DISTRIBUTION SHEET

Distribution

From:
R. T. AdamsDate:
08/01/94

-Project Title/Work Order:

Liquid Effluent Monitoring Information System (LEMIS) - Test Plans Release 1.0

EDT No.: 141244

ECN No.:

Name	MSIN	With Attachment	EDT/ECN & Comment	EDT/ECN Only
R. T. Adams	B1-18	X		
A. J. Diliberto	R3-46			X
D. A. Guettler	B1-18			X
L. L. Lang	G3-15	X		
Central Files (2)	L8-04	X		
LEMIS Library (2)	B1-18	X		
OSTI (2)	L8-07	X		

2. To: (Receiving Organization) Distribution		3. From: (Originating Organization) Information and Scientific Systems	4. Related EDT No.: N/A
5. Proj./Prog./Dept./Div.: Liquid Effluent Monitoring Information System (LEMIS)		6. Cog. Engr.: R. T. Adams	7. Purchase Order No.: N/A
8. Originator Remarks: For Release		9. Equip./Component No.: N/A	
		10. System/Bldg./Facility: N/A	
		12. Major Assm. Dwg. No.: N/A	
11. Receiver Remarks:		13. Permit/Permit Application No.: N/A	
		14. Required Response Date: N/A	

15. DATA TRANSMITTED					(F)	(G)	(H)	(I)
(A) Item No.	(B) Document/Drawing No.	(C) Sheet No.	(D) Rev. No.	(E) Title or Description of Data Transmitted	Impact Level	Reason for Transmittal	Originator Disposition	Receiver Disposition
1	WHC-SD-WM-TP-163		0	Liquid Effluent Monitoring Information System (LEMIS) - Test Plans Release 1.0	NA	2		

16. KEY		
Impact Level (F)	Reason for Transmittal (G)	Disposition (H) & (I)
1, 2, 3, or 4 (see MRP 5.43)	1. Approval 2. Release 3. Information 4. Review 5. Post-Review 6. Dist. Receipt Acknow. Required	1. Approved 2. Approved w/comment 3. Disapproved w/comment 4. Reviewed no/comment 5. Reviewed w/comment 6. Receipt acknowledged

17. SIGNATURE/DISTRIBUTION (See Impact Level for required signatures)										(G)	(H)
Reason	Disp.	(J) Name	(K) Signature	(L) Date	(M) MSIN	(J) Name	(K) Signature	(L) Date	(M) MSIN	Reason	Disp.
2		Cog. Eng. R. T. Adams	<i>R. T. Adams</i>	8/25/94	B1-18						
2		Cog. Mgr. D. A. Guettler	<i>D. A. Guettler</i>	8/25/94	B1-18						

18. Signature of EDT Originator <i>R. T. Adams</i> Date <u>8/25/94</u>	19. Authorized Representative for Receiving Organization <i>L. D. Lang</i> Date <u>8-21-94</u>	20. Cognizant/Project Engineer's Manager <i>D. A. Guettler</i> Date <u>8/25/94</u>	21. DOE APPROVAL (if required) Ltr. No. ☐ Approved ☐ Approved w/comments ☐ Disapproved w/comments
--	--	--	--

RELEASE AUTHORIZATION

**Document
Number:** WHC-SD-WM-TP-163 Rev. 0

**Document
Title:** Liquid Effluent Monitoring Information System
(LEMIS) Test Plans Release 1.0

Release Date: August 30, 1994

* * * * *

This document was reviewed following the
procedures described in WHC-CM-3-4 and
is:

APPROVED FOR PUBLIC RELEASE

* * * * *

WHC Information Release Administration Specialist:

Christine Willingham

C. Willingham

(Signature)

August 30, 1994

(Date)

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

SUPPORTING DOCUMENT

1. Total Pages 544

2. Title

Liquid Effluent Monitoring Information System
(LEMIS) Test Plans Release 1.0

3. Number

WHC-SD-WM-TP-163

4. Rev No.

0

5. Key Words

project
DMR
liquid effluent

6. Author

Name: R. T. Adams

Signature

Organization/Charge Code 62230/R40630

7. Abstract

The Liquid Effluent Monitoring Information System (LEMIS) is being developed as the organized information repository facility in support of the liquid effluent monitoring requirements of the Tri-Party Agreement. It is necessary to provide an automated repository into which the results from liquid effluent sampling will be placed. This repository must provide for effective retention, review, and retrieval of selected sample data by authorized persons and organizations.

This System Architecture document is the aggregation of the DMR P+ methodology project management deliverables. Together they represent a description of the project and its plan through four Releases, corresponding to the definition and prioritization of requirements defined by the user.

8. PURPOSE AND USE OF DOCUMENT - This document was prepared for use within the U.S. Department of Energy and its contractors. It is to be used only to perform, direct, or integrate work under U.S. Department of Energy contracts. This document is not approved for public release until reviewed.

PATENT STATUS - This document copy, since it is transmitted in advance of patent clearance, is made available in confidence solely for use in performance of work under contracts with the U.S. Department of Energy. This document is not to be published nor its contents otherwise disseminated or used for purposes other than specified above before patent approval for such release or has been secured, upon request, from the Patent Counsel, U.S. Department of Energy Field Office, Richland, Wt.

DISCLAIMER - This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or any third party's use or the results of such use of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof or its contractors or subcontractors. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

10.

RELEASE STAMP

OFFICIAL RELEASE (23)
BY WHC
DATE OCT 12 1994

Sta #10

9. Impact Level N/A

This page intentionally left blank.

