")
                        endif
                endswitch
        endif
```

```
endif
thechoice = msgYesNoCancel ("Results","Do you wish to view returns")
switch
        case theChoice = "Yes" :
        retDlg.openasdialog("ret_rep.fdl")
        case theChoice = "No"  :
        msginfo ("Print", "Use print menu to print results")
        case theChoice = "Cancel"      :return
endswitch
querybox.yearchoice.value = ""
querybox.pickwhich.value = ""
querybox.listfield.value = ""
endmethod
```

Object : mainpage.Region_end

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        endTab Table
        tmpTab Table
        endqry,end2qry Query
        endrep Report
        enddlg Form
        Tblcreate, srcTbl  Table
        theChoice String
        genus,species String
        repInfo ReportPrintInfo
        dlgval String
        destTbl string
        locchoice, usrchoice,yrchoice  String
endvar
tblcreate.attach(":fidms:endtmpsp.db")
tblcreate.empty()
tblcreate.attach(":fidms:endtmp.db")
tblcreate.empty()
if querybox.listfield.value <> "" and querybox.pickwhich.regionbutton.value = true and
  querybox.yearchoice.value <> "" then
        usrChoice = msgQuestion("Report Choice", "Do you want to view report by species?")
        switch
                case usrChoice = "Yes" :
                if genusfield.value =  "Other Species" then
                        Genus = "Enter Genus here."
                        genus.view ("Which genus?")
                        Species = "Enter Species here."
                        species.view ("Which species?")
                        locchoice = querybox.listfield.value
                        yrchoice = querybox.yearchoice.value
                        endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District       |Endorse From  |Endorse To  |
            |_end        |Check ~locChoice |            |..~yrchoice  |check       |

:fidms:end_loc.db |End_no |Block_code  |Location_Name |
            |_end   |Check _loc  |Check       |

:fidms:end_spec.db |Block_code  |End_no|Genus       |Species      |Quantity  |Unit       |Part       |
            |_loc       |_end  |Check ~genus |Check ~species|Calc sum  |Check      |Check      |

                        endQuery
                        msginfo ("Please wait", "Query now processing")
                        if not executeqbe(endqry,":fidms:endtmpsp.db") then
                                errorshow()
                        else
                                message ("Executing query")
                                sleep(5000)
                        endif
                else
                        if genusfield.value <> "" and genusfield.value <> "Other Species" then
                                genus = genusfield.value
```

```
                                locchoice = querybox.listfield.value
                                yrchoice = querybox.yearchoice.value
                                endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District       |Endorse From|Endorse To  |
          |_end        |Check ~locChoice |            |..~yrchoice |check       |

:fidms:end_loc.db |End_no |Block_code  |Location_Name |
          |_end   |Check _loc  |Check       |

:fidms:end_spec.db |Block_code  |End_no|Genus       |Species      |Quantity  |Unit       |Part       |
          |_loc       |_end  |Check ~genus |Check ~species|Calc sum  |Check      |Check      |

                                endQuery
                                if not executeqbe(endqry, ":fidms:endtmpsp.db")then
                                errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                        endif
                        else
                                if querybox.genusfield.value = ""  then
                                locchoice = querybox.listfield.value
                                yrchoice = querybox.yearchoice.value
                                endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District       |Endorse From|Endorse To |
          |_end        |Check ~locChoice |            |..~yrchoice |check      |

:fidms:end_loc.db |End_no |Block_code   |Location_Name |
          |_end   |Check _loc   |Check       |

:fidms:end_spec.db |Block_code |End_no   |Genus       |Species      |Quantity  |Unit       |Part       |
          |_loc      |_end     |Check       |Check        |Calc sum  |Check      |Check      |

                                endQuery
                                msginfo ("Please wait", "Query now processing")
                                if not executeqbe(endqry,":fidms:endtmpsp.db") then
                                        errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                                endif
                        endif
                endif
                theChoice = msgYesNoCancel ("Reports","Do you wish to view by species?")
                switch
                        case theChoice = "Yes" :      enddlg.openasdialog("end_spec.fdl")
                        case theChoice = "No"  :      msginfo ("Print", "Use print menu to print")
                        case theChoice = "Cancel"     :return
                endswitch
                case usrChoice = "No":
                locchoice = querybox.listfield.value
                yrchoice = querybox.yearchoice.value
                endqry = query

:fidms:endorse.db |Licence_no|Endorsement_no|Issue_officer|Region        |District       |Endorse
From|Endorse To  |
          |Check    |Check _end    |Check       |Check ~locChoice |Check          |..~yrchoice |check       |

:fidms:end_loc.db |End_no |Location_Name|Location Description|
          |_end   |Check       |Check            |

                endQuery
                if not executeqbe(endqry,":fidms:endrep.db") then
                        errorshow()
                endif
                endqry = query

:fidms:endorse.db |Licence_no|Endorsement_no|Issue_officer|Region        |District       |Endorse
```

```
From|Endorse To  |
        |Check_lic |Check _end   |Check      |Check ~locChoice |Check          |..~yrchoice |check      |

:fidms:end_loc.db |End_no |Location_Name|Location Description|
        |_end  |Check      |Check           |

:fidms:vehicle.db |End_no |Vehicle_reg  |Colour|Vehicle Make|Vehicle Model|Vehicle Type|
        |_end  |Check      |Check |Check     |Check      |Check     |

                endQuery
                msginfo ("Please wait", "Query now processing")
                if not executeqbe(endqry,":fidms:endtmp.db") then
                        errorshow()
                else
                        message ("Executing query")
                        sleep(5000)
                endif
                theChoice = msgYesNoCancel ("Reports","Do you wish to view?")
                switch
                        case theChoice = "Yes" :     endDlg.openasdialog("end_rep.fdl")
                        case theChoice = "No"  :     msginfo ("Print", "Use print menu to print")
                        case theChoice = "Cancel"    :return
                endswitch
        endswitch
else
        if querybox.listfield.value <> "" and querybox.pickwhich.regionbutton.value = true
          and querybox.yearchoice.value = "" then
                usrChoice = msgQuestion("Report Choice", "Do you want to view report by species?")
                switch
                        case usrChoice = "Yes" :
                        if genusfield.value =  "Other Species" then
                                Genus = "Enter Genus here."
                                genus.view ("Which genus?")
                                Species = "Enter Species here."
                                species.view ("Which species?")
                                locchoice = querybox.listfield.value
                                endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District      |Endorse From|Endorse To  |
        |_end       |Check ~locChoice |           |    <today|Check >today |

:fidms:end_loc.db |End_no |Block_code  |Location_Name |
        |_end  |Check _loc  |Check      |

:fidms:end_spec.db |Block_code  |End_no|Genus      |Species     |Quantity  |Unit      |Part      |
        |_loc      |_end  |Check ~genus |Check ~species|Calc sum  |Check     |Check     |

                                endQuery
                                msginfo ("Please wait", "Query now processing")
                                if not executeqbe(endqry,":fidms:endtmpsp.db") then
                                        errorshow()
                                else
                                        message ("Executing query")
                                        sleep(5000)
                                endif
                        else
                                if genusfield.value <> "" and genusfield.value <> "Other Species" then
                                        genus = genusfield.value
                                        species = speciesfield.value
                                        locchoice = querybox.listfield.value
                                        endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District      |Endorse From|Endorse To  |
        |_end       |Check ~locChoice |           |    <today|Check >today |

:fidms:end_loc.db |End_no |Block_code  |Location_Name |
        |_end  |Check _loc  |Check      |

:fidms:end_spec.db |Block_code  |End_no|Genus      |Species     |Quantity  |Unit      |Part      |
        |_loc      |_end  |Check ~genus |Check ~species|Calc sum  |Check     |Check     |
```

```
                                        endQuery
                                        msginfo ("Please wait", "Query now processing")
                                        if not executeqbe(endqry, ":fidms:endtmpsp.db") then
                                                errorshow()
                                        else
                                                message ("Executing query")
                                                sleep(5000)
                                        endif
                                        theChoice = msgYesNoCancel ("Reports",
                                          "Do you wish to view by species?")
                                        switch
                                                case theChoice = "Yes" :
                                                enddlg.openasdialog("end_spec.fdl")
                                                case theChoice = "No"  :
                                                msginfo ("Print", "Use print menu to print")
                                                case theChoice = "Cancel"     :return
                                        endswitch
                                else
                                        if querybox.genusfield.value = ""  then
                                                locchoice = querybox.listfield.value
                                                endqry = query

:fidms:endorse.db |Endorsement_no|Region        |District       |Endorse From|Endorse To  |
            |_end       |Check ~locChoice |           |    <today|Check >today |

:fidms:end_loc.db |End_no |Block_code  |Location_Name |
            |_end  |Check _loc  |Check       |

:fidms:end_spec.db |Block_code |End_no  |Genus       |Species      |Quantity  |Unit       |Part       |
            |_loc     |_end    |Check      |Check       |Calc sum  |Check      |Check      |

                                                endQuery
                                                msginfo ("Please wait", "Query now processing")
                                                if not executeqbe(endqry,":fidms:endtmpsp.db") then
                                                errorshow()
                                                else
                                                message ("Executing query")
                                                sleep(5000)
                                                endif
                                                theChoice = msgYesNoCancel ("Reports",
                                                  "Do you wish to view by species?")
                                                switch
                                                        case theChoice = "Yes" :
                                                        endDlg.openasdialog("end_spec.fdl")
                                                        case theChoice = "No"  :
                                                        msginfo ("Print", "Use print menu to print")
                                                        case theChoice = "Cancel"     :return
                                                endswitch
                                        endif
                                endif
                        endif
                        case usrChoice = "No":
                        locchoice = querybox.listfield.value
                        endqry = query

:fidms:endorse.db |Licence_no|Endorsement_no|Issue_officer|Region        |District       |Endorse
From|Endorse To  |
            |Check_lic |Check _end   |Check      |Check ~locChoice |Check         |    <today|Check
>today |

:fidms:end_loc.db |End_no |Location_Name|Location Description|
            |_end  |Check      |Check           |

                        endQuery
                        msginfo ("Please wait", "Query now processing")
                        if not executeqbe(endqry,":fidms:endrep.db")then
                                errorshow()
                        else
                                message ("Executing query")
                                sleep(5000)
                        endif
```

```
                        endqry = query

:fidms:endorse.db |Licence_no|Endorsement_no|Issue_officer|Region        |District      |Endorse
From|Endorse To  |
            |Check_lic |Check _end   |Check      |Check ~locChoice |Check         |    <today|Check
>today |

:fidms:end_loc.db |End_no |Location_Name|Location Description|
            |_end  |Check      |Check           |

:fidms:vehicle.db |End_no |Vehicle_reg  |Colour|Vehicle Make|Vehicle Model|Vehicle Type|
            |_end  |Check      |Check |Check      |Check      |Check      |

                        endQuery
                        msginfo ("Please wait", "Query now processing")
                        if not executeqbe(endqry,":fidms:endtmp.db")then
                                errorshow()
                        else
                                message ("Executing query")
                                sleep(5000)
                        endif
                        theChoice = msgYesNoCancel ("Reports","Do you wish to view?")
                        switch
                                case theChoice = "Yes" :     enddlg.openasdialog("end_rep.fdl")
                                case theChoice = "No"  :     msginfo ("Print", "Use print menu to print")
                                case theChoice = "Cancel"    :return
                        endswitch
                endswitch
        else
                msgstop ("Sorry", "You have not chosen Region")
        endif
endif
querybox.yearchoice.value = ""
querybox.pickwhich.value = ""
querybox.listfield.value = ""
endmethod
```

Object : mainpage.exitButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
choice = msgQuestion("Quit Flora Industry Reports Module", "Do you want to quit " + "the module?")
if choice= "Yes" then
        okToExit = True
        close()
else
        eventInfo.setErrorCode(cannotDepart)
endif
endmethod
```

Object : mainpage.querybox.pickwhich

MethodName : Var

Source :

```
Var
        locTC Tcursor
        i   Smallint
endVar
```

Object : mainpage.querybox.pickwhich

MethodName : open

Source :

```
method open(var eventInfo Event)
        locTC.open ("location.db")
        self.value = "District"
endmethod
```

Object : mainpage.querybox.pickwhich

MethodName : close

```
method close(var eventInfo Event)
```

```
        loctc.close()
endmethod
```

Object : mainpage.querybox.pickwhich

MethodName : newValue

Source :

```
method newValue(var eventInfo Event)
if eventinfo.reason() = editvalue or eventinfo.reason() = startupvalue then
        listfield.thelist.list.count  = 0
        switch
                case self = "by District" :locTC.movetorecord(1)
                case self = "by CALM Region"   :locTC.movetoRecord(23)
                case self = "by Flora Industry Region" :locTC.movetoRecord(32)
        endswitch
        for i from 1 to 23
                listfield.thelist.list.selection = i
                listfield.thelist.list.value = locTC.location
                locTC.nextrecord()
        endfor
endif
endmethod
```

Statistics Form (STATS.FSL)

Produces statistics as shown in Flora Industry 1993 report. This program is a heavy user of processing time. The form appears as shown below (but in colour).

STATS.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Methods (STATS.FSL)	**Objects**
Uses	mainPage
Var	
Const	
close	
keyPhysical	
menuAction	
mouseEnter	
mouseExit	
mouseRightUp	

Methods (mainPage)		**Objects (mainPage)**
Uses		endorsebutton
Var		exitButton
Const		floraret
		helpButton
arrive	open	licencebutton
menuAction	ANCArpt	reportprocessButton
flregseed	flregstems	
geoseed	geostems	
gridseed	gridstems	
seed_cpa	seed_gen	
spec_no	stems_cpa	
stems_gen	topseed	
topstems		

Full listings of the code follows.

Object : Statistics

MethodName : Uses

Source :
```
Uses ObjectPal
        clearPageValues(formName String, pageName String)
        createObjMenu(var formName Form, pageName String, uio UIObject)
        FIDMSObjMsg(formName String, pageName String, objID String)
        pageNameOf(ui UIObject) String
        setIsCalledTrue()
        setIsCalledFalse()
        formCanClose()
endUses
```

Object : Statistics

MethodName : Var

Source :
```
Var
        uioName, choice String
        uio UIObject
        formName, floradbFm, floraretFm, licenceFm, reportsFM, endorseFM, aboutFm Form
        fidmslib, geolib, statslib Library
        closeVCRLg Logical
        okToExit Logical
        stObjName,
        stHelpFileName String
        liContextId    LongInt
endVar
```

Object : Statistics

MethodName : Const

Source :
```
Const
        helpFile   = ":fidms:floradb.hlp" ; path to Help file
        headerHelp = LongInt(20006)
        detailHelp = LongInt(6)
        proghelp   = ":fidms:fidprog.hlp" ; path to programmer help file
endConst
```

Object : Statistics

MethodName : close

Source :
```
method close(var eventInfo Event)
if eventInfo.isPrefilter() then
        doDefault
else
        if floradbFm.isAssigned() then
                floradbFm.close()
        else
                if floraretFm.isAssigned() then
                        floraretFm.close()
                endif
                if endorseFm.isAssigned() then
                        endorseFm.close()
                endif
                if licenceFm.isAssigned() then
                        licenceFm.close()
                endif
                if reportsFm.isAssigned() then
                        reportsFM.close()
                endif
                FIDMSlib.formCanClose()
        endif
        showSpeedBar()
        removeMenu()
```

```
        helpquit(":fidms:fidprog.hlp")
endif
endmethod
```

Object : Statistics

MethodName : mouseExit

Source :

```
method mouseExit(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio) ; what is the object?
        if uio.class <> "Text" and uio.class <> "Bitmap" then
                message("")
        endif
endif
endmethod
```

Object : Statistics

MethodName : mouseRightUp

Source :

```
method mouseRightUp(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio) ; what is the object?
        if uio.name <> "calmlogoBox" then
                while uio.class = "Text" or uio.class = "Bitmap"
                        uio.attach(uio.ContainerName)
        endWhile
        endif
        fidmslib.createObjMenu(formName, fidmslib.pageNameOf(uio), uio)
endif
endmethod
```

Object : Statistics

MethodName : keyPhysical

Source :

```
method keyPhysical(var eventInfo KeyEvent)
var
        theKey String
        thePage String
endvar
if eventInfo.isPreFilter() then
        theKey = eventInfo.vChar()  ; tell me what key was pressed
        if eventInfo.isAltKeyDown() then
                switch
                        case theKey = "E" or theKey = "e" : disableDefault
                        mainpage.endorsebttn.pushbutton()
                        case theKey = "F" or theKey = "f" : disableDefault
                        mainPage.floraretButton.pushButton()
                        case theKey = "R" or theKey = "r" : disableDefault
                        mainPage.reportprocessbutton.pushbutton()
                        case theKey = "X" or theKey = "x" : disableDefault
                        mainPage.exitButton.pushButton()
                        otherwise: doDefault
                endswitch
        else
                switch
                        case theKey = "VK_F1" : disableDefault
                        helpShowContext(helpfile, detailhelp)
                otherwise : doDefault
                endSwitch
        endif
endif
endmethod
```

Object : Statistics

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
  If not eventInfo.isPreFilter() then
        If eventInfo.reason() = menuNormal and
```

```
                    eventInfo.setErrorCode(CanNotDepart)
            disableDefault
            message("Use the Exit button to exit Statistics.")
            endIf
endIf
endmethod
```

Object : MainPage

MethodName : Uses

Source :

```
Uses ObjectPal
        setCallFormName(const CallerTitle String, xPos LongInt, yPos LongInt)
        VCRisOpen()
        cpa_stems()
        cpa_seed()
        Grid_stems()
        Grid_seed()
        No_of_species()
        No_stems()
        No_seed()
        g_stems()
        g_seed()
        geo_stems()
        geo_seed()
        flreg_stems()
        flreg_seed()
        ANCA()
        VCRisClosed()
        isVCROpen() Logical
        vcrName() String
        listPageObjects(formName String, pageName String) Logical
endUses
```

Object : MainPage

MethodName : Var

Source :

```
Var
        wtext,maintext array[12] String
endVar
```

Object : MainPage

MethodName : Const

Source :

```
Const
  #IDH_MENUS_WINDOW_CURRENTWINDOW = "Current Open Windows"
  #IDH_MENUS_WINDOWS = "&Window"

  HelpHelp = 101 ; Help | Using Help command
  HelpList = 102 ; Help | Help Index command
  Help_Use = 103 ; Help | System Help command
  Prog_help= 104 ; Help | Programmer Help

  Florarets= 201 ;Modules | Flora Returns command
  Licences = 202 ; Modules | Licences command
  Stats    = 202 ; Modules | Statistics command
  Reports  = 203 ; Modules | Reports command
  Endorse  = 204 ; Modules | Endorsements command

  file_print = 300
  Exit       = 301 ; Exit command
  file_print_setup = 302

  no_spec  = 400 ;
  ANCA_item= 401 ;
  grid     = 402 ;
  stem_spec= 403 ;
  seedno   = 404 ;
  gen_stem = 405 ;
  gen_seed = 406 ;
```

```
cpaseed  = 408 ;
stems_geo= 409 ;
seed_geo = 410 ;
stems_flreg = 411 ;
seed_flreg  = 412 ;
grseed   = 413 ;
```

Object : MainPage

MethodName : open

Source :

```
method open(var eventInfo Event)
delayScreenUpdates(Yes)
if not fidmslib.open("fidmslib.ldl", globalToDesktop) then
        msgStop("Failure", "fidmslib could not be opened")
endif
if not geolib.open("geolib.ldl", globalToDesktop) then
        msgStop("Failure", "geolib could not be opened")
endif
if not statslib.open("statslib.ldl", globalToDesktop) then
        msgStop("Failure", "statslib could not be opened")
endif
wText[1]=#IDH_MENUS_WINDOW_CURRENTWINDOW
mainText[5]=#IDH_MENUS_WINDOWS
formName.attach()
hideSpeedBar()
maximize()
okToExit = False
delayScreenUpdates(No)
Message("Loading Statistics.  One moment, please...")
delayScreenUpdates(Yes)
if not isFile("stats.fdl") then
        disableDefault
        beep()
        msgStop("Directory Error", "The Paradox for Windows " +
        "working directory must be set to the directory " +
        "containing this form, for example: " +
        "T:\\fidms\\")
        close()
        blankAsZero(No)
        fidmslib.open("fidmslib.ldl", globalToDesktop)
        okToExit = False
        doDefault
        self.postAction(dataBeginEdit)
        delayScreenUpdates(No)
endIf
endmethod
```

Object : MainPage

MethodName : arrive

Source :

```
method arrive(var eventInfo MoveEvent)
var
        mainMenu        Menu
        querymenu,
        SpeciesMenu,
        stemsmenu,
        seedmenu,
        geoMenu,
        filemenu,
        ModuleMenu,
        HelpMenu,
        windowMenu      popupmenu
endVar

  fileMenu.addtext ("&Print", Menuenabled, Usermenu + file_print)
  fileMenu.addtext ("Printer &Setup", MenuEnabled, Usermenu + File_print_setup)
  filemenu.addtext ("E&xit", menuEnabled, userMenu + exit)
  mainmenu.addpopup ("&File", filemenu)
```

```
ModuleMenu.addtext ("&Flora Returns", menuEnabled, userMenu + Florarets)
ModuleMenu.addtext ("&Reports", menuEnabled, userMenu + Reports)
ModuleMenu.addtext ("&Licences", menuEnabled, userMenu + Licences)
ModuleMenu.addtext ("&Endorsements", menuEnabled, UserMenu + Endorse)
MainMenu.addpopup("&Other Modules",ModuleMenu)

speciesmenu.addtext ("&Number of species harvested", MenuEnabled, UserMenu + no_spec)
speciesmenu.addtext ("&ANCA Report - total harvest", MenuEnabled, UserMenu + ANCA_item)
querymenu.addpopup ("General &Reports", speciesmenu)

stemsmenu.addtext ("Harvest of stems by &species", MenuEnabled, UserMenu + stem_spec)
stemsmenu.addtext ("Harvest of stems by &genus", MenuEnabled, UserMenu + gen_stem)
stemsmenu.addtext ("Total harvest of stems by &land status", MenuEnabled, UserMenu + cpastems)
stemsmenu.addtext ("&Distibution of harvesting of species for stems", MenuEnabled, UserMenu + grid)
queryMenu.addpopup ("&Flowering Stems", Stemsmenu)

seedMenu.addtext ("Harvest of seed by &species", MenuEnabled, UserMenu + seedno)
seedmenu.addtext ("Harvest of seed by &genus", MenuEnabled, UserMenu + gen_seed)
seedmenu.addtext ("Total harvest of seed by &land status", MenuEnabled, UserMenu + cpaseed)
seedmenu.addtext ("&Distibution of harvesting of species for seed", MenuEnabled, UserMenu +
grseed)
querymenu.addpopup ("&Seed", Seedmenu)

geomenu.addtext ("Harvesting of stems by &grid square", MenuEnabled, UserMenu + stems_geo)
geomenu.addtext ("Harvesting of seed by g&rid square", MenuEnabled, UserMenu + seed_geo)
geomenu.addtext ("Harvesting of stems by &flora region", MenuEnabled, UserMenu + stems_flreg)
geomenu.addtext ("Harvesting of seed by f&lora region", MenuEnabled, UserMenu + seed_flreg)
querymenu.addpopup ("&Geographic", geomenu)
mainmenu.addpopup ("&Queries", querymenu)

HelpMenu.addText("Using &Help\tCtrl+F1", menuEnabled, userMenu + helpHelp)
HelpMenu.addText("Help &Index\tShift+F1", menuEnabled, userMenu + helpList)
HelpMenu.addText("&Using this Form\tF1", menuEnabled, userMenu + help_Use)
HelpMenu.addText("&Programmer Help", menuEnabled, userMenu + Prog_help)
MainMenu.addPopUp("&Help", HelpMenu)

windowmenu.addStaticText(wText[1])
mainMenu.addPopUp(mainText[5],windowmenu)

mainMenu.show()

setTitle("Statistics")

endmethod
```

Object : MainPage

MethodName : mouseEnter

Source :

```
method mouseEnter(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio)  ; what is the object?
        uioName = uio.Name
        if uio.name <> self.name AND (uio.class= "Field" OR uio.class = "Multirecord"
          OR uio.class = "Button") then
                fidmslib.FIDMSObjMsg(formName.name,
                fidmslib.pageNameOf(uio), uioName)
        endif
endif
endmethod
```

Object : MainPage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
var
        menu_Cmd        smallInt ; The constant returned by a menu command
        editTest        Logical  ; Flag variable used to test the type of
                                 ; object that is active
  calledBy              Form     ; Holds the handle of the form that opened
                                 ; this form
```

```
        d_search        DDE
        whichrec        longInt
endVar
menu_Cmd = eventInfo.id()
switch
        case menu_Cmd = MenuControlClose :
        disableDefault
        exitbutton.pushButton()
        case menu_Cmd = MenuControlMaximize or menu_Cmd = MenuControlMinimize :
        if formcaller(calledBy) then
        disableDefault
        beep()
        msgInfo("Oops!", "Because this form was opened by the " +
                        calledBy.getTitle() + " form, the Maximize and " +
                        "Minimize buttons are disabled.  Sorry...")
        endIf
        case menu_Cmd = userMenu + No_spec  :     movetorecno(1)
        Spec_no()
        case menu_Cmd = userMenu + seedno   : picture.action(datanextrecord)
        topseed()
        case menu_Cmd = userMenu + ANCA_item  :  whichRec = LongInt(PICTURE.nRecords()/4)
        PICTURE.moveToRecord(WhichRec)
        ANCArpt()
        case menu_Cmd = userMenu + grid   :
        whichRec = LongInt(PICTURE.nRecords()/3)
        PICTURE.moveToRecord(WhichRec)
        gridstems()
        case menu_Cmd = userMenu + grseed   :
        whichRec = LongInt(PICTURE.nRecords()/2)
        PICTURE.moveToRecord(WhichRec)
        gridseed()
        case menu_Cmd = userMenu + stem_spec :
        whichRec = LongInt(PICTURE.nRecords()/5)
        PICTURE.moveToRecord(WhichRec)
        topstems()
        case menu_Cmd = userMenu + gen_stem  :
        whichRec = LongInt(PICTURE.nRecords()/6)
        PICTURE.moveToRecord(WhichRec)
        stems_gen()
        case menu_Cmd = userMenu + gen_seed  :
        whichRec = LongInt(PICTURE.nRecords()/7)
        PICTURE.moveToRecord(WhichRec)
        seed_gen()
        case menu_Cmd = userMenu + cpastems  :
        whichRec = LongInt(PICTURE.nRecords()/8)
        PICTURE.moveToRecord(WhichRec)
        stems_cpa()
        case menu_Cmd = userMenu + cpaseed   :
        whichRec = LongInt(PICTURE.nRecords()/9)
        PICTURE.moveToRecord(WhichRec)
        seed_cpa()
        case menu_Cmd = userMenu + stems_geo : seed_cpa()
        whichRec = LongInt(PICTURE.nRecords()/10)
        PICTURE.moveToRecord(WhichRec)
        seed_cpa()
        case menu_Cmd = userMenu + seed_geo  : geoseed()
        case menu_Cmd = userMenu + stems_flreg: flregstems()
        case menu_Cmd = userMenu + seed_flreg: flregseed()
        case menu_Cmd = userMenu + Florarets : Floraretbutton.pushbutton()
        case menu_Cmd = userMenu + Licences    : Licencebutton.pushbutton()
        case menu_Cmd = userMenu + Reports   : reportprocessbutton.pushbutton()
        case menu_Cmd = userMenu + Endorse   : endorsebttn.pushbutton()
        case menu_Cmd = userMenu + Exit     : exitButton.pushbutton()
        case menu_Cmd = userMenu + HelpHelp  : helpOnHelp()
        case menu_Cmd = userMenu + Prog_help  : helpShowindex(progHelp)
        case menu_Cmd = userMenu + HelpList : helpshowContext(helpfile,6000)
        case menu_Cmd = userMenu + Help_Use : helpShowContext(helpFile, detailHelp)
        case menu_cmd = usermenu + file_print: active.menuAction(menuFilePrint)
        case menu_cmd = usermenu + file_print_setup: active.menuAction(menuFilePrinterSetup)
endswitch
endmethod
```

Object :	MainPage
MethodName :	spec_no
Source :	method spec_no() statslib.no_of_species() endmethod

Object :	MainPage
MethodName :	ANCArpt
Source :	method ANCArpt() geolib.anca() endmethod

Object :	MainPage
MethodName :	gridstems
Source :	method gridstems() statslib.grid_stems() endmethod

Object :	MainPage
MethodName :	topstems
Source :	method topstems() statslib.no_stems() endmethod

Object :	MainPage
MethodName :	topseed
Source :	method topseed() statslib.no_seed() endmethod

Object :	MainPage
MethodName :	stems_gen
Source :	method stems_gen() geolib.g_stems() endmethod

Object :	MainPage
MethodName :	seed_gen
Source :	method seed_gen() geolib.g_seed() endmethod

Object :	MainPage
MethodName :	stems_cpa
Source :	method stems_cpa() geolib.cpa_stems() endmethod

Object :	MainPage
MethodName :	seed_cpa
Source :	method seed_cpa() geolib.cpa_seed() endmethod

Object :	MainPage
MethodName :	geostems
	method geostems()

```
        geolib.geo_stems()
endmethod
```

Object : MainPage

MethodName : geoseed

Source :
```
method geoseed()
        geolib.geo_seed()
endmethod
```

Object : MainPage

MethodName : flregstems

Source :
```
method flregstems()
        geolib.flreg_stems()
endmethod
```

Object : MainPage

MethodName : flregseed

Source :
```
method flregseed()
        geolib.flreg_seed()
endmethod
```

Object : MainPage

MethodName : gridseed

Source :
```
method gridseed()
        statslib.grid_seed()
endmethod
```

Object : MainPage.Floraretbutton

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
if isAssigned(floraretFm) then
        floraretFm.bringtoTop()
        floraretFm.wait()
        floraretFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmsLib.setIsCalledTrue()
        if floraretFm.Open("floraret.fdl", WinStyleMaximize) then
                floraretFm.Wait()
                floraretFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Flora Return form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : MainPage.endorsebttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
if isAssigned(endorseFm) then
        endorseFm.bringtoTop()
        endorseFm.wait()
        endorseFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if endorseFm.Open("endorsfm.fdl", WinStyleMaximize) then
```

```
                endorseFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Endorsement form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : MainPage.reportprocessbutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(reportsFm) then
        reportsFm.bringtoTop()
        reportsFm.wait()
        reportsFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if reportsFm.Open("report.fdl", WinStyleMaximize) then
                reportsFm.Wait()
                reportsFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Reports form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : MainPage.helpButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        helpShowIndex(helpFile)
endMethod
```

Object : MainPage.exitButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
choice = msgQuestion("Quit F*I*D*B*M*S*", "Do you want to quit " + "the Statistics Module?")
if choice= "Yes" then
        okToExit = True
        close()
else
        eventInfo.setErrorCode(cannotDepart)
endif
endmethod
```

Object : MainPage.licencebutton

MethodName : pushButton

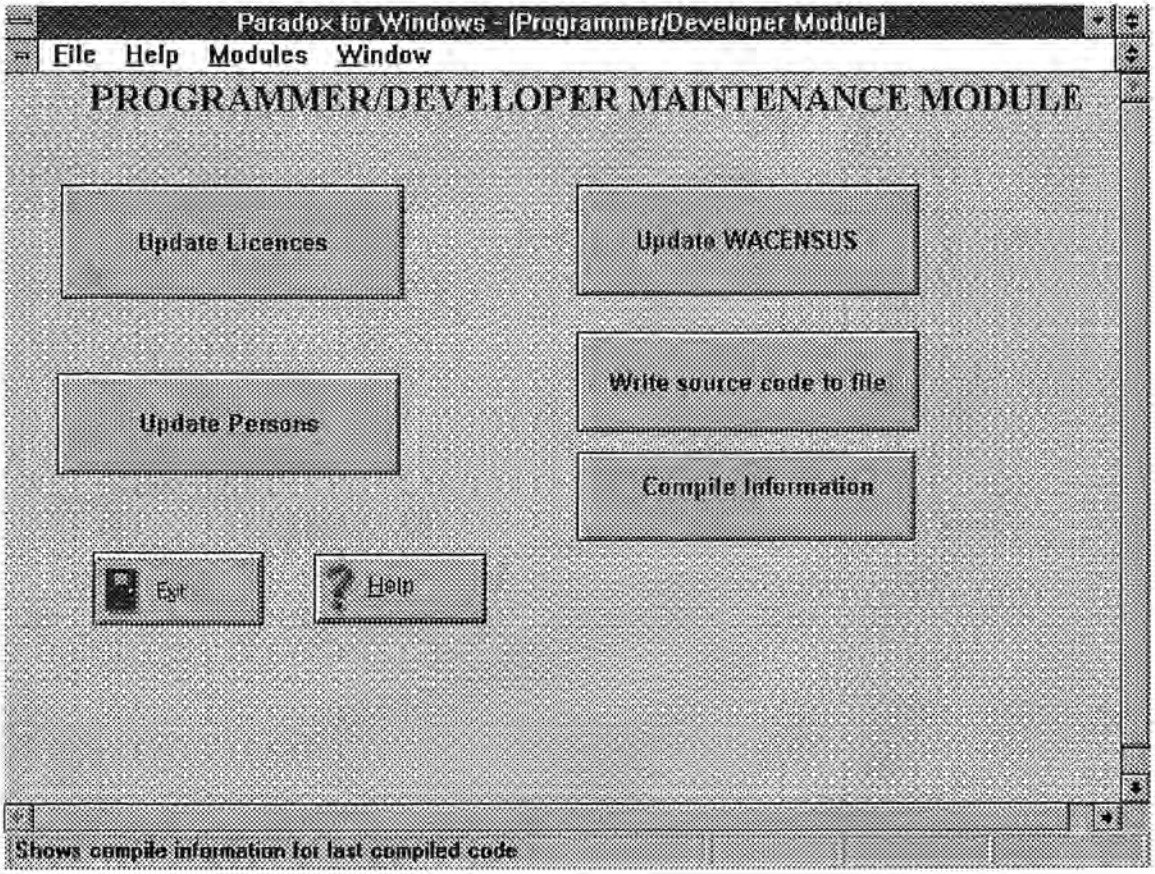
Source :

```
method pushButton(var eventInfo Event)
if isAssigned(licenceFM) then
        licenceFm.bringtoTop()
        licenceFm.wait()  ; put licence into wait state
        licenceFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
```

```
                licenceFm.Wait()
                licenceFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the licence form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Programmer/Developer Form

This form allows the FIDMS Database Management System developer to maintain currency of tables which are downloaded as read-only files from the VAX ORACLE system. The form appears as shown below (but in colour).