CONTENTS

1.0	LEMIS	1
1.1	LE0001U1 MAINTAIN EVENT TYPE CODE	1
1.2	LE200NU1 MAINTAIN LEMIS STATUS CODE	17
1.3	LE2110U1 REQUIREMENT DEFINITION	30
1.4	LE2110U2 PROJECT DEFINITION	67
1.5	LE2110U3 QAPP DEFINITION	97
1.6	LE2110U5 SAP DEFINITION	135
1.7	LE2121U1 SCHEDULE SAMPLES	181
1.8	LE4041U1 LEMIS LOCATION MAINTENANCE	212
2.0	GENERIC	237
2.1	GE1410Q1 PERSON LOV	237
2.2	GE2130Q1 DISPLAY MENU	237
2.3	GE2130R1 MENU STRUCTURE REPORT	237
2.4	GE2320Q1 SELECT ROLE	237
2.5	GE2410U1 ASSIGN ROLE	238
2.6	GE2620U1 ASSIGN DATA OWNERSHIP	245
2.7	GE3120Q1 DISPLAY FUNCTION KEY HELP	255
2.8	GE3130Q1 DISPLAY FUNCTION KEY TEMPLATE	255
2.9	GE3210U1 ENTER A COMMENT OR BUG	255
2.10	GE3250R1 PROGRAMMER'S BUG REPORT	255
2.11	GE3420Q1 RELATED DATA POPUP	255
2.12	GE3520Q1 APPLICATION STARTUP SCREEN	255
2.13	GE3721R1 ACCESS LOG REPORT	255
2.14	GE3830R1 ERROR MESSAGES BY CODE	256
2.15	GE4130G1 PICKLIST TEMPLATE FORM	256
2.16	GE6014Q1 ONLINE DOCUMENT DISPLAY	256
2.17	GETEXTU1 TEXTUAL DATA ENTRY	256
3.0	LIBRARY	257
3.1	LI1004U1 MAINTAIN UNIT MEASURE CODE	257
3.2	LI1006U1 TYPE CONSTITUENT CODE	294
3.3	LI1008U1 BOTTLE CODE TABLE MAINTENANCE	306
3.4	LI1009U1 PRESERVATIVE CODE TABLE MAINTENANCE	328
3.5	LI1011U1 MAINTAIN TYPE SAMPLE MATRIX CODE	341
3.6	LI5051Q1 CONSTITUENT PICK LIST	356
3.7	LI5051U1 MAINTAIN CONSTITUTENTS	357
3.8	LI5052U1 LABORATORY MAINTENANCE	390
3.9	LI5053Q1 METHOD PICK LIST	413
3.10	LI5053U1 METHOD MAINTENANCE	414
3.11	LI5053U2 ANALYSIS CLASS MAINTENANCE	453
3.12	LI5053U3 ANALYSIS CLASS PLUS MAINTENANCE	473
3.13	LI5055U1 ASSIGN CONSTITUENTS TO METHODS	493
3.14	LI5056U1 ASSIGN METHODS TO LABORATORY	520

This page intentionally left blank.

LIQUID EFFLUENT MONITORING INFORMATION SYSTEM (LEMIS) TEST PLANS RELEASE 1.0

1.0 LEMIS

1.1 LE0001U1 MAINTAIN EVENT TYPE CODE

LEMIS Module Test Checklist Report

Name: MAINTAIN EVENT TYPE CODE
Short Name: LE0001U1
Purpose: To maintain event type code

Initial Test Date: 11/25/92 JK

Verification Date: 11/29/92 Zn

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: TYPE_EVENT
Title: Event Type

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_EVENT	CHAR	2	()	()	()	()	()
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

Module Help

This screen supports the update and retrieval functions for the event type codes upon which the liquid effluent sampling activities are based. This screen is used to define the codes used by the system to differentiate between the kinds of events that are found in the system. The types of events are requirements, projects, QAPPs, SAPs, and SAFs. Requirements may be subdivided into TPA and

Module Help

its milestones and submilestones.

To store a new event type:

1. Enter the event type identifier. This identifier may be up to 2 characters in length. The identifier must be unique or an error will occur. The field is required.
2. TAB and enter the title of the event type. The title may be from one to 40 characters in length and is required.
3. Press the COMMIT key (F10) to store the data in the database.

To update an existing event type:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the event type to be updated. If you only know part of the identifier, enter that part followed by a percent sign (%). This will cause those event types that begin with the partial identifier to be queried.
3. Press the RUN QY key (Shift/F12). The event type will be displayed on the screen; if more than one event type was queried, the first one matching the query will be displayed. To locate the desired event type:
 - a. Press the down arrow key until the desired event type is displayed.
4. Press the TAB key until the cursor is at the field to be modified.
5. Enter the new data in the field and press ENTER.
6. If more than one field is to be updated, repeat the above process for each field.
7. Press the COMMIT key (F10) to store the changes in the database.

Event types may not be deleted if associated with any existing events.
To delete an event type:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the event type to be deleted.
3. Press the RUN QY key (Shift/F12). The event type will be displayed on the screen.
4. Press the DELETE grey key. The screen should clear.
5. Press the COMMIT key (F10) to update the database.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

Module Test Plan

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

- () 4 - Secondary testing in progress.
- (☒) 5 - Secondary testing complete, problems found.
 - Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
 - Developer _____ Date Complete _____
- () 8 - Secondary testing complete, no problems found.

When problems are found:

1. The tester passes the checklist to the project manager for review and action.

Module Test Plan

2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.

3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.

4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Test status conditions are:

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.

Project Manager *[Signature]* Date Complete 11/30/92

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.

System Architect *[Signature]* Date Complete 11/30/92

Module Test Plan

Module Test Case

(✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

(✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]

(✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

596 (✓) Transaction help describes:

- (✓) Purpose of the transaction
- (✓) Functions performed by the transaction
- (✓) Where multiple page, what happens on each page

(✓) Add a new record by using the insert key.

597 (✓) The insert key clears all blocks of the entry screen.
(✓) A new record can be successfully added and committed.

(✓) Add a new record by scrolling past the last entry using the down arrow key.

HA (✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
(✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
(✓) A new data record can be successfully added.

(✓) Query a single record.

(✓) A single record is retrieved.
(✓) Pressing ~~SHIFT~~ F12 clears the record and permits entering a second query.
(✓) The second query successfully retrieves the desired data.

(✓) Query multiple records using wild cards.

(✓) Pressing F12 clears the screen and makes the transaction ready for a query.
(✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
(✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.

- (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through
- (✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
- (✓) Change each field individually, committing after each change.
- (✓) Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
- (✓) The selected record is deleted from the database. ~~Verify by using SQL*Plus to attempt to retrieve the record.~~
- (✓) ~~Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.~~
- (✓) ~~The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.~~
- (✓) ~~Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date updated, owner role, and any other autogenerated fields.~~
- (✓) ~~All data fields are correctly stored in the database.~~

Block: TYPE_EVENT
Title: Event Type

Block Description

Block Test Case

Block Test Results Expected

Field Name: TYPE_EVENT

Field Description

Block: TYPE_EVENT
 Title: Event Type

Field Name: TYPE_EVENT

Field Test Case

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display table/block help. Verify that the narrative adequately describes the purpose and functions of the table/block in end-user terms.
 - (✓) Table/block help describes:
 - (✓) Purpose of the table/block
 - (✓) Functions performed by the table/block
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all fields of the entry block.
 - (✓) A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - (✓) All fields of the table/block blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.