PROG_DEV.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Methods (PROG_DEV.FSL)	**Objects**
Uses	mainPage
Var	
Const	
menuAction	
mouseEnter	
mouseExit	
mouseRightUp	

Methods (mainPage)	**Objects (mainPage)**
Var	endorseBttn
Const	exitButton
	floraretbutton
arrive	licence_button
menuAction	processButton
open	prog_dev
	reportprocessbutton
	update_lic
	update_pers
	update_wacensus

Object : prog_dev

MethodName : Uses

Source :
```
Uses ObjectPal
        clearPageValues(formName String, pageName String)
        createObjMenu(var formName Form, pageName String, uio UIObject)
        FIDMSObjMsg(formName String, pageName String, objID String)
        pageNameOf(ui UIObject) String
        setIsCalledTrue()
        setIsCalledFalse()
        formCanClose()
endUses
```

Object : prog_dev

MethodName : Var

Source :
```
Var
        uioName, choice String
        uio UIObject
        formName, floradbFm, floraretFm, licenceFm, reportsFM,
          endorseFM, aboutFm, statsFM, progFM Form
        fidmslib Library
        closeVCRLg Logical
        okToExit Logical
endVar
```

Object : prog_dev

MethodName : Const

Source :
```
Const
        helpFile   = ":fidms:floradb.hlp" ; path to Help file
        headerHelp = LongInt(20006)
        detailHelp = LongInt(8)
        proghelp   = ":fidms:fidprog.hlp" ; path to programmer help file
endConst
```

Object : prog_dev

MethodName : mouseEnter

Source :
```
method mouseEnter(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio)  ; what is the object?
        uioName = uio.Name
        if uio.name <> self.name AND  ; eliminate message if form
          uio.class = "Field" OR uio.class = "Multirecord" OR uio.class = "Button") then
                fidmslib.FIDMSObjMsg(formName.name, fidmslib.pageNameOf(uio), uioName)
        endif
endif
endmethod
```

Object : prog_dev

MethodName : mouseExit

Source :
```
method mouseExit(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio) ; what is the object?
        if uio.class <> "Text" and uio.class <> "Bitmap" then
                message("")
        endif
endif
endmethod
```

Object : prog_dev

MethodName : mouseRightUp

Source :
```
method mouseRightUp(var eventInfo MouseEvent)
if eventInfo.isPreFilter() then
        eventInfo.getTarget(uio) ; what is the object?
        if uio.name <> "calmlogoBox" then
                while uio.class = "Text" or uio.class = "Bitmap"
                        uio.attach(uio.ContainerName)
        endWhile
        endif
        fidmslib.createObjMenu(formName, fidmslib.pageNameOf(uio), uio)
endif
endmethod
```

Object : prog_dev

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
if not eventInfo.isPreFilter() then
        if eventInfo.reason() = menuNormal and
          eventInfo.id() = MenuCanClose and not okToExit then
        eventInfo.setErrorCode(CanNotDepart)
        disableDefault
        message("Use the Exit button to exit PROG_DEV.")
        endIf
endIf
endmethod
```

Object : mainpage

MethodName : Var

Source :
```
Var
        wtext,maintext array[12] String
endVar
```

Object : mainpage

MethodName : Const

Source :
```
Const
  #IDH_MENUS_WINDOW_CURRENTWINDOW = "Current Open Windows"
  #IDH_MENUS_WINDOWS = "&Window"

  HelpHelp = 101 ; Help | Using Help command
  HelpList = 102 ; Help | Help Index command
  Help_Use = 103 ; Help | System Help command
  Prog_help = 104 ; Help | Programmer Help

  Florarets = 201 ;Modules | Flora Returns command
  Licences = 202 ; Modules | Licences command
  Stats    = 202 ; Modules | Statistics command
  Reports  = 203 ; Modules | Reports command
  Endorse  = 204 ; Modules | Endorsements command

  file_print = 300
  Exit     = 301 ; Exit command
  file_print_setup = 302
endConst
```

Object : mainpage

MethodName : open

Source :
```
method open(var eventInfo Event)
delayScreenUpdates(Yes)
if not isFile("prog_dev.fsl") then
        msgInfo("Startup Error!", "The FIDMS application files must be " +
        "in the working directory.")
        close()
```

```
endif
if not fidmslib.open("fidmslib.ldl", globalToDesktop) then
        msgStop("Failure", "fidmslib could not be opened")
endif
wText[1]=#IDH_MENUS_WINDOW_CURRENTWINDOW
mainText[5]=#IDH_MENUS_WINDOWS
formName.attach()
hideSpeedBar()
maximize()
okToExit = False
delayScreenUpdates(No)
Message("Loading Programmer/Developer Form.  One moment, please...")
delayScreenUpdates(Yes)
if not isFile("prog_dev.fsl") then
        disableDefault
        beep()
        msgStop("Directory Error", "The Paradox for Windows " +
        "working directory must be set to the directory " +
        "containing this form, for example: " +
        "T:\\fidms\\")
        close()
        blankAsZero(No)
        fidmslib.open("fidmslib.ldl", globalToDesktop)
        okToExit = False
        doDefault
        self.postAction(dataBeginEdit)
        delayScreenUpdates(No)
endIf
endmethod
```

Object : mainpage

MethodName : arrive

Source :

```
method arrive(var eventInfo MoveEvent)
var
        mainMenu        Menu    ; The main menu
        filemenu,
        HelpMenu,
        ModuleMenu,
        windowMenu      popupmenu   ; The Help menu
endVar

 fileMenu.addtext ("&Print", Menuenabled, Usermenu + file_print)
 fileMenu.addtext ("Printer &Setup", MenuEnabled, Usermenu + File_print_setup)
 filemenu.addtext ("E&xit", menuEnabled, userMenu + exit)
 mainmenu.addpopup ("&File", filemenu)

 HelpMenu.addText("Using &Help\tCtrl+F1", menuEnabled, userMenu + helpHelp)
 HelpMenu.addText("Help &Index\tShift+F1", menuEnabled, userMenu + helpList)
 HelpMenu.addText("&Using this Form\tF1", menuEnabled, userMenu + help_Use)
 HelpMenu.addText("&Programmer Help", menuEnabled, userMenu + Prog_help)
 MainMenu.addPopUp("&Help", HelpMenu)

 ModuleMenu.addtext ("&Flora Returns", menuEnabled, userMenu + Florarets)
 ModuleMenu.addtext ("&Statistics", menuEnabled, userMenu + Stats)
 ModuleMenu.addtext ("&Reports", menuEnabled, userMenu + Reports)
 ModuleMenu.addtext ("&Licences", menuEnabled, userMenu + Licences)
 ModuleMenu.addtext ("&Endorsements", menuEnabled, UserMenu + Endorse)
 MainMenu.addpopup("&Modules",ModuleMenu)

 windowmenu.addStaticText(wText[1])
 mainMenu.addPopUp(mainText[5],windowmenu)
 mainMenu.show()
 setTitle("Programmer/Developer Module")

endmethod
```

Object : mainpage

MethodName : menuAction

```
method menuAction(var eventInfo MenuEvent)
```

Source :

```
var
  menu_Cmd smallInt ; The constant returned by a menu command
  editTest Logical  ; Flag variable used to test the type of
                    ; object that is active
  calledBy Form     ; Holds the handle of the form that opened
                    ; this form
  d_search        DDE
endVar
menu_Cmd = eventInfo.id()
switch
        case menu_Cmd = MenuControlClose :
        disableDefault
        exitbutton.pushButton()
        case menu_Cmd = MenuControlMaximize or
        menu_Cmd = MenuControlMinimize :
        If formcaller(calledBy) then
        disableDefault
        beep()
        msgInfo("Oops!", "Because this form was opened by the " +
                calledBy.getTitle() + " form, the Maximize and " +
                "Minimize buttons are disabled.  Sorry...")
        endIf
        case menu_Cmd = userMenu + Florarets : Floraretbutton.pushbutton()
        case menu_Cmd = userMenu + Licences    : Licencebutton.pushbutton()
        case menu_Cmd = userMenu + Stats     : processButton.pushbutton()
        case menu_Cmd = userMenu + Reports   : reportprocessbutton.pushbutton()
        case menu_Cmd = userMenu + Endorse   ; endorsebttn.pushbutton()
        case menu_Cmd = userMenu + Exit     : exitButton.pushbutton()
        case menu_Cmd = userMenu + HelpHelp  : helpOnHelp()
        case menu_Cmd = userMenu + Prog_help : helpShowindex(progHelp)
        case menu_Cmd = userMenu + HelpList  : helpshowContext(helpfile,1000)
        case menu_Cmd = userMenu + Help_Use : helpShowContext(helpFile, detailHelp)
        case menu_cmd = usermenu + file_print: active.menuAction(menuFilePrint)
        case menu_cmd = usermenu + file_print_setup: active.menuAction(menuFilePrinterSetup)
endswitch
endmethod
```

Object : mainpage.Compile

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        dynCompileInfo Dynarray[]  AnyType
endVar
compileInformation(dynCompileInfo)
dynCompileInfo.view()
endmethod
```

Object : mainpage.Floraretbutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(floraretFm) then
        floraretFm.bringtoTop()
        floraretFm.wait()  ; put floraretFm into wait state
        floraretFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmsLib.setIsCalledTrue()
        if floraretFm.Open("floraret.fdl", WinStyleMaximize) then
                floraretFm.Wait()  ; put floraretFm into wait state
                floraretFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Flora Return form is not available")
                fidmslib.setIsCalledFalse()
```

```
            endif
endif
endmethod
```

Object : mainpage.licencebutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(licenceFM) then
            licenceFm.bringtoTop()
            licenceFm.wait()  ; put licence into wait state
            licenceFm.hide()
            maximize()
            mainPage.moveTo()
else
            fidmslib.setIsCalledTrue()
            if licenceFM.Open("licence.fdl", WinStyleMaximize) then
                        licenceFm.Wait()  ; put licence into wait state
                        licenceFm.hide()
                        maximize()
                        mainPage.moveTo()
            else
                        msgInfo("Status", "Sorry, the licence form is not available")
                        fidmslib.setIsCalledFalse()
            endif
endif
endmethod
```

Object : mainpage.processButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(statsFM) then
            statsFM.bringtoTop()
            statsFM.wait()  ; put statsFM into wait state
            statsFM.hide()
            maximize()
            mainpage.moveTo()
else
            fidmslib.setIsCalledTrue()
            if statsFM.Open("Stats.FdL", WinStyleMaximize) then
                        statsFM.Wait()  ; put statsFM into wait state
                        statsFM.hide()
                        maximize()
                        mainpage.moveTo()
            else
                        msgInfo("Status", "Sorry, the Statistics form is not available")
                        fidmslib.setIsCalledFalse()
            endif
endif
endmethod
```

Object : mainpage.reportprocessbutton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(reportsFm) then
            reportsFm.bringtoTop()
            reportsFm.wait()  ; put reportsFm into wait state
            reportsFm.hide()
            maximize()
            mainPage.moveTo()
else
            fidmslib.setIsCalledTrue()
            if reportsFm.Open("report.fdl", WinStyleMaximize) then
                        reportsFm.Wait()  ; put reportsFm into wait state
                        reportsFm.hide()
                        maximize()
```

```
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Reports form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.endorsebttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if isAssigned(endorseFm) then
        endorseFm.bringtoTop()
        endorseFm.wait()  ; put endorseFm into wait state
        endorseFm.hide()
        maximize()
        mainPage.moveTo()
else
        fidmslib.setIsCalledTrue()
        if endorseFm.Open("endorsfm.fdl", WinStyleMaximize) then
                endorseFm.Wait()  ; put endorseFm into wait state
                endorseFm.hide()
                maximize()
                mainPage.moveTo()
        else
                msgInfo("Status", "Sorry, the Endorsement form is not available")
                fidmslib.setIsCalledFalse()
        endif
endif
endmethod
```

Object : mainpage.helpButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        helpShowIndex("floradb.hlp")
endMethod
```

Object : mainpage.exitButton

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event
choice = msgQuestion("Quit F*I*D*B*M*S*", "Do you want to quit " +
      "the application?")
if choice= "Yes" then
        okToExit = True
        close()
else
        eventInfo.setErrorCode(cannotDepart)
endif
endmethod
```

Object : mainpage.enum_button

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        floradbform   Form
        thechoice string
        temprpt   report
        formname    string
endVar
formname = ""
formname.view ("Enter name of form (include .fsl extension)")
if formname <> "" then
        floradbform.open(formname)          ; open another form
```

```
        floradbForm.enumSource("tempsrc.db", True)
        okToExit = true
        floradbform.exitbutton.pushbutton()                ; close the form
else
        msginfo("Error", "You have not chosen a form")
endif
thechoice = msgquestion ("Source Code", "Do you wish to view the code?")
switch
        case thechoice = "Yes" :  temprpt.open ("tempsrc.rdl")
        temprpt.maximize()
        otherwise : return
endswitch
endMethod
```

Object : mainpage.update_wacensus

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        updatehbttaxon Query
        updatehbttrans Query
        ansTbl Table
        destTbl String
endVar
destTbl = "hbttaxon.db"
updatehbttaxon.readFromFile("hbttaxon.qbe")
if updatehbttaxon.executeQBE() then       ; if the query succeeds
        ansTbl.attach(":PRIV:Answer.db")
        if isTable(destTbl) then
                if NOT ansTbl.add(destTbl) then
                        errorShow()
                endif
        else
                msgStop("Error", "Can't find " + destTbl + ".")
        endif
else
        errorShow("Query failed.")
endIf
destTbl = "hbttrans.db"
updatehbttrans.readFromFile("hbttrans.qbe")
if updatehbttrans.executeQBE() then       ; if the query succeeds
        ansTbl.attach(":PRIV:Answer.db")
        if isTable(destTbl) then
                if NOT ansTbl.add(destTbl) then
                        errorShow()
                endif
        else
                msgStop("Error", "Can't find " + destTbl + ".")
        endif
else
        errorShow("Query failed.")
endIf
endMethod
```

Object : mainpage.update_pers

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        updatepers Query
        ansTbl Table
        destTbl String
endVar
destTbl = "persons.db"
updatepers.readFromFile("persons.qbe")
if updatepers.executeQBE() then       ; if the query succeeds
        ansTbl.attach(":PRIV:Answer.db")
        if isTable(destTbl) then
                if NOT ansTbl.add(destTbl) then
                        errorShow()
```

```
                endif
        else
                msgStop("Error", "Can't find " + destTbl + ".")
        endif
else
        errorShow("Query failed.")
endIf
endMethod
```

Object : mainpage.update_lic

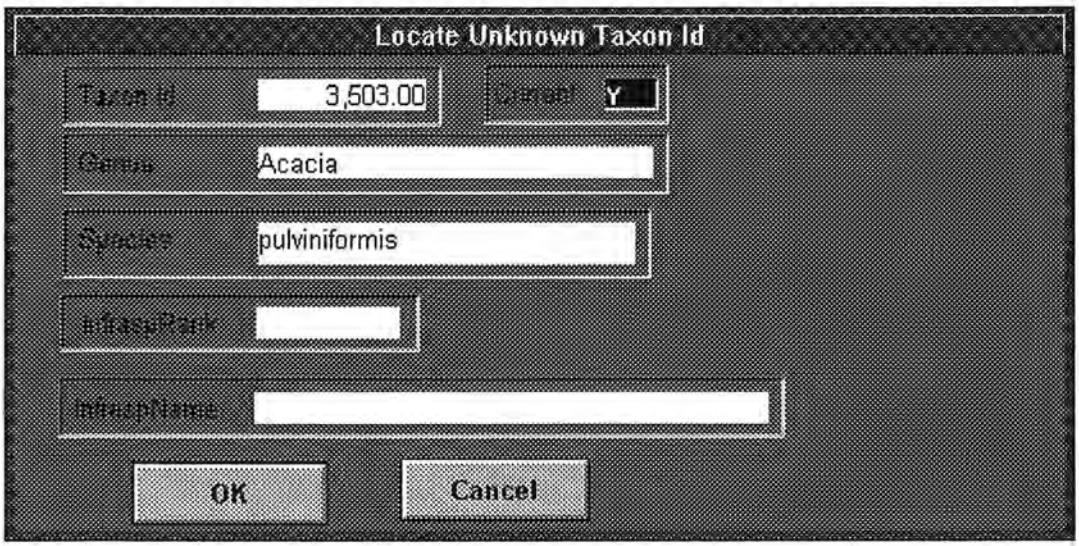
MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
var
        updatelic Query
        updateloc Query
        ansTbl Table
        destTbl String
endVar
destTbl = "licences.db"
updatelic.readFromFile("licence.qbe")
if updatelic.executeQBE() then      ; if the query succeeds
        ansTbl.attach(":PRIV:Answer.db")
        if isTable(destTbl) then
                if NOT ansTbl.add(destTbl) then
                        errorShow()
                endif
        else
                msgStop("Error", "Can't find " + destTbl + ".")
        endif
else
        errorShow("Query failed.")
endIf
destTbl = "loc.db"
updateloc.readFromFile("location.qbe")
if updateloc.executeQBE() then      ; if the query succeeds
        ansTbl.attach(":PRIV:Answer.db")
        if isTable(destTbl) then
                if NOT ansTbl.add(destTbl) then
                        errorShow()
                endif
        else
                msgStop("Error", "Can't find " + destTbl + ".")
        endif
else
        errorShow("Query failed.")
endIf
endMethod
```

Species Locate Dialog Box

The Species Locate (SPLOC.FSL) Dialog box is opened when the user requests the taxon id for a species where there is no known taxon id. The form appears as shown below (but in colour).



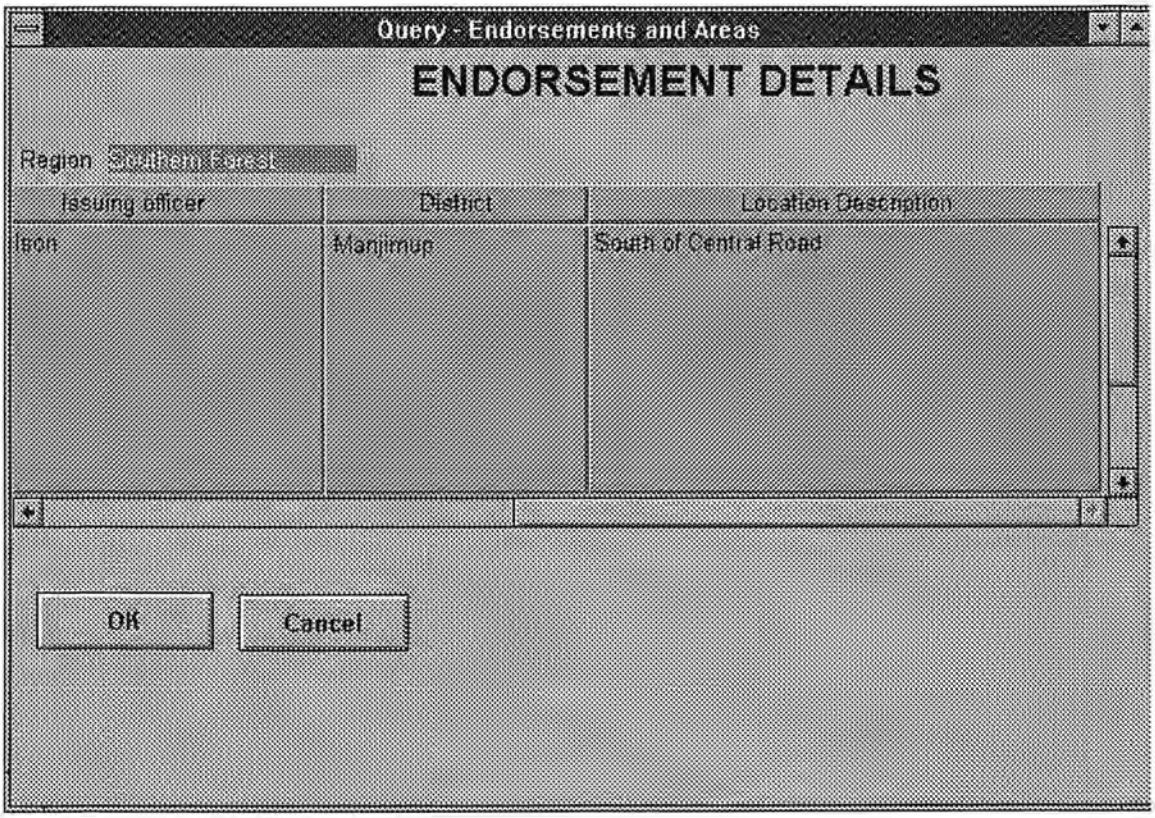
SPLOC.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Objects (SPLOC.FSL)
cancelButton
okButton

Endorsements by Region/District Dialog Box

EndRep displays in the application as the Endorsements dialog box, which is used to display the results of a user-defined query by CALM Region or CALM District generated by the Report Form. The form appears as shown below (but in colour).



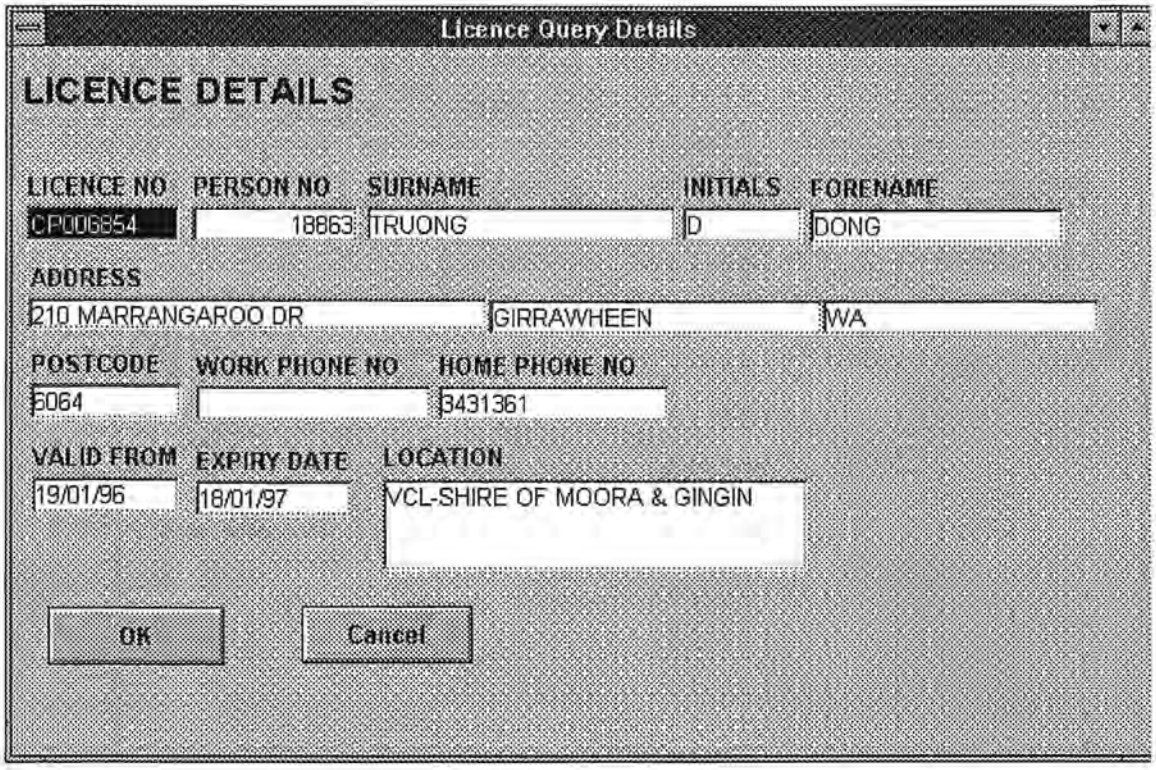
END_REP.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Objects (END_REP.FSL)
cancelButton
okButton

Licence Locate Dialog Box

The GEN_LIC Dialog box is opened to show the results of a licence search of a user-defined licence number in the Person and Licensing Form mainPage find_lic method. The form appears as shown below (but in colour).



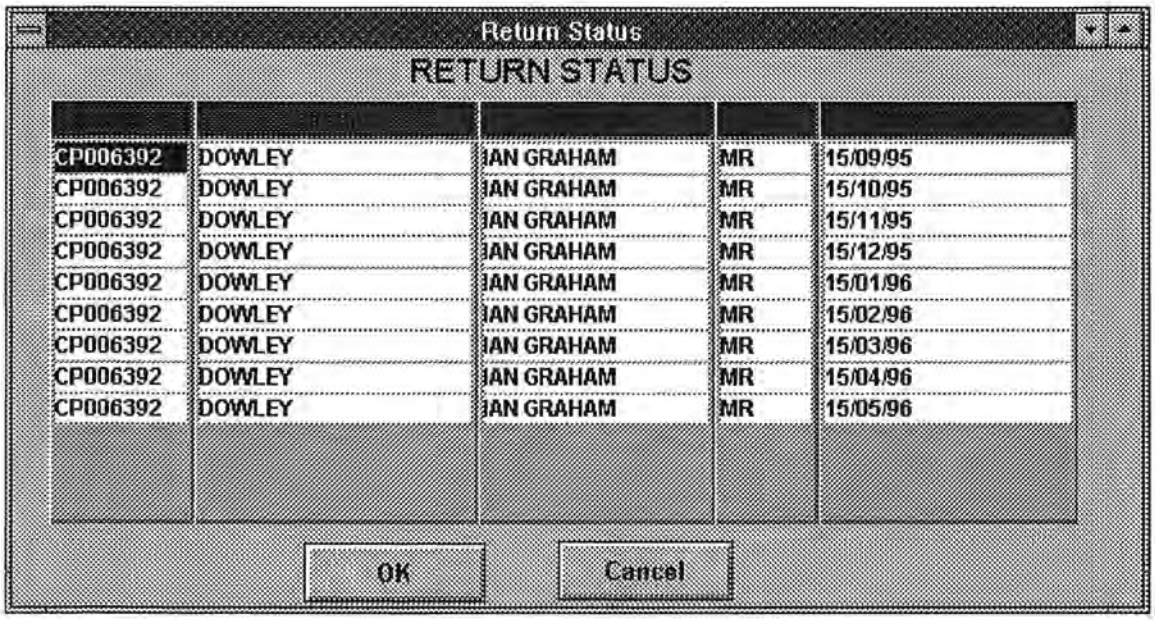
GEN_LIC.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Objects (GEN_LIC.FSL)
cancelButton
okButton

Return Status Dialog Box

The RetStat Dialog box is opened when the user requests the return status for a specified licence number. This data is required before issuing a licence. The form appears as shown below (but in colour).



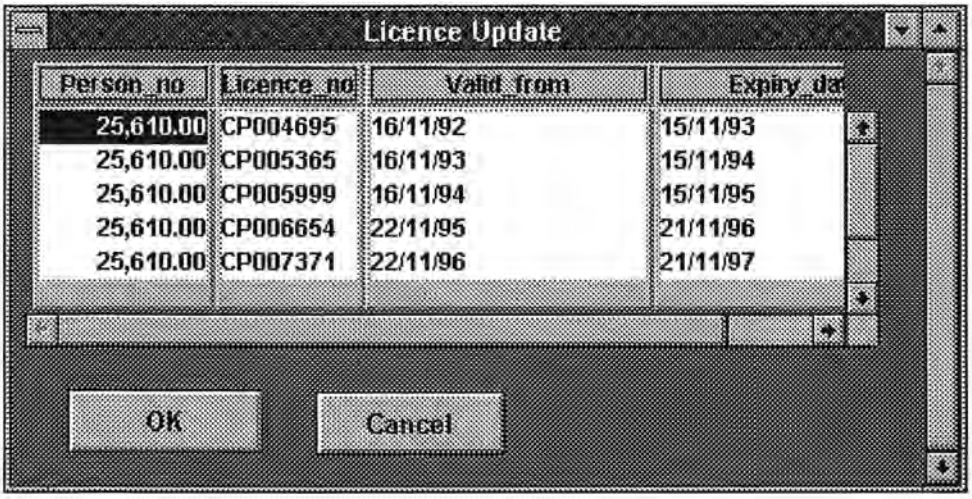
RETSTAT.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Objects (GEN_LIC.FSL)
cancelButton
okButton

Licence Update Dialog Box

The LICLOC Dialog box is opened when a date which is not valid for the active licence number is entered. The dialog box shows all licences held by the person and their valid period so the user can choose the correct licence number. The form appears as shown below (but in colour).



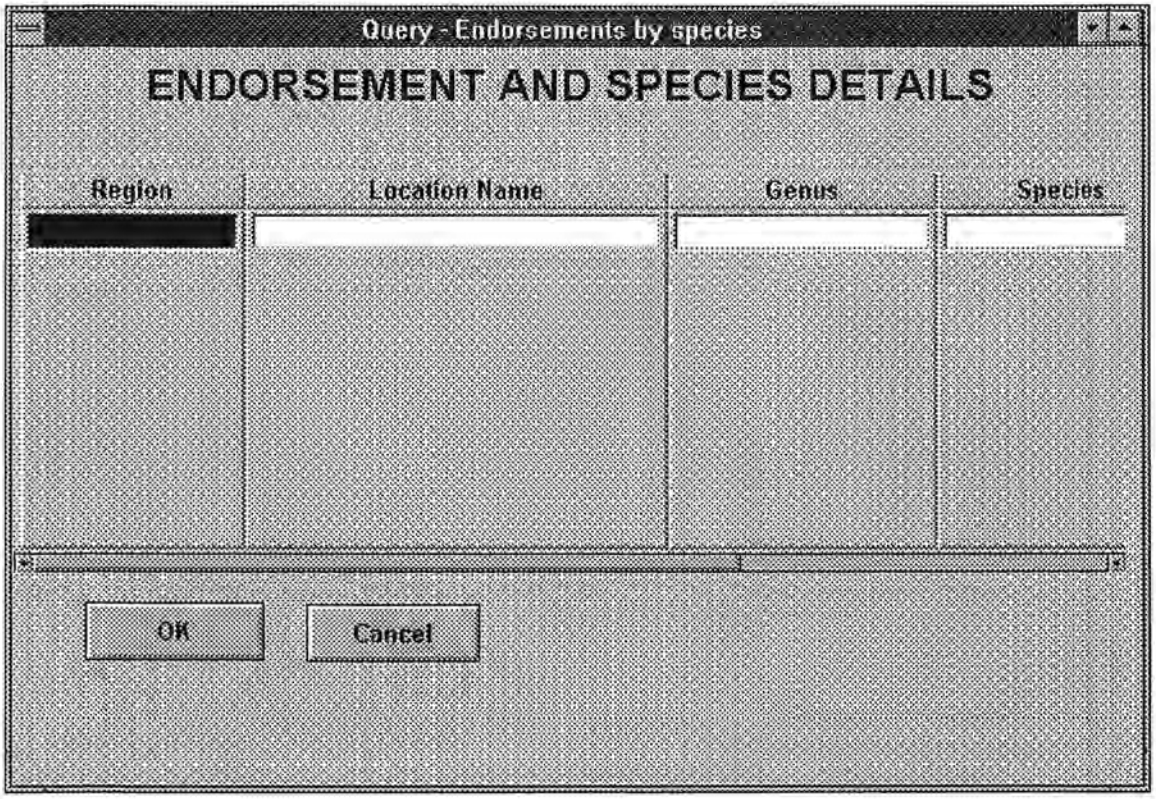
LICLOC.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Objects (LICLOC.FSL)
cancelButton
okButton

Endorsement and Species Details Dialog Box

End_spec displays in the application as the Endorsements and Species Details dialog box, which is used to display the results of a user-defined query of species details by CALM Region or CALM District generated by the Report Form. The form appears as shown below (but in colour).



END_SPEC.FSL Methods and Objects

Methods and objects attached to the form are as follows:

Objects (END_SPEC.FSL)
cancelButton
okButton

cancelButton

CancelButton lets the user cancel the operation and return to the calling form without returning a value.

```
method pushButton(var eventInfo Event)
      formreturn("Cancel")
endmethod
```

okButton

OKButton lets the user return a value to the calling form.

```
method pushButton(var eventInfo Event)
      formreturn("OK")
endmethod
```

Libraries

Libraries

A library is a collection of custom methods. Libraries are useful for storing and maintaining frequently used routines, and for sharing custom methods and variables among several forms. FIDMS uses library methods to display Help and menus and to perform some queries in the Statistics and Report Forms. Go to a topic for information on a library and its code.

- **FIDMSLIB.LSL Library (see p 72)**
- **GEOLIB.FSL Library (see p 102**
- **STATS.LSL Library (see p 82**
- **REPLIB.LSL Library (see p 139)**

FIDMSLIB.LSL Library

FIDMSLIB contains custom methods for displaying help for the user and the programmer.

Var

Defines variables used throughout the FIDMS Library methods. See Common Variables for more information on common variables used throughout FIDMS.

```
Var
      #canClose, #isCalled, VCROpenLg Logical
      TitleAndSource DynArray[] String
      ; 2 fields: LookUpTitle and ListSource
      myMsgTc, objHelpTc Tcursor
      qRpt Report
      aboutFm Form
endVar
```

Const

Defines constants used by FIDMS to display help

```
Const
      proghelp = ":fidms:fidmsprog.hlp"
endConst
```

clearPageValues

Clears any previous Query Value for the page (given by the pageName argument) of the form given by the formName argument. The two arguments map to a Query ID through QBEMAP.DB. The Query ID then locates the entry in QBEVAL.DB. Called prior to getPageValues.

```
method clearPageValues(formName String, pageName String)
var
      myQuery           Query
endVar

myQuery=Query

qbemap.db    | Form Name | Page ID    | Query ID  |
             | ~formName | ~pageName | _qID      |

qbeval.db    | Query ID        | Query Value    |
             | _qID            | changeto blank |

EndQuery

executeQBE(myQuery)

endmethod
```

createObjMenu

Creates and displays an object-sensitive pop-up menu. Called with formName, pageName, and UIObject arguments passed by the formName.mouseRightUp method. The target object is determined by the focus (mouse cursor position, for example) but some objects (such as texts or bitmaps) are excluded by mouseRightUp for particular forms. An object menu is only created and displayed if an entry for the target object is located in MSGHELP.DB. The standard menu option is Code Help and Programmer Help. Other options depend on the target object.

```
method createObjMenu(var formName Form, pageName String, uio UIObject)

; This is generalized object menu system called by mouseRightUp methods.
; It provides object help by locating the object's name in MsgHelp.db or
; MsgUser.db and then offers the user a PopUpMenu with "Object Help" or
; "Programmer Help" as the standard options. If the user selects "Object
; Help" from the menu, an info box is displayed containing the object's
; help text contained in Msghelp.

; STEPS:
; First, determine if MsgHelp is successfully opened and, if it is,
; locate the object in the table. Not all objects have help. If the
; object is found in MsgHelp, display a standard object help Popupmenu.
; Next, add to the PopUpMenu any additional items that only pertain to
; the object. Finally, process the menu choice that the user selects.

var
      p PopUpMenu
      choice String
endvar

; opened object help table already --
; now locate matching formName.pageName.object

if objUsrHelpTc.Locate("Form Name",formName.Name,
   "Page Name",pageName,"Object ID",uio.name)
   and objProgHelpTc.locate("Form Name",formName.Name,"Page Name",
 pageName,"Object ID",uio.name) then
            p.empty() ; First, empty old items
            p.addStaticText("Object Properties")
            p.addSeparator()
            p.addText("Programmer Code Help")
            p.addtext("User Help")
            p.addText ("About FIDMS")
            choice=p.Show()
      switch
            case choice ="Programmer Code Help"
:helpShowContext(proghelp,
               objProgHelptc."Context ID")
            case choice="User Help":helpShowContext(userhelp,
               objUsrHelpTc."Context ID")
            case choice = "About FIDMS"     : setIsCalledTrue()
            if aboutFm.open("Abtfidms.fsl") then
                  aboutFm.Wait()
                  aboutFm.Close()
                  setIscalledFalse()
            endif
      endswitch
```

```
else
      beep()
      msgStop("No Help Found", "Form: " + formName.Name + " Page: " +
              pageName + " Object: " + uio.Name)
      Return
endif
endmethod
```

Endorse_button

Opens endorsFM.FSL, maximizes it, and gives it focus.

```
method endorse_button()
; open endorseFm.fsl

; endorsFm already open -- display it and put it into wait state
if isAssigned(endorseFM)
  then
      endorseFM.bringtoTop()
      endorseFm.wait()  ; put endorseFm into wait state
      ; user returned -- hide endorseFm
      endorseFm.hide()
      maximize()
      else
  ; endorse not open -- open it
  setIsCalledTrue()
  if endorseFm.Open("endorsfm", WinStyleMaximize)
      then
       endorseFm.Wait()  ; put endorseFm into wait state
       ; user returned -- hide endorseFm
       endorseFm.hide()
    maximize()
      else
       ; open failed
       msgInfo("Status", "Sorry, the Endorsement form is not available")
       setIsCalledFalse()
  endif
endif

endmethod
```

FIDMSObjMsg

Opens MSGHELP.DB and MSGUSR.DB and tries to locate an entry matching the given formName, pageName, and ObjID arguments. If found, the Message Text value is displayed in a message.

```
method FIDMSObjMsg(formName String, pageName String, ObjID String)

;locate matching formName.pageName.object

If myMsgTc.locate("Form Name", formName, "Page Name",
   pageName, "Object ID", ObjID) or myUsrMsgTc.locate("Form Name",
    formName, "Page Name", pageName, "Object ID", ObjID) then
      message(myMsgTc."Message Text".value)
```

```
else
      message ("Couldn't find object ", FormName, PageName, ObjId)
      sleep (5000)
endif
endmethod
```

Floraret Button

Opens floraret.FSL, maximizes it, and gives it focus.

```
method FLORARET_BUTTON()
; open floraretFm.fsl

; floraretFm already open -- display it and put it into wait state
if isAssigned(floraretFm)
  then
      floraretFm.bringtoTop()
      floraretFm.wait()  ; put floraretFm into wait state
      ; user returned -- hide floraretFm
      floraretFm.hide()
      maximize()
      reportsfm.moveTo()

else
  ; floraret not open -- open it
  setIsCalledTrue()
  if floraretFm.Open("floraret", WinStyleMaximize)
      then
       floraretFm.Wait()  ; put floraretFm into wait state
       ; user returned -- hide floraretFm
       floraretFm.hide()
    maximize()
       floraretfm.moveTo()
  else
       ; open failed
       msgInfo("Status", "Sorry, the floraret form is not available")
       setIsCalledFalse()
  endif

endif

endmethod
```

formCanClose

Sets a flag to indicate that a form can be closed.

```
method formCanClose()
      canClose = True
endMethod
```

getDataSource

Returns the item stored at the ListSource index of the TitleAndSource DynArray declared in FIDMSLIB.LSL. See Also setDataSource

```
method getDataSource() String
      return TitleAndSource["ListSource"] ; return value stored in
DynArray
endmethod
```

getLookUpTitle

Returns the item stored at the LookUpTitle index of TitleAndSource a DynArray declared in FIDMSLIBB.LSL. See Also setLookUpTitle

```
method getLookUpTitle() String
      return(TitleAndSource["LookUpTitle"])
endmethod
```

isCanClose

Checks the value of the variable canClose to report whether a form can close.

```
method isCanClose() Logical
      ; is floradb asking for a close
      if isAssigned(canClose) then
            return canClose
      ; floradb hasn't asked for close -- cannot close form
      else
            return False
      endif
endmethod
```

isVCROpen

Reports whether a custom SpeedBar is open.

```
method isVCROpen() Logical
      ; is VCR open?
      if isAssigned(VCROpenLg) then ; Videobar has been opened
            return VCROpenLg  ; Videobar has not been opened
      else
            return False
      endif
endmethod
```