Block: TYPE_EVENT
 Title: Event Type

Field Name: TYPE_EVENT

Field Test Results Expected

- 598*
- (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the block and makes the block ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. ~~Verify by running the same query through~~
 - (✓) A query using a prefix % wildcard retrieves the correct data. ~~Verify by running the same query under SQL*Plus.~~
 - (✓) A query using both prefix and suffix % wildcards retrieves the correct data. ~~Verify by running the same query through SQL*Plus.~~
 - (✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
 - (✓) Change each field individually, committing after each change.
 - (✓) Change each field, as per the field test plan, for both valid ~~and~~ and invalid values.
 - (✓) Delete an existing record.
 - (✓) The selected record is deleted from the database. ~~Verify by using SQL*Plus to attempt to retrieve the record.~~
 - (-) ~~Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.~~
 - (-) ~~The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.~~
 - (-) ~~Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date-updated, owner role, and any other autogenerated fields.~~
 - (-) ~~All data fields are correctly stored in the database.~~
 - (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user

Dup. TEST

Block: TYPE_EVENT
Title: Event Type

Field Name: TYPE_EVENT

Field Test Results Expected

terms.

- (✓) Field help describes:
 - (✓) Purpose of the field
 - ~~H/A~~ LOV reference if available
 - (✓) Valid range of values where present
 - ~~H/A~~ Required indicator when field is NOT NULL
 - ~~H/A~~ Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~ Dup.
- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- (✓) Store a series of records using the SHIFT/F6 [duplicate field] key to carry forward field values from the previously stored record. UNIQUE FIELD
- ~~() Display the JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.~~
- ~~() The JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.~~
- ~~() Use SQL*Plus to verify all data fields are processed correctly.~~
- ~~() All data fields are correctly stored in the database.~~
- (✓) Any alphanumeric characters may be entered into the field. YES
- (✓) Delete an existing value.
- (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. ERROR diff.
- (✓) Attempt to bypass the field. An error message should tell you

Block: TYPE_EVENT
Title: Event Type

Field Name: TYPE_EVENT

Field Test Results Expected

that the field data must be entered.

- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~

Auto CAPS

Block: TYPE_EVENT
 Title: Event Type

Field Name: DESCRIPTION

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - N/A If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

() Field help describes:

- (✓) Purpose of the field
- N/A) LOV reference if available
- (✓) Valid range of values where present
- (✓) Required indicator when field is NOT NULL
- N/A Default value when appropriate

- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~

Block: TYPE_EVENT
Title: Event Type

Field Name: DESCRIPTION

Field Test Results Expected

- (✓) Change the field, committing after each change.
- (✓) Change the field where a range of valid values exists:
N/A () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- (✓) Store a series of records using the SHIFT/END [duplicate field] key to carry forward field values from the previously stored record.
- () ~~Display the JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.~~
- () ~~The JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.~~
- () ~~Use SQL*Plus to verify all data fields are processed correctly.~~
- () ~~All data fields are correctly stored in the database.~~
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- () Enter data into the field, backspace the data out of the field and TAB to the next field. ~~The database should contain a null value.~~ Dup
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. An error should result.

Transaction: LE0001U1 Maintain Event Type

=====

Comment ID : 596 Priority: LOW Date: 25-NOV-92

=====

Status : DOCCHNG Date: 26-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By : LARRY TOWNER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40208 Form running in query-only mode. Can't chan

Table : TYPE_EVENT

Block : TYPEVE

Field : TYPEVE.TYPE_EVENT

DESCRIPTION

item 4 of help text did not complete sentence.

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE0001U1 Maintain Event Type

=====

Comment ID : 597 Priority: LOW Date: 25-NOV-92

=====

Status : ASSIGNED Date: 26-NOV-92 Class:

Type : COMMENT

Assigned To: TODD ADAMS

Entered By : LARRY TOWNER

System ID : 92000008893 Oracle ROWID: 00000355.0000.0007

Last Error : -0

Last Msg : FRM-40208 Form running in query-only mode. Can't chan

Table : TYPE_EVENT

Block : TYPEVE

Field : TYPEVE.TYPE_EVENT

DESCRIPTION

should the query be displayed in alphabetical order? It didn't during
my test.FIXED

Transaction: LE0001U1 Maintain Event Type

```
=====
Comment ID : 598    Priority: LOW          Date: 25-NOV-92 ,
=====
Status      : ASSIGNED , Date: 26-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : LARRY TOWNER
System ID   : 92000009105      Oracle ROWID: 00000355.0008.0007
Last Error  : -0
Last Msg    : FRM-40400 Transaction complete -- 1 records posted an
Table       : TYPE_EVENT
Block       : TYPEVE
Field       : TYPEVE.TYPE_EVENT
```

DESCRIPTION

expect Type field to be unique, but allows user to update/change.

O.K. TO UPDATE

Programmer's Bug Report

Transaction: LE0001U1 Maintain Event Type

```
=====
Comment ID : 599   Priority: LOW           Date: 25-NOV-92 '
=====
Status      : ASSIGNED   Date: 26-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : LARRY TOWNER
System ID   : 92000008893      Oracle ROWID: 00000355.0000.0007
Last Error  : -0
Last Msg    : FRM-40100 At first record.
Table       : TYPE_EVENT
Block       : TYPEVE
Field       : TYPEVE.DESCRPTION
```

DESCRIPTION

field can be null, but help does not say so.

CORRECTIVE ACTION

11/26/92 - Ken - As far as I know is should not be null?

CHANGED FIELD TO NOT NULL

1.2 LE200NU1 MAINTAIN LEMIS STATUS CODE

LEMIS Module Test Checklist Report

Name: MAINTAIN LEMIS STATUS CODE
 Short Name: LE200NU1
 Purpose: Maintain the LEMIS status codes.

Initial Test Date: 11/25/92 JK
 Verification Date: 11/29/92 LG

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: TYPE_LEMIS_STATUS
 Title: Lemis Status Types

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_STATUS	CHAR	4	()	()	()	()	()
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

 This form is used to add, update or delete event status codes.

Module Help

 This module supports the maintenance and retrieval of lemis status codes. This table maintains a list of valid codes that describe the types of lemis status that may occur.