Licence_Button

Opens licence.FSL, maximizes it, and gives it focus.

```
method LICENCE_BUTTON()
; open licenceFm.fsl

; licenceFm already open -- display it and put it into wait state
if isAssigned(licenceFm) then
      licenceFm.bringtoTop()
      licenceFm.wait()  ; put licenceFm into wait state
      ; user returned -- hide licenceFm
      licenceFm.hide()
      maximize()
      reportsfm.moveTo()

else
  ; licence not open -- open it
  setIsCalledTrue()
  if licenceFm.Open("licence", WinStyleMaximize)
      then
       licenceFm.Wait()  ; put licenceFm into wait state
       ; user returned -- hide licenceFm
       licenceFm.hide()
    maximize()
       reportsfm.moveTo()
  else
       ; open failed
       msgInfo("Status", "Sorry, the licence form is not available")
       setIsCalledFalse()
  endif

endif
endmethod
```

listPageObjects

Gets the values of objects on the current page and writes them to a table.

```
method listPageObjects(formName String, pageName String) Logical

var
     pageObjsQBE Query
endVar

pageObjsQBE = Query

qbeMap.db    | Form Name | Page ID   | Object Path |
             | ~formName | ~pageName | Check       |

endQuery

executeQBE(pageObjsQBE, "Objlist.db")
RETURN TRUE
endmethod
```

open

Opens MSGHELP.DB and MSGUSR.db, tables that store help messages.

```
method open(var eventInfo Event)
if not objProgHelpTc.Open("Msghelp.db") then
      msgStop("Problem", "Couldn't open MsgHelp.db")
endif
if not myMsgTC.open("Msghelp.db") then
      msgStop("Problem", "Couldn't open MsgHelp.db")
endif
if not myUsrMsgTC.open("MsgUsr.db") then
      msgStop("Problem", "Couldn't open MsgUsr.db")
endif
if not objUsrHelpTc.Open ("msgusr.db") then
      msgStop("Problem", "Couldn't open msgusr.db")
endif
endmethod
```

pageNameOf

Returns the name of the page that contains the UIObject specified in the argument ui. A null string is returned if no containing page is found.

```
method pageNameOf(ui UIObject) String

; this method returns page name for the current object
var
      s string
endvar
while ui.Class <> "Page" and ui.ContainerName <> ""
      ui.attach(ui.ContainerName)
endWhile
if ui.class = "Page" then
      s = ui.name
else
      s = ""
endif
return s
endmethod
```

returnisCalled

Returns the value (True or False) assigned to the FIDMSLIB variable isCalled. Returns False if no value has been assigned. isCalled is tested by the open methods of the dialog box ABTFIDMS to ensure that this forms is opened only by FLORADB.

```
method returnIsCalled() Logical
if not isCalled.isAssigned()then
      isCalled = False
endif
return isCalled
endmethod
```

setDataSource

Assigns the lookSrc string argument to the ListSource index of TitleAndSource, a DynArray declared in FIDMSLIB.LSL. See Also getDataSource

```
method setDataSource(const lookSrc String)
      TitleAndSource["ListSource"] = lookSrc
endmethod
```

setIsCalledTrue

Sets the FIDMSLIB variable isCalled to True. isCalled is used to prevent the opening of certain forms directly rather than by FLORADB. See Also returnisCalled

```
method setIsCalledTrue()
      isCalled = True
endmethod
```

setIsCalledFalse

Sets the FIDMSLIB variable isCalled to False. isCalled is used to prevent the opening of certain forms directly rather than by FLORADB. See Also returnisCalled

```
method setIsCalledFalse()
      isCalled = False
endmethod
```

setLookUpTitle

Assigns the lookTtl string argument to the LookUpTitle index in TitleAndSource DynArray (declared in FIDMSLIB.LSL). See Also getLookUpTitle

```
method setLookUpTitle(const lookTtl String)
      TitleAndSource["LookUpTitle"] = lookTtl ; set value of DynArray
item
endmethod
```

setPageValues

```
method setPageValues(formName String, pageName String) Logical

; This method runs a query (mapValQBE) to find all value objects on
; the current form & page, sending the results to a tCursor named
; newValsTc.

var
      newValsTC, valCursor TCursor
```

```
      mapValQBE Query
      fieldObj UIObject
endVar

clearPageValues(formName, pageName) ; clear QBEVAL.DB Query Value values
mapValQBE = Query

qbeMap.db    | Form Name |Page ID     | Query ID  | Object Path |
             | ~formName |  ~pageName | _qID      | Check       |

qbeVal.db    | Query ID   | Field Name |
             | _qID       | Check      |

endQuery
executeQBE(mapValQBE, newValsTc)
valCursor.open("qbeVal.db")
valCursor.edit()
scan newValsTC :
      valCursor.locate("Field Name", newValsTC."Field Name")
      fieldObj.attach(newValsTC."Object Path")
      if not isBlank(fieldObj.value) then
            if isBlank(valCursor."Query Value") then
                  valCursor."Query Value" = fieldObj.value
            else
                  valCursor."Query Value"=valCursor."Query Value"+
                     " OR "+fieldObj.value
            endIf
      endIf
endScan
valCursor.endEdit()
RETURN TRUE
endmethod
```

Stats_button

Opens stats.FSL, maximizes it, and gives it focus.

```
method STATS_BUTTON()

if isAssigned(statsFM)
  then
      statsFM.bringtoTop()
      statsFM.wait()  ; put statsFM into wait state
      ; user returned -- hide statsFM
      statsFM.hide()
      maximize()
else
  ; reports not open -- open it
  setIsCalledTrue()
  if statsFM.Open("Stats.FSL", WinStyleMaximize)
      then
       statsFM.Wait()  ; put statsFM into wait state
       ; user returned -- hide statsFM
       statsFM.hide()
    maximize()
      else
       ; open failed
```

```
        msgInfo("Status", "Sorry, the Statistics form is not available")
        setIsCalledFalse()
    endif

endif

endmethod
```

VCRisOpen

Sets a flag to indicate that a custom SpeedBar is open.

```
method VCRisOpen()
      VCROpenLg = True
endmethod
```

VCRisClosed

Sets a flag to indicate that a custom SpeedBar is closed.

```
method VCRisClosed()
      ; videobar closed
      VCROpenLg = False
endmethod
```

vcrName

Returns the text in the title bar of a custom SpeedBar.

```
method vcrName() String
; return Videbar title
      return "Click button to change record"
endmethod
```

STATSLIB.LSL Library

STATSLIB contains custom methods for generating complex reports and statistics for the user.

grid_seed()

This method calculates the relative geographic spread of seed harvesting in two user-defined years.

```
method grid_seed()
var
      tc                        TCursor
      tblcreate                 Table
      srcTbl                    Table
      destTbl                   String
      top20rsl                  Report
      top20qry                  Query
      top20yr                   Smallint
      top20compare              Smallint
      usrChoice                 String
      fullMonth Array[12]       String
      year                      String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()
tblcreate.attach (":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20com.db")

tc.open(":fidms:retnum.db")
tc.edit()
scan tc for isblank(tc.yr) :
      tc.yr = year(tc.Collecting_date)
endscan

top20yr = 0
top20yr.view ("Enter year (eg. 94) for Geographic analysis")

top20qry = query

:fidms:retnum.db  |retnum|yr              |
                  |_abcd |..~top20yr      |

:fidms:retex.db   |retnum|TaxonID|Quantity |Unit|Part  |Grid square
|
                  |_abcd |Check  |Calc Sum |kg  |seed  |Check not blank
|

endquery
```

```
message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

top20qry = query

:priv:answer.db   |Grid square|Taxonid |
                  |Calc count |Check   |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

top20qry = query

:priv:top20.db    |TaxonId     |Count of Grid Square |
                  |Check_taxid |Check as Grid_num    |

:priv:answer.db   |TaxonId   |Sum of Quantity  |
                  |_taxid    |Calc sum         |

:fidms:hbttaxon.db|TaxonId |Genus   |Species |
                  |_taxid  |Check   |Check   |

endquery

if not executeQBE(top20qry,":priv:top20int.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
   like (":priv:top20int.db")
   key "TaxonID"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20int.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertRecord()
tc.Year_of_Harvest= top20yr

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
```

```
tblcreate.delete()

top20compare = 0
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > 90 then

      top20qry = query

      :fidms:retnum.db  |retnum|yr                    |
                        |_abcd |..~top20compare       |

      :fidms:retex.db   |retnum|TaxonID|Quantity |Unit|Part|Grid square
|
                        |_abcd |Check |Calc sum |kg  |seed|Check not
blank |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry) then
            errorshow()
            close()
      endif

      tc.endedit()
      top20qry = query

      :priv:answer.db   |Grid square|Taxonid    |
                        |Calc count |Check      |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
      endif

      top20qry = query

      :priv:top20.db    |TaxonId      |Count of Grid Square |
                        |Check _taxid |Check as Grid_num    |

      :priv:answer.db   |TaxonId   |Sum of Quantity  |
                        |_taxid    |Calc sum         |

      :fidms:hbttaxon.db|TaxonId |Genus   |Species     |
                        |_taxid  |Check   |Check       |

      endquery
      if not executeQBE(top20qry,":priv:top20int.db") then
            errorshow()
      endif

      tblcreate = create(":priv:top20com.db")
            like ":priv:top20int.db"
            key "TaxonID"
      endcreate
```

```
      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20int.db")
      srcTbl.add (destTbl)

      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate

      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest=top20compare

      top20rsl.open ("gridseed.rsl")
else
      msginfo ("Error", "Invalid year has been chosen")
endif
endMethod
```

grid_stems

This method calculates the relative geographic spread of seed harvesting (by grid square) in two user-defined years.

```
method grid_stems()

var
      tc                      TCursor
      tblcreate               Table
      srcTbl                  Table
      destTbl                 String
      top20rsl                Report
      top20qry                Query
      top20yr                 Smallint
      top20compare            Smallint
      usrChoice               String
      fullMonth Array[12]     String
      year                    String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()
tblcreate.attach (":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20com.db")

tc.open(":fidms:retnum.db")
tc.edit()
scan tc for isblank(tc.yr) :
      tc.yr = year(tc.Collecting_date)
endscan
```

```
top20yr = 0
top20yr.view ("Enter year (eg. 94) for Geographic analysis")

top20qry = query

:fidms:retnum.db  |retnum|yr              |
                  |_abcd |..~top20yr      |

:fidms:retex.db   |retnum|TaxonID|Genus |Species |Quantity |Unit
                     |Part                                |Grid square
|
                  |_abcd |Check  |Check |Check   |Calc Sum |Check not kg
                     |Check stems or blossom or sprays |Check not blank
|

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

tc.open (":priv:answer.db")
tc.edit()
scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus" and
   tc.species <> "scariosus") or
tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species <>
   "scabra") or
tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
   "angustifolia") or
tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
   "cucullata") or
tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
   "platysperma") or
tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
   "holoschoenus") or
tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
   "australis") or
tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species <>
   "drouynianus") or
tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
   "occidentale")) :
      tc."sum of quantity" = tc."sum of quantity"*10
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and tc.part = "blossom" and (tc.genus =
   "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*148
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and (tc.part = "sprays" or tc.part = "stems")
   and (tc.genus = "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*65
      tc.unit = "single"
endscan
```

```
scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
   tc.species = "scariosus"):
      tc."sum of quantity" = tc."sum of quantity"*50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
tc.species
   = "drouynianus"):
      tc."sum of quantity" = tc."sum of quantity" *20
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
tc.species
   = "occidentale"):
      tc."sum of quantity" = tc."sum of quantity" *7
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and tc.species
=
   "australis"):
      tc."sum of quantity" = tc."sum of quantity" *50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and tc.species
=
   "holoschoenus"):
      tc."sum of quantity" = tc."sum of quantity" *40
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and tc.species
=
   "angustifolium"):
      tc."sum of quantity" = tc."sum of quantity" *15
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and tc.species =
   "platysperma"):
      tc."sum of quantity" = tc."sum of quantity" *1
      tc.unit = "single"
endscan
tc.endedit()

top20qry = query

:priv:answer.db   |Grid square|Taxonid |
                  |Calc count |Check   |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif
```

```
top20qry = query

:priv:top20.db     |TaxonId      |Count of Grid Square |
                   |Check_taxid |Check as Grid_num    |

:priv:answer.db    |TaxonId    |Sum of Quantity  |
                   |_taxid     |Calc sum         |

:fidms:hbttaxon.db|TaxonId |Genus    |Species |
                   |_taxid  |Check    |Check   |

endquery

if not executeQBE(top20qry,":priv:top20int.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
   like (":priv:top20int.db")
   key "TaxonID"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20int.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertRecord()
tc.Year_of_Harvest= top20yr

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20compare = 0
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > 90 then

      top20qry = query

      :fidms:retnum.db  |retnum|yr                    |
                        |_abcd |..~top20compare       |

      :fidms:retex.db   |retnum|TaxonID|Genus|Species|Quantity|Unit
                           |Part                                |Grid square
|
                        |_abcd |Check  |Check|Check  |Calc sum|Check not
kg
                           |Check stems or blossom or
sprays|Checknotblank|
```

```
      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry) then
            errorshow()
            close()
      endif

      tc.open (":priv:answer.db")
      tc.edit()
      scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus"
and
         tc.species <> "scariosus") or
      tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species <>
         "scabra") or
      tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
         "angustifolia") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "cucullata") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "platysperma") or
      tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
         "holoschoenus") or
      tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
         "australis") or
      tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species
<>
         "drouynianus") or
      tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
         "occidentale")) :
      tc."sum of quantity" = tc."sum of quantity"*10
      tc.unit = "single"
endscan

      scan tc for tc.unit = "kg" and tc.part = "blossom" and (tc.genus =
         "Boronia" and tc.species = "megastigma"):
            tc."sum of quantity" = tc."sum of quantity"*148
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "kg" and (tc.part = "sprays" or
         tc.part = "stems")
         and (tc.genus = "Boronia" and tc.species = "megastigma");
            tc."sum of quantity" = tc."sum of quantity"*65
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
         tc.species = "scariosus"):
            tc."sum of quantity" = tc."sum of quantity"*50
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus"
         and tc.species = "drouynianus"):
            tc."sum of quantity" = tc."sum of quantity" *20
            tc.unit = "single"
      endscan
```

```
scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
   tc.species = "occidentale"):
      tc."sum of quantity" = tc."sum of quantity" *7
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and
   tc.species = "australis"):
      tc."sum of quantity" = tc."sum of quantity" *50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and
   tc.species = "holoschoenus"):
      tc."sum of quantity" = tc."sum of quantity" *40
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and
   tc.species = "angustifolium"):
      tc."sum of quantity" = tc."sum of quantity" *15
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and
   tc.species = "platysperma"):
      tc."sum of quantity" = tc."sum of quantity" *1
      tc.unit = "single"
endscan

tc.endedit()
top20qry = query

:priv:answer.db   |Grid square|Taxonid     |
                  |Calc count |Check       |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

top20qry = query

:priv:top20.db    |TaxonId      |Count of Grid Square |
                  |Check _taxid |Check as Grid_num    |

:priv:answer.db   |TaxonId   |Sum of Quantity  |
                  |_taxid    |Calc sum         |

:fidms:hbttaxon.db|TaxonId |Genus   |Species     |
                  |_taxid  |Check   |Check       |

endquery
if not executeQBE(top20qry,":priv:top20int.db") then
      errorshow()
endif
```

```
      tblcreate = create(":priv:top20com.db")
            like ":priv:top20int.db"
            key "TaxonID"
      endcreate

      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20int.db")
      srcTbl.add (destTbl)

      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate

      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest=top20compare

      top20rsl.open ("grid.rsl")
else
      msginfo ("Error", "Invalid year has been chosen")

endif
endMethod
```

No_of_species

This method calculates the number of species used in the commercial flower and seed trades in all years since 1991 and in each of the surveys that have been undertaken into the industry prior to this (uses table speccnt.db to hold results.

```
method No_of_species()

var
      destTbl     String
      srcTbl      Table
      tc          TCursor
      TblCreate   Table
      top20rsl    Report
      nospecqry   Query
      yeardate    Date
endVar

message ("Calculating number of wild harvested species")
sleep (5000)
srcTbl.attach(":priv:speccnt.db")
srcTbl.delete()

tc.open(":fidms:retnum.db")
tc.edit()
scan tc for isblank(tc.yr) :
      tc.yr = year(tc.Collecting_date)
endscan

nospecqry.readFromFile("getstems.qbe")
message ("Executing query, please wait")
sleep (3000)
```

```
if not executeQBE(nospecqry, ":priv:specstem.db") then
      errorshow()
      close()
endif

nospecqry.readFromFile("getseed.qbe")
message ("Executing query, please wait")
sleep (3000)
if not executeQBE(nospecqry) then
      errorshow()
      close()
endif

nospecqry = query

:priv:specstem.db  |yr          |stems |
                   |check _abcd|check |

:priv:answer.db    |yr              |seed  |
                   |_abcd           |check |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(nospecqry) then
      errorshow()
endif

tblcreate=create (":priv:speccnt.db")
      with "Yr" : "N", "Stems" : "N", "Seed" : "N"
      key "Yr"
endcreate

destTbl = ":priv:speccnt.db"
srcTbl.attach(":fidms:burghop.db")
if not srcTbl.add (destTbl) then
      errorshow()
endif

destTbl = ":priv:speccnt.db"
srcTbl.attach(":priv:answer.db")
if not srcTbl.add (destTbl) then
      errorshow()
endif
top20rsl.open (":fidms:speccnt.rsl")
endMethod
```

no_seed

This method calculates shows the total harvest of seed (in kg) for each species and the percentage of the total harvest that each species constitutes for any two user-defined years.

```
method no_seed()

var
      tc                TCursor
      tblcreate         Table
      srcTbl            Table
      destTbl           String
      top20rsl          Report
      top20qry          Query
      top20yr           String
      top20compare      String
      usrChoice         String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20yr = ""
top20yr.view ("Enter year (eg. 94) for top seed species")
if top20yr <> "" then

      top20qry = query

      :fidms:retnum.db  |retnum|Collecting_date |
                        |_abcd |..~top20yr      |

      :fidms:retex.db   |taxonId|retnum|Genus|Species|Quantity|Unit
                           |Part        | Crown_private  |
                        |Check  |_abcd |Check|Check  |Calc sum|Check kg
                           |Check seed | not A          |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry) then
            errorshow()
            close()
      endif

else
      return
endif

top20qry = query

:priv:answer.db   |Taxonid|Genus |Species          |Sum of Quantity|
                  |Check  |check |check not blank |calc sum       |
```

```
endquery
message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
      like ":priv:top20.db"
      key "sum of sum of quantity", "Taxonid"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertrecord()
tc."year_of_harvest" = top20yr

srcTbl.attach(":priv:top20srt.db")
sort srcTbl
      on "Sum of Sum of Quantity" D
endsort

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()

top20compare = ""
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > "90" then
      top20qry = query

      :fidms:retnum.db  |retnum|Collecting_date |
                        |_abcd |..~top20compare |

:fidms:retex.db   |taxonId|retnum|Genus |Species |Quantity |Unit
                     |Part        |Crown_private  |
                  |Check  |_abcd |Check |Check   |Calc sum |Check kg
                     | Check seed |Not A          |
      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20int.db") then
            errorshow()
            close()
      endif

      top20qry = query
```

```
:priv:top20int.db |Taxonid|Genus |Species         |Sum of Quantity|
                  |Check  |check |check not blank |calc sum       |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
      endif

      tblcreate=create (":priv:top20com.db")
            with "Taxonid" : "N", "Genus" : "A30", "Species" : "A30",
            "Sum of Sum of Quantity" : "N"
            key "Taxonid"
      endcreate
      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20.db")
      srcTbl.add (destTbl)
      srcTbl.attach(":priv:top20com.db")
      sort srcTbl on "Sum of Sum of Quantity" D
      endsort

      INDEX srcTbl
      MAINTAINED
      ON "Sum of Sum of Quantity"
      ENDINDEX
      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate
      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare
      top20rsl.open ("topsdcom.rsl")
else
      msginfo("Error", "Invalid year chosen - must be > 90")
endif
endMethod
```

no_stems

This method calculates shows the total harvest of single stem for each species and the percentage of the total harvest that each species constitutes for any two user-defined years.

```
method no_stems()

var
  tc                TCursor
  tblcreate                 Table
  srcTbl                          Table
  destTbl                   String
  top20rsl             Report
  top20qry                  Query
  top20yr                   String
  top20compare              String
  usrChoice                 String
```

```
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20yr = ""
top20yr.view ("Enter year (eg. 94) for top flower species")

top20qry = query

:fidms:retnum.db        |retnum|Collecting_date |
                                        |_abcd |..~top20yr      |

:fidms:retex.db         |taxonId    |retnum|Genus     |Species
|Quantity    |Unit    |Part
      |Crown_private|
                                        |Check      |_abcd |Check
|Check        |Calc sum    |Check   |Check stems or blossom or sprays
|not A        |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

tc.open (":priv:answer.db")
tc.edit()
scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus" and
tc.species <> "scariosus") or
      tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species <>
"scabra") or
      tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
"angustifolia") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
"cucullata") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
"platysperma") or
      tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
"holoschoenus") or
      tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
"australis") or
      tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species
<> "drouynianus") or
      tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
"occidentale")) :
      tc."sum of quantity" = tc."sum of quantity"*10
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and tc.part = "blossom" and (tc.genus =
"Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*148
```

```
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and (tc.part = "sprays" or tc.part = "stems")
and (tc.genus = "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*65
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
tc.species = "scariosus"):
      tc."sum of quantity" = tc."sum of quantity"*50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
tc.species = "drouynianus"):
      tc."sum of quantity" = tc."sum of quantity" *20
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
tc.species = "occidentale"):
      tc."sum of quantity" = tc."sum of quantity" *7
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and tc.species
= "australis"):
      tc."sum of quantity" = tc."sum of quantity" *50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and tc.species
= "holoschoenus"):
      tc."sum of quantity" = tc."sum of quantity" *40
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and tc.species
= "angustifolium"):
      tc."sum of quantity" = tc."sum of quantity" *15
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and tc.species =
"platysperma"):
      tc."sum of quantity" = tc."sum of quantity" *1
      tc.unit = "single"
endscan

tc.endedit()

top20qry = query

:priv:answer.db          |Taxonid     |Genus       |Species
      |Sum of Quantity  |
                                        |Check       |check       |check
not blank    |calc sum                 |
```

```
endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
      like ":priv:top20.db"
   key "sum of sum of quantity", "Taxonid"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertrecord()
tc."year_of_harvest" = top20yr

srcTbl.attach(":priv:top20srt.db")
sort srcTbl
  on "Sum of Sum of Quantity" D
endsort

srcTbl.unattach()

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()

top20compare = ""
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > "90" then

      top20qry = query

      :fidms:retnum.db        |retnum|Collecting_date |
                                                |_abcd |..~top20compare
|

      :fidms:retex.db         |taxonId    |retnum|Genus     |Species
|Quantity    |Unit    |Part
      |Crown_private|
                                                |Check      |_abcd
|Check     |Check         |Calc sum    |Check   |Check stems or blossom
or sprays|Not A        |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
```

```
      if not executeQBE(top20qry, ":priv:top20int.db") then
            errorshow()
            close()
      endif

      tc.open (":priv:top20int.db")
      tc.edit()
      scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus"
and tc.species <> "scariosus") or
            tc.unit = "bunches" and (tc.genus <> "Anarthria" and
tc.species <> "scabra") or
            tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species
<> "angustifolia") or
            tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species
<> "cucullata") or
            tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species
<> "platysperma") or
            tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species
<> "holoschoenus") or
            tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species
<> "australis") or
            tc.unit = "bunches" and (tc.genus <> "Podocarpus" and
tc.species <> "drouynianus") or
            tc.unit = "bunches" and (tc.genus <> "Xylomelum" and
tc.species <> "occidentale")) :
            tc."sum of quantity" = tc."sum of quantity"*10
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
tc.species = "scariosus"):
            tc."sum of quantity" = tc."sum of quantity"*50
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
tc.species = "drouynianus"):
            tc."sum of quantity" = tc."sum of quantity" *20
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
tc.species = "occidentale"):
            tc."sum of quantity" = tc."sum of quantity" *7
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and
tc.species = "australis"):
            tc."sum of quantity" = tc."sum of quantity" *50
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and
tc.species = "holoschoenus"):
            tc."sum of quantity" = tc."sum of quantity" *40
            tc.unit = "single"
      endscan
```

```
      scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and
tc.species = "angustifolium"):
            tc."sum of quantity" = tc."sum of quantity" *15
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and
tc.species = "platysperma"):
            tc."sum of quantity" = tc."sum of quantity" *1
            tc.unit = "single"
      endscan

      tc.endedit()

      top20qry = query

      :priv:top20int.db       |Taxonid    |Genus      |Species
|Sum of Quantity  |
                                                  |Check      |check
      |check not blank  |calc sum                 |
      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
      endif

      tblcreate=create (":priv:top20com.db")
            with "Taxonid" : "N", "Genus" : "A30", "Species" : "A30",
            "Sum of Sum of Quantity" : "N"
      key "Taxonid"
      endcreate

      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20.db")
      srcTbl.add (destTbl)

      srcTbl.attach(":priv:top20com.db")
      sort srcTbl on "Sum of Sum of Quantity" D
      endsort

      INDEX srcTbl
            MAINTAINED
            ON "Sum of Sum of Quantity"
      ENDINDEX

      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate
      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare
      top20rsl.open ("top20com.rsl")
      else
            usrChoice = msgquestion ("Do you wish to view",
               "Records for 1980/81 only available")
      Switch
            case usrChoice = "Yes":
```

```
            top20rsl.open("top20stm.rsl")

            case usrChoice = "No" :
            return
        endSwitch

endif

endMethod
```

GEOLIB.LSL Library

GEOLIB contains custom methods for generating complex reports and statistics for the user.

ANCA

This method produces a summary of the amount of harvest amounts for all species, products and land tenures for a user-defined year.

```
method ANCA()
var
      tc                TCursor
      tblcreate         Table
      srcTbl            Table
      destTbl           String
      top20rsl          Report
      top20qry          Query
      top20yr           String
      top20compare      String
      usrChoice         String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20yr = ""
top20yr.view ("Enter year (eg. 94) for ANCA report")

top20qry = query
:fidms:retnum.db  |retnum|Collecting_date |
                  |_abcd |..~top20yr      |

:fidms:retex.db   |taxonId     |retnum |Genus |Species  |Quantity |Unit
                     |Part   |Crown_private|
                  |Check _abcd |Check  |Check |Calc sum |Check    |Check
                     |Check  |Check        |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

tc.open (":priv:answer.db")
tc.edit()
scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus" and
         tc.species <> "scariosus") or
      tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species <>
```

```
        "scabra") or
      tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
        "angustifolia") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
        "cucullata") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
        "platysperma") or
      tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
        "holoschoenus") or
      tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
        "australis") or
      tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species
<>
        "drouynianus") or
      tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
        "occidentale")) : tc."sum of quantity" = tc."sum of
quantity"*10
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and tc.part = "blossom" and
   (tc.genus = "Boronia" and tc.species = "megastigma"):
   tc."sum of quantity" = tc."sum of quantity"*148
   tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and (tc.part = "sprays" or tc.part = "stems")
   and (tc.genus = "Boronia" and tc.species = "megastigma"):
   tc."sum of quantity" = tc."sum of quantity"*65
   tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
   tc.species = "scariosus"):
   tc."sum of quantity" = tc."sum of quantity"*50
   tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
   tc.species = "drouynianus"):
   tc."sum of quantity" = tc."sum of quantity" *20
   tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
   tc.species = "occidentale"):
   tc."sum of quantity" = tc."sum of quantity" *7
   tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and
   tc.species = "australis"):
   tc."sum of quantity" = tc."sum of quantity" *50
   tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and tc.species
=
   "holoschoenus"):
   tc."sum of quantity" = tc."sum of quantity" *40
```

```
   tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and tc.species
=
   "angustifolium"):
   tc."sum of quantity" = tc."sum of quantity" *15
   tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and tc.species =
   "platysperma"):
tc."sum of quantity" = tc."sum of quantity" *1
tc.unit = "single"
endscan

tc.endedit()
top20qry = query

:priv:answer.db  |Taxonid|Genus |Species        |Sum of Quantity |Unit
                    |Part |Crown_private |
                 |Check  |check |check not blank|calc sum        |Check
                    |Check|Check         |
endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
      like ":priv:top20.db"
      key "part", "unit", "sum of sum of quantity", "Taxonid"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate
tc.open (":priv:top20yr.db")
tc.edit()
tc.insertrecord()
tc."year_of_harvest" = top20yr

srcTbl.attach(":priv:top20srt.db")
sort srcTbl
      on "part", "unit", "Sum of Sum of Quantity" D
endsort
srcTbl.unattach()
tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
top20rsl.open ("ANCA.rsl")
endMethod
```

cpa_seed

This method produces a summary of the number and percentage of seed picked from Crown land, and private property in any two user-defined years.

method cpa_seed()

```
var
      ansTbl                  Table
      tc                      TCursor
      tblcreate               Table
      srcTbl                  Table
      destTbl                 String
      top20rsl                Report
      top20qry                Query
      top20yr                 Smallint
      top20compare            Smallint
      usrChoice               String
      fullMonth Array[12]     String
endVar

fullMonth[1] =  "January"
fullMonth[2] =  "February"
fullMonth[3] =  "March"
fullMonth[4] =  "April"
fullMonth[5] =  "May"
fullMonth[6] =  "June"
fullMonth[7] =  "July"
fullMonth[8] =  "August"
fullMonth[9] =  "September"
fullMonth[10] = "October"
fullMonth[11] = "November"
fullMonth[12] = "December"

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

tc.open(":fidms:retnum.db")
tc.edit()
scan tc for isblank(tc.month) :
      tc.month = fullmonth[month(tc.Collecting_date)]
      tc.mth_num = month(tc.collecting_date)
      tc.yr = year(tc.Collecting_date)
endscan

tc.endedit()

top20yr = 0
top20yr.view ("Enter year (eg. 94) for Crown/Private calculation")

top20qry = query

:fidms:retnum.db  |mth_num    |retnum|yr              |Month    |
```

```
                    |check      |_abcd |..~top20yr      |Check    |

:fidms:retex.db     |retnum|Quantity  |Unit     |Part       |Crown_private
|
                    |_abcd |Calc sum  |Check kg |Check seed|Check not
blank|

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
      close()
endif

top20qry.readfromfile("cpaseed.qbe")
if not executeQBE(top20qry,":priv:top20int.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
      with "Mth_num" : "S", "Month" : "A12", "Cqty": "N" ,
      "Pqty":  "N"
      key "Mth_num"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20int.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertRecord()
tc.Year_of_Harvest=top20yr

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20compare = 0
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > 90 then

      top20qry = query

:fidms:retnum.db    |mth_num |retnum|yr              |Month    |
                    |check    |_abcd |..~top20compare |Check    |

:fidms:retex.db     |retnum|Quantity |Unit     |Part       |Crown_private
|
```

```
                  |_abcd |Calc sum |Check kg |Check seed |Check not
blank |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
            close()
      endif

      top20qry.readfromfile("cpaseed.qbe")
      if not executeQBE(top20qry,":priv:top20int.db") then
            errorshow()
      endif

      tblcreate=create (":priv:top20com.db")
            with "Mth_num" : "S", "Month" : "A12", "Cqty": "N" ,
            "Pqty":  "N"
      key "Mth_num"
      endcreate

      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20int.db")
      srcTbl.add (destTbl, true, true)

      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate

      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare

      top20rsl.open ("cpaseed.rsl")
else
      msginfo ("Sorry", "The year you have chosen is not valid")
endif
endMethod
```

cpa_stems

```
method cpa_stems()

var
      tc                      TCursor
      tblcreate               Table
      srcTbl                  Table
      destTbl                 String
      top20rsl                Report
      top20qry                Query
      top20yr                 Smallint
      top20compare            Smallint
      usrChoice               String
      fullMonth Array[12]     String
```

```
endVar

fullMonth[1] =  "January"
fullMonth[2] =  "February"
fullMonth[3] =  "March"
fullMonth[4] =  "April"
fullMonth[5] =  "May"
fullMonth[6] =  "June"
fullMonth[7] =  "July"
fullMonth[8] =  "August"
fullMonth[9] =  "September"
fullMonth[10] = "October"
fullMonth[11] = "November"
fullMonth[12] = "December"

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

tc.open(":fidms:retnum.db")
tc.edit()
scan tc for isblank(tc.month) :
      tc.month = fullmonth[month(tc.Collecting_date)]
      tc.mth_num = month(tc.collecting_date)
      tc.yr = year(tc.Collecting_date)
endscan
tc.endedit()

top20yr = 0
top20yr.view ("Enter year (eg. 94) for Crown/Private calculation")

top20qry = query

:fidms:retnum.db  |mth_num|retnum|yr            |Month    |
                  |check  |_abcd |..~top20yr    |Check    |

:fidms:retex.db   |retnum|Genus |Species |Quantity |Unit
                     |Part                              |Crown_private
|
                  |_abcd |Check |Check   |Calc sum |Check
                     |Check stems or blossom or sprays|Check not blank
|

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

tc.open (":priv:answer.db")
tc.edit()
scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus" and
tc.species <> "scariosus") or
      tc.unit = "bunches" and (tc.genus <> "Anarthria"
```

```
         and tc.species <> "scabra") or
      tc.unit = "bunches" and (tc.genus <> "Crowea" and
         tc.species <> "angustifolia") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "cucullata") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "platysperma") or
      tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
         "holoschoenus") or
      tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
         "australis") or
      tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species
<>
         "drouynianus") or
      tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
         "occidentale")) :
      tc."sum of quantity" = tc."sum of quantity"*10
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and tc.part = "blossom" and
(tc.genus = "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*148
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and (tc.part = "sprays" or tc.part = "stems")
   and (tc.genus = "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*65
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
   tc.species = "scariosus"):
      tc."sum of quantity" = tc."sum of quantity"*50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
   tc.species = "drouynianus"):
      tc."sum of quantity" = tc."sum of quantity" *20
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
   tc.species = "occidentale"):
      tc."sum of quantity" = tc."sum of quantity" *7
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and
   tc.species = "australis"):
      tc."sum of quantity" = tc."sum of quantity" *50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and
   tc.species = "holoschoenus"):
      tc."sum of quantity" = tc."sum of quantity" *40
      tc.unit = "single"
```

```
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and tc.species
=
   "angustifolium"):
      tc."sum of quantity" = tc."sum of quantity" *15
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and tc.species =
   "platysperma"):
      tc."sum of quantity" = tc."sum of quantity" *1
      tc.unit = "single"
endscan
tc.endedit()

top20qry = query

:priv:answer.db    |mth_num|Sum of Quantity |Month     |Crown_private|
                   |check  |calc sum        |Check     |Check        |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

top20qry.readfromfile("cpa.qbe")
if not executeQBE(top20qry,":priv:top20int.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
      with "Mth_num" : "S", "Month" : "A12", "Cqty": "N" ,
      "Pqty":  "N" , "Aqty":  "N"
      key "Mth_num"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20int.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertRecord()
tc.Year_of_Harvest= top20yr

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()
```

```
top20compare = 0
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > 90 then
      top20qry = query

:fidms:retnum.db  |Mth_num|retnum|Yr              |Month   |
                  |Check  |_abcd |..~top20compare |Check   |

:fidms:retex.db   |retnum|Genus |Species |Quantity |Unit |
                     Part                         |Crown_private|
                  |_abcd |Check |Check   |Calc sum |Check|
                     Check stems or blossom or sprays|Check        |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry) then
            errorshow()
            close()
      endif

      tc.open (":priv:answer.db")
      tc.edit()
      scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus"
and
         tc.species <> "scariosus") or
            tc.unit = "bunches" and (tc.genus <> "Anarthria" and
               tc.species <> "scabra") or
            tc.unit = "bunches" and (tc.genus <> "Crowea" and
               tc.species <> "angustifolia") or
            tc.unit = "bunches" and (tc.genus <> "Hakea" and
               tc.species <> "cucullata") or
            tc.unit = "bunches" and (tc.genus <> "Hakea" and
               tc.species <> "platysperma") or
            tc.unit = "bunches" and (tc.genus <> "Juncus" and
               tc.species <> "holoschoenus") or
            tc.unit = "bunches" and (tc.genus <> "Kingia" and
               tc.species <> "australis") or
            tc.unit = "bunches" and (tc.genus <> "Podocarpus" and
               tc.species <> "drouynianus") or
            tc.unit = "bunches" and (tc.genus <> "Xylomelum" and
               tc.species <> "occidentale")) :
            tc."sum of quantity" = tc."sum of quantity"*10
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
         tc.species = "scariosus"):
            tc."sum of quantity" = tc."sum of quantity"*50
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
         tc.species = "drouynianus"):
            tc."sum of quantity" = tc."sum of quantity" *20
            tc.unit = "single"
      endscan
```

```
      scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
         tc.species = "occidentale"):
            tc."sum of quantity" = tc."sum of quantity" *7
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and
         tc.species = "australis"):
            tc."sum of quantity" = tc."sum of quantity" *50
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and
         tc.species = "holoschoenus"):
            tc."sum of quantity" = tc."sum of quantity" *40
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and
         tc.species = "angustifolium"):
            tc."sum of quantity" = tc."sum of quantity" *15
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and
         tc.species = "platysperma"):
            tc."sum of quantity" = tc."sum of quantity" *1
            tc.unit = "single"
      endscan
      tc.endedit()

      top20qry = query

:priv:answer.db   |Mth_num    |Sum of Quantity  |Month  |Crown_private |
                  |check      |calc sum         |Check  |Check         |
      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
      endif

      top20qry.readfromfile("cpa.qbe")
      if not executeQBE(top20qry,":priv:top20int.db") then
            errorshow()
      endif

      tblcreate=create (":priv:top20com.db")
            with "Mth_num" : "S", "Month" : "A12", "Cqty": "N" ,
            "Pqty":  "N" , "Aqty":  "N"
            key "Mth_num"
      endcreate

      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20int.db")
      srcTbl.add (destTbl)

      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
```

```
      endcreate

      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare

      top20rsl.open ("cpa_comp.rsl")
else
      usrChoice = msgquestion ("Do you wish to view",
         "Records for 1980/81 only available")
      Switch
            case usrChoice = "Yes":
            top20rsl.open("cpa_hop.rsl")
            case usrChoice = "No" :
            return
      endSwitch
endif
endMethod
```