Module Help

To store a new lemis status code:

1. Use the down arrow key to move the cursor to a blank line.

If all of the spaces are filled and more lemis status codes are to be added, press the down arrow key to the bottom entry and then continue to press the down arrow key until a blank entry appears. The already entered codes will scroll upward, leaving space for an additional code to be entered.

2. Enter the appropriate lemis status code.
3. Enter a 1-30 character description of the code.
4. To add more codes, press the down arrow key to the next blank space and repeat the entry process.
5. Press the COMMIT key (F10) to store the selected codes in the database.

To update an existing code:

1. Press the down arrow key until the cursor displays the entry to be updated.
2. If the code is to be changed, enter the replacement code over the existing code. CAUTION: Changing a lemis status code will cause the replacement value to be entered everywhere in the database where the original value was used.
3. If the description is to be changed, TAB to the description field and make the change.
4. Press the COMMIT key (F10) to update the selected codes in the

To delete an existing code:

1. Press the down arrow key until the cursor is located at the code to be deleted.
2. Press the DELETE key. NOTE: If the code is being used by other data in the database, it cannot be deleted. An error message will be displayed.
3. If no error occurs, press the COMMIT key (F10) to delete the code from the database.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.

Module Test Plan

4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
- (✓) 5 - Secondary testing complete, problems found.
 Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
 Developer _____ Date Complete 11/25/92
- (✓) 8 - Secondary testing complete, no problems found.

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the

Module Test Plan

corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.

4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.

Project Manager ZGJ Date Complete 11/29/52

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu

System Architect EA Date Complete 11/30/52

Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

(✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]

- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

(✓) Transaction help describes:

- (✓) Purpose of the transaction
- (✓) Functions performed by the transaction
- (✓) Where multiple page, what happens on each page

*No problem w/ HELP
(no bug report from #600)*

- () Add a new record by using the insert key.

- (✓) The insert key clears all blocks of the entry screen.
- (✓) A new record can be successfully added and committed.

- (✓) Add a new record by scrolling past the last entry using the down arrow key.

N/K) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.

- (✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
- (✓) A new data record can be successfully added.

- (✓) Query a single record.

- (✓) A single record is retrieved.
- (✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
- (✓) The second query successfully retrieves the desired data.

- (✓) Query multiple records using wild cards.

- (✓) Pressing F12 clears the screen and makes the transaction ready for a query.
- (✓) A query composed of a trailing % wildcard retrieves the desired data. ~~Verify by running the same query through SQL*Plus.~~
- (✓) A query using a prefix % wildcard retrieves the correct data. ~~Verify by running the same query under SQL*Plus.~~
- (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through

- (✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~

- (✓) Change each field individually, committing after each change.
- (✓) Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
- (✓) The selected record is deleted from the database. ~~Verify by using SQL*Plus to attempt to retrieve the record.~~

NA () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.

NA () All data fields are correctly stored in the database.

Block: TYPE_LEMIS_STATUS
Title: Lemis Status Types

Block Description

Block Test Case

Block Test Results Expected

(✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

(✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]

() Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

NA () Transaction help describes:

- () Purpose of the transaction
- () Functions performed by the transaction
- () Where multiple page, what happens on each page

dup
Test

Block: TYPE LEMIS STATUS
 Title: Lemis Status Types

Block Test Results Expected

- () Add a new record by using the insert key.
- N/A* () The insert key clears all blocks of the entry screen.
- () A new record can be successfully added and committed.
- () Add a new record by scrolling past the last entry using the down arrow key.
- () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
- () All fields of the transaction blank when the end of the string of retrieved records is reached.
- () A new data record can be successfully added.
- () Query a single record.
- () A single record is retrieved.
- () Pressing SHIFT/F12 clears the record and permits entering a second query.
- () The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
- () Pressing F12 clears the screen and makes the transaction ready for a query.
- () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- () Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Change each field individually, committing after each change.
- () Change each field, as per the field test plan, for both valid and invalid values.
- () Delete an existing record.
- () The selected record is deleted from the database. Verify

*dup.
Test*

*dup
Test*

*dup
Test*

Block: TYPE_LEMIS_STATUS
Title: Lemis Status Types

Block Test Results Expected

by using SQL*Plus to attempt to retrieve the record.

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- () All data fields are correctly stored in the database.

drop table
module 2
Block
one same

Field Name: TYPE_STATUS

Field Description

Field Test Case

- (✓) Begin field with a space. *OK*
- (✓) Enter data beyond length of field.
 - ~~✗~~ If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

won't let you - it works.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

Block: TYPE_LEMIS_STATUS
 Title: Lemis Status Types

Field Name: TYPE_STATUS

Field Test Results Expected

- (✓) Field help describes:
 - (✓) Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - NR (✓) Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - NR () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record. *won't allow*
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- NR () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. An error should result.
- (✓) Delete an existing value.
- (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

diff msg.
error msg-40202:
"Field must be entered"

Block: TYPE_LEMIS_STATUS
Title: Lemis Status Types

Field Name: TYPE_STATUS

Field Test Results Expected

- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: TYPE_LEMIS_STATUS
 Title: Lemis Status Types

Field Name: DESCRIPTION

Field Description

Field Test Case

() TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.

() Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.

(✓) Begin field with a space.

(✓) Enter data beyond length of field.

NA If autoskip is in effect, cursor will skip to the next field.

(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

(✓) Clear an existing field and attempt to update the record.

"Field must be entered"

Field Test Results Expected

() Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

() Field help describes:

(✓) Purpose of the field

NA LOV reference if available

(✓) Valid range of values where present

(✓) Required indicator when field is NOT NULL

NA Default value when appropriate

(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and

Block: TYPE_LEMIS_STATUS
Title: Lemis Status Types

Field Name: DESCRIPTION

Field Test Results Expected

~~after the changes were made.~~

- (✓) Change the field, committing after each change.
- (✓) Change the field where a range of valid values exists:
 - NA () Value below valid range should display message COM00101 or
 - NA () Value above valid range should display message COM00101 or COM00103 (if no LOV).

- (✓) Store a series of records using the SHIFT/F⁶ [duplicate field] key to carry forward field values from the previously stored record.

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

- () Use SQL*Plus to verify all data fields are processed correctly.

- NA () Bypass the field by using the TAB key. *see prob log #602*
 - () No error should occur.
 - () The database contains a null value for the field.

- () Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- NA () Delete an existing value. The field should be cleared and stored as a null value.

see #602

- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. An error should result.
- (✓) Delete an existing value.