flreg_seed

This method calculates the amount of seed (in kg) harvested from each of the CALM Picking Regions.

```
method flreg_seed()
var
      tc                TCursor
      tblcreate         table
      srcTbl            Table
      destTbl           String
      top20rsl          Report
      top20qry          Query
      top20yr           Smallint
      top20compare      Smallint
      usrChoice         String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20yr = 0
top20yr.view ("Enter year (eg. 94) for Geographic analysis")

top20qry = query

:fidms:retnum.db  |retnum|Collecting_date|
                  |_abcd |..~top20yr     |

:fidms:retex.db   |retnum|TaxonID|Genus |Species |Quantity  |Unit
                     |Part        |Grid square |
                  |_abcd |Check  |Check |Check   |Calc sum  |Check kg
                     |Check seed  |_grid       |
```

```
:fidms:geograph.db |Grid Square|Picking Region  |
                   |_grid      |Check           |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

top20qry = query

:priv:answer.db   |Picking Region|Sum of Quantity  |
                  |check         |calc sum         |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
      with "Picking Region" : "A30", "Qty": "N"
   key "Picking Region"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertRecord()
tc.Year_of_Harvest= top20yr

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20compare = 0
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > 90 then

      top20qry = query

      :fidms:retnum.db  |retnum|Collecting_date  |
                        |_abcd |..~top20compare  |
```

```
      :fidms:retex.db   |retnum|TaxonID|Genus |Species |Quantity |Unit
                           | Part        |Grid square |
                        |_abcd |Check  |Check |Check   |Calc sum |Check
kg
                           | Check seed   |_grid      |

      :fidms:geograph.db      |Grid Square|Picking Region   |
                              |_grid      |Check            |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry) then
            errorshow()
            close()
      endif

      tc.endedit()

      top20qry = query

      :priv:answer.db         |Picking Region|Sum of Quantity     |
                              |check         |calc sum            |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
      endif

      tblcreate=create (":priv:top20com.db")
            with "Picking Region" : "A30", "Qty": "N"
      key "Picking Region"
      endcreate

      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20.db")
      srcTbl.add (destTbl)

      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate

      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare
      top20rsl.open ("pickseed.rsl")
else
      msginfo("Sorry", "The year you have chosen is not valid")
endif
endMethod
```

flreg_stems

This method calculates the amount of seed (in kg) harvested from each of the CALM Picking Regions.

```
method flreg_stems()
var
      tc                TCursor
      tblcreate         Table
      srcTbl            Table
      destTbl           String
      top20rsl          Report
      top20qry          Query
      top20yr           Smallint
      top20compare      Smallint
      usrChoice         String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20yr = 0
top20yr.view ("Enter year (eg. 94) for Geographic analysis")

top20qry = query

:fidms:retnum.db  |retnum|Collecting_date|
                  |_abcd |..~top20yr     |

:fidms:retex.db   |retnum|TaxonID|Genus |Species |Quantity |Unit
                     |Part                        |Grid square      |
                  |_abcd |Check  |Check |Check   |Calc sum  |Check|Check
                     stems or blossom or sprays |_grid            |

:fidms:geograph.db|Grid Square|Picking Region   |
                  |_grid      |Check            |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

tc.open (":priv:answer.db")
tc.edit()
scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus" and
   tc.species <> "scariosus") or
tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species <>
   "scabra") or
tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
```

```
   "angustifolia") or
tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
   "cucullata") or
tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
   "platysperma") or
tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
   "holoschoenus") or
tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
   "australis") or
tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species <>
   "drouynianus") or
tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
   "occidentale")) :
      tc."sum of quantity" = tc."sum of quantity"*10
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and tc.part = "blossom" and
   (tc.genus = "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*148
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and (tc.part = "sprays" or tc.part = "stems")
   and (tc.genus = "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*65
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
   tc.species = "scariosus"):
      tc."sum of quantity" = tc."sum of quantity"*50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
tc.species
   = "drouynianus"):
      tc."sum of quantity" = tc."sum of quantity" *20
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
tc.species
   = "occidentale"):
      tc."sum of quantity" = tc."sum of quantity" *7
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and tc.species
=
   "australis"):
      tc."sum of quantity" = tc."sum of quantity" *50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and tc.species
=
   "holoschoenus"):
      tc."sum of quantity" = tc."sum of quantity" *40
```

```
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and tc.species
=
   "angustifolium"):
      tc."sum of quantity" = tc."sum of quantity" *15
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and tc.species =
   "platysperma"):
      tc."sum of quantity" = tc."sum of quantity" *1
      tc.unit = "single"
endscan

tc.endedit()

top20qry = query

:priv:answer.db          |Picking Region|Sum of Quantity      |
                         |check         |calc sum             |

endquery
message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
      with "Picking Region" : "A30", "Qty": "N"
   key "Picking Region"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertRecord()
tc.Year_of_Harvest= top20yr

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20compare = 0
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > 90 then
```

```
      top20qry = query

:fidms:retnum.db  |retnum|Collecting_date       |
                  |_abcd |..~top20compare       |

:fidms:retex.db   |retnum|TaxonID|Genus |Species |Quantity |Unit  |
                     Part                       |Grid square      |
                  |_abcd |Check  |Check |Check   |Calc sum |Check |Check
                     stems or blossom or sprays |_grid            |

:fidms:geograph.db|Grid Square|Picking Region   |
                  |_grid      |Check            |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry) then
            errorshow()
            close()
      endif

      tc.open (":priv:answer.db")
      tc.edit()
      scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus"
and
         tc.species <> "scariosus") or
      tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species <>
         "scabra") or
      tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
         "angustifolia") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "cucullata") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "platysperma") or
      tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
         "holoschoenus") or
      tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
         "australis") or
      tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species
<>
         "drouynianus") or
      tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
         "occidentale")) :
            tc."sum of quantity" = tc."sum of quantity"*10
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
         tc.species = "scariosus"):
            tc."sum of quantity" = tc."sum of quantity"*50
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
         tc.species = "drouynianus"):
            tc."sum of quantity" = tc."sum of quantity" *20
            tc.unit = "single"
      endscan
```

```
      scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
         tc.species = "occidentale"):
            tc."sum of quantity" = tc."sum of quantity" *7
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and
         tc.species = "australis"):
            tc."sum of quantity" = tc."sum of quantity" *50
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and
         tc.species = "holoschoenus"):
            tc."sum of quantity" = tc."sum of quantity" *40
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and
         tc.species = "angustifolium"):
            tc."sum of quantity" = tc."sum of quantity" *15
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and
         tc.species = "platysperma"):
            tc."sum of quantity" = tc."sum of quantity" *1
            tc.unit = "single"
      endscan
      tc.endedit()

      top20qry = query

      :priv:answer.db   |Picking Region|Sum of Quantity     |
                        |check         |calc sum            |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
      endif

      tblcreate=create (":priv:top20com.db")
            with "Picking Region" : "A30", "Qty": "N"
      key "Picking Region"
      endcreate
      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20.db")
      srcTbl.add (destTbl)
      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate
            tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare
      top20rsl.open ("pick_reg.rsl")
else
```

```
      msginfo("Sorry", "The year you have chosen is not valid")
endif
endMethod
```

g_seed

This method calculates the total harvest of seed (in kg) for each genus and the total number of species which are commercially exploited within each genus for any two user-defined years.

```
method g_seed()

var
      tc                TCursor
      tblcreate         Table
      srcTbl            Table
      destTbl           String
      top20rsl          Report
      top20qry          Query
      top20yr           String
      top20compare      String
      usrChoice         String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20yr = ""
top20yr.view ("Enter year (eg. 94) for top seed genera")

top20qry = query

:fidms:retnum.db  |retnum|Collecting_date |
                  |_abcd |..~top20yr      |

:fidms:retex.db   |taxonId|retnum|Genus |Species          |Quantity
                     |Unit    | Part        |Crown_private    |
                  |Check  |_abcd |Check |Check not blank |Calc sum
                     |Check kg|Check seed   |Not A           |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

top20qry = query

:priv:answer.db   |Genus |Species   |Sum of Quantity  |
                  |check |calc count|calc sum         |

endquery

message ("Executing query, please wait")
sleep (3000)
```

```
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

tblcreate=create (":priv:top20srt.db")
      with "Genus" : "A30", "No_Species" : "N",
      "Sum of Sum of Quantity" : "N"
      key "sum of sum of quantity", "Genus"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertrecord()
tc."year_of_harvest" = top20yr

srcTbl.attach(":priv:top20srt.db")
sort srcTbl
      on "Sum of Sum of Quantity" D
endsort

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()

top20compare = ""
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > "90" then

      top20qry = query

:fidms:retnum.db  |retnum|Collecting_date |
                  |_abcd |..~top20compare |

:fidms:retex.db   |taxonId|retnum|Genus |Species |Quantity |Unit
                     |Part       |Crown_private  |
                  |Check  |_abcd |Check |Check   |Calc sum |Check kg
                     |Check seed |Not A          |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20int.db") then
            errorshow()
            close()
      endif

      top20qry = query

      :priv:top20int.db |Genus |Species    |Sum of Quantity |
```

```
                        |check |calc count |calc sum          |
      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
      endif

      tblcreate=create (":priv:top20com.db")
            with "Genus" : "A30","No_Species" : "N",
            "Sum of Sum of Quantity" : "N"
            key "Genus"
      endcreate

      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20.db")
      srcTbl.add (destTbl)

      srcTbl.attach(":priv:top20com.db")
      sort srcTbl on "Sum of Sum of Quantity" D
      endsort

      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate

      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare
      top20rsl.open ("sdcogen.rsl")
else
      msgstop ("Error", "You have chosen an invalid year")
endif
endMethod
```

g_stems

This method calculates the total harvest of seed (in kg) for each genus and the total number of species which are commercially exploited within each genus for any two user-defined years.

```
method g_stems()
var
      tc                TCursor
      tblcreate         Table
      srcTbl            Table
      destTbl           String
      top20rsl          Report
      top20qry          Query
      top20yr           String
      top20compare      String
      usrChoice         String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
```

```
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20yr = ""
top20yr.view ("Enter year (eg. 94) for top flower genera")

top20qry = query

:fidms:retnum.db  |retnum|Collecting_date |
                  |_abcd |..~top20yr      |

:fidms:retex.db   |taxonId |retnum|Genus |Species          |Quantity
|Unit
                     |Part                                |Crown_private |
                  |Check   |_abcd |Check |Check not blank |Calc sum
|Check
                     |Check stems or blossom or sprays |Not A         |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

tc.open (":priv:answer.db")
tc.edit()
scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus" and
   tc.species <> "scariosus") or
tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species <>
   "scabra") or
tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
   "angustifolia") or
tc.unit = "bunches" and (tc.genus<>"Hakea" and tc.species <>
"cucullata")or
tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
   "platysperma") or
tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
   "holoschoenus") or
tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
   "australis") or
tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species <>
   "drouynianus") or
tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
   "occidentale")) :
      tc."sum of quantity" = tc."sum of quantity"*10
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and tc.part = "blossom" and (tc.genus =
   "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*148
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and (tc.part = "sprays" or tc.part = "stems")
```

```
   and (tc.genus = "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*65
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
   tc.species = "scariosus"):
      tc."sum of quantity" = tc."sum of quantity"*50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
tc.species
   = "drouynianus"):
      tc."sum of quantity" = tc."sum of quantity" *20
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
tc.species
   = "occidentale"):
      tc."sum of quantity" = tc."sum of quantity" *7
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and tc.species
=
   "australis"):
      tc."sum of quantity" = tc."sum of quantity" *50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and tc.species
=
   "holoschoenus"):
      tc."sum of quantity" = tc."sum of quantity" *40
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and tc.species
=
   "angustifolium"):
      tc."sum of quantity" = tc."sum of quantity" *15
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and tc.species =
   "platysperma"):
      tc."sum of quantity" = tc."sum of quantity" *1
      tc.unit = "single"
endscan
tc.endedit()

top20qry = query

:priv:answer.db   |Genus |Species   |Sum of Quantity  |
                  |check |calc count|calc sum         |

endquery
```

```
message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

tblcreate=create (":priv:top20srt.db")
      with "Genus" : "A30", "No_Species" : "N",
      "Sum of Sum of Quantity" : "N"
      key "Sum of sum of quantity", "Genus"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertrecord()
tc."year_of_harvest" = top20yr

srcTbl.attach(":priv:top20srt.db")
sort srcTbl
      on "Sum of Sum of Quantity" D
endsort

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()

top20compare = ""
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > "90" then

      top20qry = query

:fidms:retnum.db  |retnum|Collecting_date |
                  |_abcd |..~top20compare |

:fidms:retex.db   |taxonId|retnum|Genus |Species |Quantity |Unit
                     |Part                          |Crown_private  |
                  |Check  |_abcd |Check |Check   |Calc sum |Check
                     |stems or blossom or sprays  |Not A          |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20int.db") then
            errorshow()
            close()
      endif

      tc.open (":priv:top20int.db")
```

```
      tc.edit()
      scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus"
         and tc.species <> "scariosus") or
      tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species <>
         "scabra") or
      tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
         "angustifolia") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "cucullata") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "platysperma") or
      tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
         "holoschoenus") or
      tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
         "australis") or
      tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species
<>
         "drouynianus") or
      tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
         "occidentale")) :
            tc."sum of quantity" = tc."sum of quantity"*10
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "kg" and tc.part = "blossom" and (tc.genus =
         "Boronia" and tc.species = "megastigma"):
            tc."sum of quantity" = tc."sum of quantity"*148
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "kg" and (tc.part = "sprays" or tc.part =
         "stems") and (tc.genus = "Boronia" and tc.species =
"megastigma"):
            tc."sum of quantity" = tc."sum of quantity"*65
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
         tc.species = "scariosus"):
            tc."sum of quantity" = tc."sum of quantity"*50
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
         tc.species = "drouynianus"):
            tc."sum of quantity" = tc."sum of quantity" *20
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
         tc.species = "occidentale"):
            tc."sum of quantity" = tc."sum of quantity" *7
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and
         tc.species = "australis"):
            tc."sum of quantity" = tc."sum of quantity" *50
            tc.unit = "single"
      endscan
```

```
      scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and
         tc.species = "holoschoenus"):
            tc."sum of quantity" = tc."sum of quantity" *40
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and
         tc.species = "angustifolium"):
            tc."sum of quantity" = tc."sum of quantity" *15
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and
         tc.species = "platysperma"):
            tc."sum of quantity" = tc."sum of quantity" *1
            tc.unit = "single"
      endscan
      tc.endedit()

      top20qry = query

      :priv:top20int.db |Genus      |Species    |Sum of Quantity  |
                        |check      |calc count |calc sum         |
      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
      endif

      tblcreate=create (":priv:top20com.db")
            with "Genus" : "A30", "No_Species" : "N",
            "Sum of Sum of Quantity" : "N"
            key "Genus"
      endcreate

      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20.db")
      srcTbl.add (destTbl)

      srcTbl.attach(":priv:top20com.db")
      sort srcTbl on "Sum of Sum of Quantity" D
      endsort

      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate

      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare

      top20rsl.open ("topcogen.rsl")
else
      usrChoice = msgquestion ("Do you wish to view",
         "Records for 1980/81 only available")
      Switch
```

```
            case usrChoice = "Yes":
                  top20rsl.open("topstgen.rsl")
                  case usrChoice = "No" :
            return
      endSwitch
endif
endMethod
```

geo_seed

This method calculates the number of species and the percentage of the total of seed (in kg) harvested from each grid square in two user-defined years.

```
method geo_seed()
var
      tc                        TCursor
      tblcreate                 Table
      srcTbl                    Table
      destTbl                   String
      top20rsl                  Report
      top20qry                  Query
      top20yr                   Smallint
      top20compare              Smallint
      usrChoice                 String
      fullMonth Array[12]       String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20yr = 0
top20yr.view ("Enter year (eg. 94) for Geographic analysis")

top20qry = query

:fidms:retnum.db  |retnum|Collecting_date |
                  |_abcd |..~top20yr      |

:fidms:retex.db   |retnum|TaxonID|Genus |Species |Quantity |Unit     |
                     Part           |Grid square       |
                  |_abcd |Check  |Check |Check   |calc sum |check kg|
                     Check seed     |Check not blank   |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

top20qry = query
```

```
:priv:answer.db   |Grid square|Sum of Quantity  |Taxonid          |
                  |check      |calc sum         |Check            |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

top20qry.readfromfile("geo.qbe")
if not executeQBE(top20qry,":priv:top20int.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
      with "Grid_num" : "S", "No_species" : "S", "Qty": "N"
      key "Grid_num"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20int.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertRecord()
tc.Year_of_Harvest= top20yr

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20compare = 0
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > 90 then

      top20qry = query

      :fidms:retnum.db  |retnum|yr                    |
                        |_abcd |..~top20compare       |

      :fidms:retex.db   |retnum|TaxonID|Genus |Species |Quantity |
                           Unit    |Part        |Grid square      |
                        |_abcd |Check  |Check |Check   |Calc sum |
                           Check kg|Check seed  |Check not blank  |

      endquery

      message ("Executing query, please wait")
```

```
      sleep (3000)
      if not executeQBE(top20qry) then
            errorshow()
            close()
      endif

      top20qry = query

      :priv:answer.db          |Grid square|Sum of Quantity  |Taxonid  |
                               |check      |calc sum         | Check   |

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry, ":priv:top20.db") then
            errorshow()
      endif

      top20qry.readfromfile("geocom.qbe")
      if not executeQBE(top20qry,":priv:top20int.db") then
            errorshow()
      endif

      tblcreate=create (":priv:top20com.db")
            with "Grid_num" : "S", "No_species" : "S", "Qty": "N"
      key "Grid_num"
      endcreate

      destTbl = ":priv:top20com.db"
      srcTbl.attach(":priv:top20int.db")
      srcTbl.add (destTbl)

      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate

      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare

      top20rsl.open ("geoseed.rsl")
else
      msginfo("Sorry", "Data not available for this year")
endif
endMethod
```