- (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

diff msg

- (✓) Attempt to bypass the field. An error message should tell you

Block: TYPE_LEMIS_STATUS
Title: Lemis Status Types

Field Name: DESCRIPTION

Field Test Results Expected

that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

YES

1.3 LE2110U1 REQUIREMENT DEFINITION

LEMIS Module Test Checklist Report

Name: REQUIREMENT DEFINITION
 Short Name: LE2110U1
 Purpose: Maintain Requirement Information.

Initial Test Date: 11/29/92 *RC* 12/1/92 *JK*

Verification Date: 12/1/92 *JK*

Help	Disp Fkeys	Security			
		Ins	Del	Query	Upd
()	()	()	()	()	()

Block: EVENT

Title: Requirement

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
TITLE_EVENT	CHAR	40	()	()	()	()	()
TYPE_EVENT	CHAR	2	()	()	()	()	()
REV_EVENT	NUMBER	3	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: EVENT

Title: Parent Requirement

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
TITLE_EVENT	CHAR	36	()	()	()	()	()
TYPE_EVENT	CHAR	2	()	()	()	()	()

Block: EVENT_NEST

Title: Parent Requirement

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: LEMIS_STATUS

Title: Requirement Status

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_STATUS	CHAR	4	()	()	()	()	()
DATE_STATUS	DATE	9	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: TYPE_EVENT

Title: Event Type

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DESCRIPTION	CHAR	30	()	()	()	()	()

Block: TYPE_EVENT
Title: Event Type

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DESCRIPTION	CHAR	30	()	()	()	()	()

Block: TYPE_LEMIS_STATUS
Title: Status Types

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

A requirement, for LEMIS, is a high-level basis for the liquid effluent program. In this case, the highest level requirement is the TPA. Subordinate to the TPA are TPA milestones; TPA submilestones are subordinate to milestones.

A requirement must be able to identify its higher level requirement. This reference must be optional to allow for no higher level reference of the higher level.

Text may be associated with a requirement.

Requirements are optionally related to:

1. Organizations associated. Use LOV link to organization list. Multiple organizations may be associated with specific relationship codes.
2. Persons associated. Use literal values. Multiple persons can be associated with specific relationship codes.

Requirements are NOT associated with methods or locations.
Requirements are NOT associated with procedures.

Requirement status information will default to CR (create), with a default date of the current date (the date will likely be changed by the operator). Text may be associated with the status. Multiple status lines are possible, display at least two.

Module Help

This screen supports the update and retrieval functions for the requirement definitions upon which the liquid effluent sampling activities are based. This screen is used to define the TPA and its milestones and submilestones that relate to liquid effluent monitoring. The status of a requirement may also be updated from this screen.

To store a new requirement:

1. Enter the requirement identifier. This identifier may be up to 20 characters in length. If the requirement is the TPA, "TPA" is adequate as the identifier. TPA milestones and submilestones should be identifier as "TPA MILESTONE 17.nn". The identifier must be unique or an error will occur. The field is required.
2. TAB and enter the title of the requirement. The title may be from one to 40 characters in length and is required.
3. TAB to the REQ TYPE field. Use LOV to select the appropriate type code:
 - a. Press the down arrow key until the cursor points to the desired code.
 - b. Press the ENTER key to select the desired code onto the screen.
 - c. The desired code and its description will be displayed.
4. TAB to the REV field. This is the revision identifier of the requirement. The field is required. A default value of zero is displayed. To accept the default, press the ENTER key. To enter a different revision level, key the revision number over the default and press ENTER. (A number is required.)

The Text? flag indicates whether free-form text is associated with the requirement. If the flag is set to 'N', no text is present. To add text, press the TEXT key (F7). A popup screen will appear that will permit you to enter as much text as desired. Several different types of text may be available. Once the text has been stored, the text flag will be set to 'Y'. Additional text processing assistance can be obtained by pressing CTRL/K while the text popup window is being displayed.

5. Press the COMMIT key (F10) to store the data in the database.

To update an existing requirement:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the requirement to be updated.
3. Press the RUN QY key (Shift/F12). The requirement will be displayed on the screen.
4. Press the TAB key until the cursor is at the field to be modified.
5. Enter the new data in the field and press ENTER.
6. If more than one field is to be updated, repeat the above

Module Help

process for each field.

7. Press the COMMIT key (F10) to store the changes in the database.

If the text associated with the requirement is to be modified:

1. Press the TEXT key (F7). The default description text will be displayed. If you wish to modify another text type:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type will be displayed.
2. Press the PAGE DOWN key to move the cursor to the text block.
3. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
4. Press the COMMIT key (F10) to update the database.

Requirements may not be deleted if subordinate requirements or projects are present. To delete a requirement:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the requirement to be updated.
3. Press the RUN QY key (Shift/F12). The requirement will be displayed on the screen.
4. Press the DELETE grey key. The screen should clear.
5. Press the COMMIT key (F10) to update the database.

Each requirement may be assigned to a higher level requirement. For example, a TPA milestone is assigned to the TPA; a submilestone is assigned to a milestone.

To associate a requirement with its parent:

1. Press the PAGE DOWN key to move the cursor to the REQ ID field of the Parent Requirement block.
2. Enter the identifier of the parent requirement, if known. If not known, press LOV (F9) to display a list of available parent requirements:
 - a. Press the down arrow key until the cursor points to the desired parent requirement.
 - b. Press the ENTER key to select the parent requirement. The parent requirement and its information will be displayed on the screen.
3. Press the COMMIT key (F10) to store the association of the requirement to its parent in the database.

Module Help

To remove the association between a requirement and its parent:

1. Press the PAGE DOWN key to move the cursor to the REQ ID field of the Parent Requirement block.
2. If the requirement identifier displayed is the one whose relationship is to be deleted, press the DELETE key.
3. Press the COMMIT key (F10) to complete the update of the database.

The status of each requirement is tracked in the Requirement Status block. Here you may identify the different types of status conditions and the dates on which the condition occurs. Unlimited text may be added to clarify the status condition. To add a new status condition:

1. Press the PAGE DOWN key to move the cursor to the first status line.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Press LOV to display the available valid status condition codes.
4. Use the up and down arrow keys to select the desired status condition.
5. Press the COMMIT (F10) key to move the status to the screen.
6. To add more statuses, press the down arrow key to the next blank space and repeat the selection process.

If all of the spaces are filled and more statuses are to be added, press the down arrow key and the already entered records will scroll upward, leaving space for an additional record to be added.

7. Press the COMMIT (F10) key to store the selected status conditions in the database as associated with the requirement.

To store text expanding the description of a status condition:

1. Use the up and down arrow keys to move the cursor to the status condition line where text is to be stored.
2. Press the TEXT key (F7). The text popup window will be displayed with cursor at the text type code. If a text type other than that displayed is desired:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type, if any, will be displayed.
3. Press the PAGE DOWN key to move the cursor to the text block.
4. If no text is present, enter the desired text. If text is present:
 - a. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP

Module Help

- (Shift/F1) to display the editing keys available.
 - b. To add new text at the end, press the down arrow key until a blank line appears. Enter the text.
 - 5. Press the COMMIT key (F10) to store the text in the database.
 - 6. Press the ENTER key until the popup window disappears and the requirement screen reappears.
 - 7. If more text is to be added to other status condition entries, use the up and down arrow keys to move the cursor to the desired entry and repeat the above steps.
-

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Module Test Plan

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
Developer _____ Date Complete _____
- (✓) 8 - Secondary testing complete, no problems found.

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Test status conditions are:

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

Module Test Plan

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

() Reviews tests results.

() Reviews problem reports to ensure all corrections done.

() Updates NOTES text to reflect testing results.

Project Manager [Signature] Date Complete 12/1/92

() 20 - The transaction is placed into the production environment.

() Software files in production directories.

() DMR 550 updated to add transaction.

() DMR 900 updated to add transaction.

() Transaction added to production usage menu.

System Architect [Signature] Date Complete 12/1/92

Module Test Case

✓(✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

✓(✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]

✓(✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

✓(✓) Transaction help describes:

✓(✓) Purpose of the transaction

✓(✓) Functions performed by the transaction

✓(✓) Where multiple page, what happens on each page

✓(✓) Add a new record by using the insert key.

✓(✓) The insert key clears all blocks of the entry screen.

✓(✓) A new record can be successfully added and committed.

✓(✓) Add a new record by scrolling past the last entry using the down arrow key.

✓(✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.

✓(✓) All fields of the transaction blank when the end of the string

- ✓(✓) of retrieved records is reached.
 - ✓(✓) A new data record can be successfully added.
 - ✓(✓) Query a single record.
 - ✓(✓) A single record is retrieved.
 - ✓(✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - ✓(✓) The second query successfully retrieves the desired data.
 - ✓(✓) Query multiple records using wild cards.
 - ✓(✓) Pressing F12 clears the screen and makes the transaction ready for a query.
 - ✓(✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - ✓(✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - ✓(✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through
 - ✓(✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - ✓(✓) Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
 - ✓(✓) Delete an existing record.
 - ✓(✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- NA
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - () All data fields are correctly stored in the database.

Block: EVENT
Title: Requirement

Block Description

The event table is a multi-use table. It identifies those activities that are defined to guide and control the liquid effluent program.

An event, as currently (10/9/92) defined may be:

- The TPA itself.
- A TPA milestone (milestone 17 for LEMIS and will be known as a project.
- A submilestone within milestone 17 (will be known as a task).
- The QAPP defining LEMIS.
- The SAPs defining the waste stream processing.
- The SAFs defining the actual sampling activities.

In the hierarchy of things:

- The TPA is the top level.
- TPA Milestone 17 is dependent on the TPA.
- TPA submilestones of 17 depend on Milestone 17.

Project

On HEIS, a project is a funded program, such as 300-Area cleanup. For release one of LEMIS, all sampling will be part of the same project, which is Liquid Effluent Sampling, or Tri-Party Agreement milestone number 17.

A project is very tightly coupled to its QAPjP which describes many aspects of how the project will be done.

LEMIS will record the project for each sample that is recorded. During release one, project book keeping may be invisible to the user, and may be done automatically by LEMIS, since only one project will exist.

A project will have the following attributes:
Project name: "TPA Liquid Effluent Sampling " is suggested.
User Id: M-17-00 is suggested.

A project is related to:
One QAPjP at a time.
Many tasks.
Perhaps a high level location, such as Hanford.

Block: EVENT
 Title: Requirement

Block Description

QAPP/QAPjP (Quality Assurance Project Plan)

The QAPP, covering all 33 major streams, is dependent on TPA Milestone 17 (the project).

For Release One, the project will be related to one QAPjP at a time. However, a QAPjP may be revised and changed many times, so a program (project) may be related to many different QAPjPs (or at least many different versions) over the history of the project.

A QAPjP will have the following characteristics:

ID Number: The document number
 Title: Title of the document
 Rev Number: Revision number of the document

A QAPjP is related to one and only one project.

LEMIS will know the QAPjP title and rev number for each sample that is recorded.

Task

Each project consists of many tasks. On HEIS, a task is typically a number from a project document that itemizes tasks to be performed. For LEMIS, a task is typically a stream specific milestone, such as M-17-XX.

The task stream (which is a high level location), and Sampling Analysis Plan (SAP) are very tightly coupled. A task corresponds to one and only one stream and one and only one SAP at a given time. This allows the user to select a task by either the milestone number or the stream name. Only one SAP describes a task at a time, however the SAP may be revised and changed throughout a program. Also, only one SAF will be active for a task at a time.

For each sample taken, LEMIS will record the corresponding task, the applicable SAP, and the applicable SAF.

A task has the following attributes:

Task name: Descriptive title.
 User Id: M-17-XX is suggested.

A task is related to:

Exactly one project
 Exactly one stream location (a high level location)

Block: EVENT
Title: Requirement

Block Description

One or many sampling locations (the actual sampling locations)
Exactly one SAP at a time
Zero or or active SAFs at a time (but many SAFs may apply over time).

LEMIS will know the task, SAP name and revision, and SAF number for each sample that is recorded.

SAP (Sample Analysis Plan)

The SAPs, one for each waste stream, are dependent on the QAPP. Each SAP is also related to the submilestone (task). At any given point in time, a SAP and task are on a one to one relationship. It is possible for there to be multiple SAPs over time related to a single task.

The SAFs occur as part of the sampling defined by the SAP. Several SAFs may occur for a SAP but it is unlikely that more than one will be "active" at one time. This does not include those pre-scheduled for future sampling. Each SAF has a subordinate Sample-Authorization table entry containing specific information related to the SAF.