geo_stems

This method calculatesthe number of species and the percentage of the total of single stems harvested from each grid square in two user-defined years.

```
method geo_stems()

var
      tc                        TCursor
      tblcreate                 Table
      srcTbl                    Table
      destTbl                   String
      top20rsl                  Report
      top20qry                  Query
      top20yr                   Smallint
      top20compare              Smallint
      usrChoice                 String
      fullMonth Array[12]       String
endVar

tblcreate.attach(":priv:top20srt.db")
tblcreate.empty()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()

top20yr = 0
top20yr.view ("Enter year (eg. 94) for Geographic analysis")

top20qry = query

:fidms:retnum.db  |retnum|Collecting_date  |
                  |_abcd |..~top20yr       |

:fidms:retex.db   |retnum|TaxonID|Genus |Species |Quantity |Unit
                     |Part                                |Grid square
|
                  |_abcd |Check  |Check |Check   |Calc sum |Check
                     |Check stems or blossom or sprays |Check not blank
|

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry) then
      errorshow()
      close()
endif

tc.open (":priv:answer.db")
tc.edit()
scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus" and
    tc.species <> "scariosus") or
tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species <>
    "scabra") or
```

```
tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
   "angustifolia") or
tc.unit = "bunches" and (tc.genus <> "Hakea" and
   tc.species <> "cucullata") or
tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
   "platysperma") or
tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
   "holoschoenus") or
tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
   "australis") or
tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species <>
   "drouynianus") or
tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
   "occidentale")) :
      tc."sum of quantity" = tc."sum of quantity"*10
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and tc.part = "blossom" and (tc.genus =
   "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*148
      tc.unit = "single"
endscan

scan tc for tc.unit = "kg" and (tc.part = "sprays" or tc.part = "stems")
   and (tc.genus = "Boronia" and tc.species = "megastigma"):
      tc."sum of quantity" = tc."sum of quantity"*65
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
   tc.species = "scariosus"):
      tc."sum of quantity" = tc."sum of quantity"*50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
tc.species
   = "drouynianus"):
      tc."sum of quantity" = tc."sum of quantity" *20
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
tc.species
   = "occidentale"):
      tc."sum of quantity" = tc."sum of quantity" *7
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and tc.species
=
   "australis"):
      tc."sum of quantity" = tc."sum of quantity" *50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and tc.species
=
   "holoschoenus"):
```

```
      tc."sum of quantity" = tc."sum of quantity" *40
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and tc.species
=
   "angustifolium"):
      tc."sum of quantity" = tc."sum of quantity" *15
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and tc.species =
   "platysperma"):
      tc."sum of quantity" = tc."sum of quantity" *1
      tc.unit = "single"
endscan
tc.endedit()

top20qry = query

:priv:answer.db   |Grid square|Sum of Quantity  |Taxonid          |
                  |check       |calc sum         |Check            |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

top20qry.readfromfile("geo.qbe")
if not executeQBE(top20qry,":priv:top20int.db") then
      errorshow()
endif

tblcreate = create (":priv:top20srt.db")
      with "Grid_num" : "S", "No_species" : "S", "Qty": "N"
      key "Grid_num"
endcreate

destTbl = ":priv:top20srt.db"
srcTbl.attach(":priv:top20int.db")
srcTbl.add (destTbl)

tblcreate = create (":priv:top20yr.db")
      with "Year_of_Harvest": "A4"
endcreate

tc.open (":priv:top20yr.db")
tc.edit()
tc.insertRecord()
tc.Year_of_Harvest= top20yr

tblcreate.attach(":priv:answer.db")
tblcreate.delete()
tblcreate.attach(":priv:top20.db")
tblcreate.delete()
tblcreate.attach(":priv:top20int.db")
tblcreate.delete()
```

```
top20compare = 0
top20compare.view ("Enter year (eg. 94) to compare with")

if top20compare > 90 then

      top20qry = query

      :fidms:retnum.db  |retnum|Collecting_date       |
                        |_abcd |..~top20compare       |

:fidms:retex.db   |retnum|TaxonID|Genus |Species |Quantity |Unit
                     |Part                                |Grid square    |
                  |_abcd |Check  |Check |Check   |Calc sum |Check
                     |Check stems or blossom or sprays|Check not blank|

      endquery

      message ("Executing query, please wait")
      sleep (3000)
      if not executeQBE(top20qry) then
            errorshow()
            close()
      endif

      tc.open (":priv:answer.db")
      tc.edit()
      scan tc for (tc.unit = "bunches" and (tc.genus <> "Leptocarpus"
and
         tc.species <> "scariosus") or
      tc.unit = "bunches" and (tc.genus <> "Anarthria" and tc.species
         <> "scabra") or
      tc.unit = "bunches" and (tc.genus <> "Crowea" and tc.species <>
         "angustifolia") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "cucullata") or
      tc.unit = "bunches" and (tc.genus <> "Hakea" and tc.species <>
         "platysperma") or
      tc.unit = "bunches" and (tc.genus <> "Juncus" and tc.species <>
         "holoschoenus") or
      tc.unit = "bunches" and (tc.genus <> "Kingia" and tc.species <>
         "australis") or
      tc.unit = "bunches" and (tc.genus <> "Podocarpus" and tc.species
<>
         "drouynianus") or
      tc.unit = "bunches" and (tc.genus <> "Xylomelum" and tc.species <>
         "occidentale")) :
            tc."sum of quantity" = tc."sum of quantity"*10
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Leptocarpus" and
         tc.species = "scariosus"):
            tc."sum of quantity" = tc."sum of quantity"*50
            tc.unit = "single"
      endscan

      scan tc for tc.unit = "bunches" and (tc.genus = "Podocarpus" and
         tc.species = "drouynianus"):
            tc."sum of quantity" = tc."sum of quantity" *20
```

```
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Xylomelum" and
   tc.species = "occidentale"):
      tc."sum of quantity" = tc."sum of quantity" *7
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Kingia" and
   tc.species = "australis"):
      tc."sum of quantity" = tc."sum of quantity" *50
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Juncus" and
   tc.species = "holoschoenus"):
      tc."sum of quantity" = tc."sum of quantity" *40
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Crowea" and
   tc.species = "angustifolium"):
      tc."sum of quantity" = tc."sum of quantity" *15
      tc.unit = "single"
endscan

scan tc for tc.unit = "bunches" and (tc.genus = "Hakea" and
   tc.species = "platysperma"):
      tc."sum of quantity" = tc."sum of quantity" *1
      tc.unit = "single"
endscan
tc.endedit()

top20qry = query

:priv:answer.db   |Grid square|Sum of Quantity  |Taxonid        |
                  |check       |calc sum         |Check          |

endquery

message ("Executing query, please wait")
sleep (3000)
if not executeQBE(top20qry, ":priv:top20.db") then
      errorshow()
endif

top20qry.readfromfile("geocom.qbe")
if not executeQBE(top20qry,":priv:top20int.db") then
      errorshow()
endif

tblcreate=create (":priv:top20com.db")
      with "Grid_num" : "S", "No_species" : "S", "Qty": "N"
      key "Grid_num"
endcreate

destTbl = ":priv:top20com.db"
srcTbl.attach(":priv:top20int.db")
srcTbl.add (destTbl)
```

```
      tblcreate = create (":priv:top20cyr.db")
            with "Year_of_Harvest": "A4"
      endcreate

      tc.open (":priv:top20cyr.db")
      tc.edit()
      tc.insertRecord()
      tc.Year_of_Harvest= top20compare

      top20rsl.open ("geostem.rsl")
else
      usrChoice = msgquestion ("Do you wish to view",
         "Records for 1980/81 only available")
      Switch
            case usrChoice = "Yes":
            top20rsl.open("geo_hop.rsl")
            case usrChoice = "No" :
            return
      endSwitch
endif
endMethod
```

REPLIB.LSL Library

REPLIB contains custom methods for generating reports and for the user in the form REPORT.FSL.

print_renew

```
method print_renew()
var
  theChoice string
  renew, norenew Report
endvar

theChoice = msgYesNoCancel ("Letters","Do you wish to print (check
printer set to letterhead)")
switch
   case thechoice = "Yes" : renew.open ("renew.rsl")
   renew.print()
   norenew.open ("norenew.rsl")
   norenew.print()
   otherwise                          :return
endswitch
endmethod
```

Print_lists

Opens the renewlst.rsl and norenlst.rsl reports to print the monthly lists of licensees whose licences are due to expire.

```
method print_lists()
var
theChoice string
renew, norenew Report
endvar

theChoice = msgYesNoCancel ("Renew Lists","Do you wish to print (check
printer set to plain paper)")
switch
   case theChoice = "Yes" : renew.open ("renewlst.rsl")
   renew.print()
   norenew.open ("norenlst.rsl")
   norenew.print()
   otherwise                          :return
endswitch
endmethod
```

floraind

Opens the floraind.rsl report showing details of returns for each Flora Industry Region. These details can also be printed.

```
method floraind()
var
florind_ret report
repinfo ReportPrintInfo
endvar
florind_ret.open ("florind.rsl")
endmethod
```

regret

Opens the reg_ret.rsl report showing details of returns for each CALM District. These details can also be printed.

```
method regret()
var
reg_ret report
repinfo ReportPrintInfo
endvar
reg_ret.open ("reg_ret.rsl")
endmethod
```

distret

Opens the dis_ret.rsl report showing details of returns for each CALM District. These details can also be printed.

```
method distret()
var
dis_ret report
repinfo ReportPrintInfo
endvar
dis_ret.open ("Dist_ret.rsl")
endmethod
```

blockret

Opens the blockret.rsl report showing details of returns for each CALM District. These details can also be printed.

```
method blockret()
var
   block_ret    report
   repinfo      ReportPrintInfo
endvar
block_ret.open ("blockret.rsl")
endmethod
```

block_pick

Opens the end_dis.rsl report showing details of persons endorsed by CALM District and location. These details can also be printed.

```
method block_pick()
var
end_dis report
repinfo ReportPrintInfo
endvar
end_dis.open ("end_dis.rsl")
endmethod
```

end_spec

Opens the end_spec.rsl form to show summary of species allocated in endorsement by block for a nominated CALM Region.

```
method end_spec()
var
end_spec report
repinfo ReportPrintInfo
endvar
end_spec.open ("end_spec.rsl")
endmethod
```

dis_spec

Opens the dis_spec.rsl report showing details of species endorsed by block for each CALM District. These details can also be printed.

```
method dis_spec()
var
dis_spec report
repinfo ReportPrintInfo
endvar
dis_spec.open ("dis_spec.rsl")
endmethod
```

vehicle_dis

Opens the veh_dis.rsl report showing details of vehicles by block for each CALM District. These details can also be printed.

```
method vehicle_dis()
var
veh_dis report
repinfo ReportPrintInfo
endvar
veh_dis.open ("veh_dis.rsl")
endmethod
```

end_reg

Opens the end_reg.rsl report showing details of persons endorsed by CALM Region and location. These details can also be printed.

```
method end_reg()
var
end_reg report
repinfo ReportPrintInfo
endvar
end_reg.open ("end_reg.rsl")
endmethod
```

veh_reg

Opens the veh_reg.rsl report showing details of vehicles by block for each CALM Region. These details can also be printed.

```
method vehicle_reg()
var
veh_reg report
repinfo ReportPrintInfo
endvar
veh_reg.open ("veh_reg.rsl")
endmethod
```

Reports

FIDMS uses reports to print out summaries of the information stored in its tables. Click a topic for information on a FIDMS report.

ANCA.RSL
BLOCKRET.RSL
BORQUOTA.RSL
CPASEED.RSL
CPACOMP.RSL
CPAHOP.RSL
DIS_SPEC.RSL
DIST_RET.RSL
END_DIS.RSL
END_REG.RSL
END_SPEC.RSL
ENDORSE.RSL
FLORALIC.RSL
FLORARET.RSL
FLORAIND.RSL
GEO_HOP.RSL
GEOSTEM.RSL
GEOSEED.RSL
GRIDSEED.RSL
GRID.RSL
NORENEW.RSL
NORENLST.RSL
PCODE.RSL
PICK_REG.RSL
PICKSEED.RSL
REG_RET.RSL
RENEW.RSL
RENEWLST.RSL
SDCOGEN.RSL
SPECCNT.RSL
TOP20COM.RSL
TOP20STM.RSL
TOPCOGEN.RSL
TOPSDCOM.RSL
TOPSTGEN.RSL
VEH_DIS.RSL
VEH_REG.RSL

anca.rsl

Provides a summary of the amount of harvest amounts for all species, products and land tenures for a user-defined year. This report is forwarded to ANCA on a twice-yearly basis in accordance with the management program.

blockret.rsl

Provides a summary of return details (including person and licence details, species details and areas) for a nominated Block or area (chosen via dialog box)

borquota.rsl

Provides a summary of the quota amounts for *Boronia megastigma* for CALM Districts on Crown land. Quotas are divided into blossom, sprays or seed and totals are given by District for each product and by total for the product. Uses table borquota.db

cpaseed.rsl

Provides a breakdown of the number and percentage of seed picked from Crown land and private property in any two user-defined years.

cpacomp.rsl

Provides a breakdown of the number and percentage of stems picked from Crown land, private property natural stands and artificially propagated stands in any two user-defined years.

cpahop.rsl

Provides a breakdown of the number and percentage of stems picked from Crown land, private property natural stands and artificially propagated stands in any user-defined year and for 1980/81 when Burgman and Hopper's study was undertaken.

dis_ret.rsl

Provides a summary of return details (including person and licence details, species details and areas) for a nominated District (chosen in the Report Form).

dis_spec.rsl

Provides a summary of the total harvest amounts issuedin endorsements (CLM 727) for each species by block for a CALM District.

end_dis.rsl

Provides a summary of the person and licence details for endorsements (CLM 727) issued by a nominated District (chosen in the Report Form).

end_spec.rsl

Provides a summary of the total harvest amounts issued in endorsements (CLM 727) for each species by block for a CALM Region.

end_reg.rsl

Provides a summary of the person and licence details for endorsements (CLM 727) issued by a nominated CALM Region (chosen in the Report Form).

endorse.rsl

This report is used to print details of an endorsement issued by CALM Region/District Offices. Uses the endprn.db table.

floraind.rsl

Provides a summary of return details (including person and licence details, species details and areas) for a nominated Flora Industry Region (chosen in the Report Form).

floralic.rsl

This report shows the results of a licence search of a user-defined licence number in the Person and Licensing Form printBtn method.

floraret.rsl

Provides a summary of the return details for a licence number requested via the Flora Industry Return Form.

geo_hop.rsl

This report compares the number of species and the percentage of the total of single stems harvested from each grid square in a user-defined year and in 1980/81 with data from Burgman and Hopper's study.

geoseed.rsl

This report compares the number of species and the percentage of the total of seed (in kg) harvested from each grid square in two user-defined years.

geostem.rsl

This report compares the number of species and the percentage of the total of single stems harvested from each grid square in two user-defined years.

grid.rsl

This report compares the relative geographic spread of single stem harvesting in two user-defined years. Species are ordered by their level of harvest from highest to lowest.

gridseed.rsl

This report compares the relative geographic spread of seed harvesting in two user-defined years. Species are ordered by their level of harvest from highest to lowest (in kg).

norenew.rsl

Provides a form letter for licensees who have not complied with the requirement to submit returns. Letters are printed on letterhead (a dialog box provides a reminder to the user to check the paper type).

norenlst.rsl

Provides a list of licensees who have not complied with the requirement to submit returns. The lists are printed on plain paper (a dialog box provides a reminder to the user to check the paper type).

pcode.rsl

Provides a summary of licences and licensee details.

pick_reg.rsl

Shows the relative importance of areas by calculating the total amount of single stems harvested from each of the CALM Picking Regions in any two user-defined years.

pickseed.rsl

Shows the relative importance of areas by calculating the total amount of seed (in kg) harvested from each of the CALM Picking Regions in any two user-defined years.

reg_ret.rsl

Provides a summary of return details (including person and licence details, species details and areas) for a nominated CALM Region (chosen in the Report Form).

renew.rsl

Provides a form letter for licensees who have complied with the requirement to submit returns. Letters are printed on letterhead (a dialog box provides a reminder to the user to check the paper type).

renewlst.rsl

Provides a list of licensees who have complied with the requirement to submit returns. The lists are printed on plain paper (a dialog box provides a reminder to the user to check the paper type).

sdcogen.rsl

This report shows the total harvest of seed (in kg) for each genus and the total number of species which are commercially exploited within each genus for any two user-defined years. The data are ordered from highest quantity to lowest. The total harvest and number of species are also calculated.

speccnt.rsl

This report compares the number of species used in the commercial flower and seed trades in all years since 1991 and in each of the surveys that have been undertaken into the industry prior to this (Rye, 1979 and Burgman and Hopper, 1982). The table is ordered by year from lowest to highest.

top20com.rsl

This report shows the total harvest of single stems for each species and the percentage of the total harvest that each species constitutes for any two user-defined years. The data are ordered from highest quantity to lowest. The total harvest and the percentage of the total is also calculated.

top20stm.rsl

This report shows the total harvest of single stems for each species and the percentage of the total harvest that each species constitutes for any a user-defined year and for 1980/81 from Burgman and Hopper (1982) data. The data are ordered from highest quantity to lowest. The total harvest and the percentage of the total is also calculated.

topcogen.rsl

This report shows the total harvest of single stems for each genus and the total number of species which are commercially exploited within each genus for any two user-defined years. The data are ordered from highest quantity to lowest. The total harvest and number of species is also calculated.

topsdcom.rsl

This report shows the total harvest of seed (in kg) for each species and the percentage of the total harvest that each species constitutes for any two user-defined years. The data are ordered from highest quantity to lowest. The total harvest and the percentage of the total is also calculated.

topstgen.rsl

This report shows the total harvest of single stems for each genus and the total number of species which are commercially exploited within each genus for any user-defined year and for 1980/81 when Burgman and Hopper's study was carried out. The data are ordered from highest quantity to lowest. The total harvest and number of species is also calculated.

veh_dis.rsl

Provides a summary of the vehicle details for endorsements (CLM 727) issued by a nominated District (chosen in the Report Form).

veh_reg.rsl

Provides a summary of the vehicle details for endorsements (CLM 727) issued by a nominated Region (chosen in the Report Form).

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

FIDMSLIB_Library_canClose
FIDMSLIB_Library_isCalled