Block Test Case

Block Test Results Expected

Field Name: IDENT_EVENT

Field Description

This field uniquely identifies the event.

Field Test Case

✓(✓) Begin field with a space.

✓(✓) Enter data beyond length of field.

() If autoskip is in effect, cursor will skip to the next field.

✓(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

Block: EVENT
Title: Requirement

Field Name: IDENT_EVENT

Field Test Case

✓(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

✓(✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

✓(✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

✓(✓) Field help describes:

- () Purpose of the field
- () LOV reference if available
- ✓(✓) Valid range of values where present
- ✓(✓) Required indicator when field is NOT NULL
- () Default value when appropriate

✓(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

✓(✓) Change the field, committing after each change.

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

✓(✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

Block: EVENT
Title: Requirement

Field Name: IDENT_EVENT

Field Test Results Expected

- () All data fields are correctly stored in the database.
- ✓ (✓) Any alphanumeric characters may be entered into the field.
- ✓ (✓) Attempt to enter data as lower case. ~~An error should result.~~
- ✓ (✓) Delete an existing value.
- ✓ (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- ✓ (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- ✓ (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT
Title: Requirement

Field Name: TITLE_EVENT

Field Description

Field Test Case

- ✓(✓) Begin field with a space.
- ✓(✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - ✓(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ✓(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- ✓(✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- ✓(✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - ✓(✓) Field help describes:
 - ✓(✓) Purpose of the field
 - () LOV reference if available
 - ✓(✓) Valid range of values where present
 - ✓(✓) Required indicator when field is NOT NULL
 - () Default value when appropriate
- ✓(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - ✓(✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- ✓(✓) Store a series of records using the SHIFT/~~FB~~ [duplicate field] key

Block: EVENT
Title: Requirement

Field Name: TITLE_EVENT

Field Test Results Expected

to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

✓ (✓) Any alphanumeric characters may be entered into the field.

✓ (✓) Attempt to enter data as lower case. ~~An error should result.~~

✓ (✓) Delete an existing value.

(✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

✓ (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

✓ (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT
Title: Requirement

Field Name: TYPE_EVENT

Field Description

This is a system-generated logical type code that identifies what supporting item within LEMIS is represented by the row occurrence. The TYPE_EVENT code table contains a list of valid event type codes.

Field Test Case

- 70x
Null
field*
- () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
 - () Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- array
test*

✓(✓) Accept the default.

✓(✓) Override the default with a valid entry value.

N/A() Override the default with an invalid data value.

n/A() Begin field with a space.

✓(✓) Enter data beyond length of field.

() If autoskip is in effect, cursor will skip to the next field.

✓(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

✓(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

Field Test Results Expected

614 ✓(✓) Display table/block help. Verify that the narrative adequately describes the purpose and functions of the table/block in end-user terms.

✓(✓) Table/block help describes:

✓(✓) Purpose of the table/block

✓(✓) Functions performed by the table/block

Block: EVENT
Title: Requirement

Field Name: TYPE EVENT

Field Test Results Expected

- ✓(✓) Add a new record by using the insert key.
 - ✓(✓) The insert key clears all fields of the entry block.
 - ✓(✓) A new record can be successfully added and committed.
- ✓(✓) Add a new record by scrolling past the last entry using the down arrow key.
 - ✓(✓) All fields of the table/block blank when the end of the string of retrieved records is reached.
 - ✓(✓) A new data record can be successfully added.
- ✓() Query a single record.
 - ✓() A single record is retrieved.
 - ✓() Pressing SHIFT/F12 clears the record and permits entering a second query.
 - ✓() The second query successfully retrieves the desired data.
- ✓(✓) Query multiple records using wild cards.
 - () Pressing F12 clears the block and makes the block ready for a query.
 - () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- NA () Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made. Jup
 - () Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values. Jup
- NA (✓) Delete an existing record.
 - () The selected record is deleted from the database. Verify

Block: EVENT
Title: Requirement

Field Name: TYPE_EVENT

Field Test Results Expected

- by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - NK () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - () All data fields are correctly stored in the database.
 - () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - ✓ () Field help describes:
 - () ✓ Purpose of the field
 - ✓ () ✓ LOV reference if available
 - () ✓ Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
 - ✓ () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - ✓ () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
 - ✓ () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
 - () Display the -JN journal record rows for the table to verify that

Block: EVENT
Title: Requirement

Field Name: TYPE_EVENT

Field Test Results Expected

the records changed and deleted above are entered into the journal table correctly.

NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() Bypass the field by using the TAB key.

() No error should occur.

() The database contains a null value for the field.

() Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

() Delete an existing value. The field should be cleared and stored as a null value.

✓✓ () Modify each default value with a valid alternative.

NA () Modify each default value with an invalid alternative.

✓✓ () Any alphanumeric characters may be entered into the field.

✓✓ () Attempt to enter data as lower case. ~~An error should result.~~

Field Name: REV_EVENT

Field Description

This field represents a revision to the event.

Field Test Case

Field Test Results Expected

Block: EVENT
Title: Requirement

Field Name: FLAG_TEXT

Field Description

Field Test Case

Not testable

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.

Block: EVENT
Title: Requirement

Field Name: FLAG_TEXT

Field Test Results Expected

-
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
 - () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
 - () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
 - () Modify each default value with a valid alternative.
 - () Modify each default value with an invalid alternative.
 - () Any alphanumeric characters may be entered into the field.
 - () Attempt to enter data as lower case. An error should result.
 - () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT NEST
 Title: Parent Requirement

Block Description

The table links events to each other. The most common linkage will be that of a series of events that are grouped together under another event definition. Another option is to link a series of events that support the same repetitive set of sampling.

Block Test Case

Block Test Results Expected

no
 testing

Block: LEMIS_STATUS
 Title: Requirement Status

Block Description

Block Test Case

Block Test Results Expected

- ✓(✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
- ✓(✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- ✓(✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - ✓(✓) Transaction help describes:
 - ✓(✓) Purpose of the transaction
 - ✓(✓) Functions performed by the transaction
 - () Where multiple page, what happens on each page
- ✓(✓) Add a new record by using the insert key.
 - ✓(✓) The insert key clears all blocks of the entry screen.
 - ✓(✓) A new record can be successfully added and committed.
- ✓(✓) Add a new record by scrolling past the last entry using the down arrow key.
 - ✓() Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - ✓() All fields of the transaction blank when the end of the string of retrieved records is reached.
 - ✓(✓) A new data record can be successfully added.
- ✓(✓) Query a single record.
 - ✓(✓) A single record is retrieved.
 - ✓(✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - ✓(✓) The second query successfully retrieves the desired data.
- ✓(✓) Query multiple records using wild cards.
 - ✓(✓) Pressing F12 clears the screen and makes the transaction

Block: LEMIS_STATUS
Title: Requirement Status

Block Test Results Expected

- ✓ (✓) ready for a query.
- ✓ (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- ✓ (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- ✓ (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.

7/1A ✓ () Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- ✓ () Change each field individually, committing after each change.
- () Change each field, as per the field test plan, for both valid and invalid values.

6/5 ✓ () Delete an existing record.

- ✓ () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date-updated, owner role, and any other autogenerated fields.

() All data fields are correctly stored in the database.

Field Name: TYPE_STATUS

Field Description

Block: LEMIS_STATUS
Title: Requirement Status

Field Name: TYPE_STATUS

Field Test Case

- ✓ (✓) Accept the default.
- ✓ (✓) Override the default with a valid entry value.
- N/A ✓ (✓) Override the default with an invalid data value.
- N/A ✓ (✓) Begin field with a space. *won't handle - must be Lcr*
- ✓ (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - ✓ (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ✓ (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- N/A ✓ (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- ✓ (✓) Display table/block help. Verify that the narrative adequately describes the purpose and functions of the table/block in end-user
 - ✓ () Table/block help describes:
 - ✓ () Purpose of the table/block
 - ✓ () Functions performed by the table/block
- ✓ (✓) Add a new record by using the insert key.
 - ✓ (✓) The insert key clears all fields of the entry block.
 - ✓ (✓) A new record can be successfully added and committed.
- ✓ (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - ✓ (✓) All fields of the table/block blank when the end of the string of retrieved records is reached.
 - ✓ () A new data record can be successfully added.
- ✓ () Query a single record.

Dul

Block: LEMIS_STATUS
Title: Requirement Status

Field Name: TYPE STATUS

Field Test Results Expected

- () A single record is retrieved.
- () Pressing SHIFT/F12 clears the record and permits entering a second query.
- () The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
- () Pressing F12 clears the block and makes the block ready for a query.
- () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- ✓ () Change each field in an existing record. Verify that the change
 - ✓ () Change each field individually, committing after each change.
 - ~~NA~~ () Change each field, as per the field test plan, for both valid and invalid values.
- ✓ () Delete an existing record.
 - ✓ () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- ~~NA~~ () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- ✓ () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - ✓ () Field help describes:
 - ✓ () Purpose of the field
 - ✓ () LOV reference if available
 - ✓ () Valid range of values where present

not all changeable

W.P. Del... H... 10/1/81

Block: LEMIS_STATUS
Title: Requirement Status

Field Name: TYPE STATUS

Field Test Results Expected

- ✓(✓) Required indicator when field is NOT NULL
- ✓(✓) Default value when appropriate

N/A () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or

Dup

N/A () Store a series of records using the SHIFT ^{F6} [duplicate field] key to carry forward field values from the previously stored record.

omit dup
by field

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

N/A () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

N/A () Modify each default value with a valid alternative.

() Modify each default value with an invalid alternative.

mods not
allowed

() Any alphanumeric characters may be entered into the field.

✓() Attempt to enter data as lower case. ~~An error should result.~~

N/A () Delete an existing value.

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

N/A

Block: LEMIS_STATUS
Title: Requirement Status

Field Name: TYPE_STATUS

Field Test Results Expected

- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

} mods not allowed

Block: LEMIS_STATUS
Title: Requirement Status

Field Name: DATE STATUS

Field Description

Field Test Case

- ✓✓ Accept the default.
- ✓✓ Override the default with a valid entry value.
- N/A ✓ () Override the default with an invalid data value.
- ✓✓ Clear an existing field and attempt to update the record.
- ✓✓ Attempt to store an invalid date:
 - ✓✓ Enter the day as 00.
 - ✓✓ Enter the day as 32.
 - ✓✓ Enter the day as 30 in February.
 - ✓✓ Enter the day as 31 September.

Field Test Results Expected

- ✓✓ Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms. mm/00/11
- ✓✓ Field help describes:
 - ✓✓ Purpose of the field
 - ✓✓ LOV reference if available
 - ✓✓ Valid range of values where present
 - ✓✓ Required indicator when field is NOT NULL
 - ✓✓ Default value when appropriate
- ✓✓ Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- ✓✓ Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or

Block: LEMIS_STATUS
Title: Requirement Status

Field Name: DATE_STATUS

Field Test Results Expected

- ✓(✓) Store a series of records using the SHIFT/^{FL}~~F7~~ [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- ✓(✓) Modify each default value with a valid alternative.
- NA () Modify each default value with an invalid alternative.
- ✓(✓) Enter an invalid date. The invalid date should be rejected.
- ✓() Database record must contain the proper autogenerated value. Use SQL*Plus to verify contents.
- ✓(✓) Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- ✓(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- ✓(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: LEMIS_STATUS
 Title: Requirement Status

Not testable

Field Name: FLAG_TEXT

Field Description

This field is a flag that identifies when supporting text is present for a data row. The valid values are either "Y" (yes) or "N" (no). When the user desires to add text in support of a data row, the flag is set to "Y". The system then brings up the general text form and permits the user to enter text of varying types for the data row. Where text is already present, the user places the cursor on the flag field and selects to retrieve the text data. Text data may be updated. Note that updating text data causes the same hierarchical cloning that takes place elsewhere when a data row is updated. This ensures that the text data is also properly audit trail covered.

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- () Field help describes:

Block: LEMIS_STATUS
 Title: Requirement Status

Field Name: FLAG_TEXT

Field Test Results Expected

- () Purpose of the field
- () LOV reference if available
- () Valid range of values where present
- () Required indicator when field is NOT NULL
- () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.

Block: LEMIS_STATUS
Title: Requirement Status

Field Name: FLAG_TEXT

Field Test Results Expected

-
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2110U1 Requirement Definition

=====
Comment ID : 613 Priority: LOW Date: 29-NOV-92
=====

Status : ASSIGNED Date: 30-NOV-92 Class:
Type : COMMENT
Assigned To: TODD ADAMS
Entered By : TEST USER
System ID : 92000009153 Oracle ROWID: 000004EF.0012.0007
Last Error : -0
Last Msg : FRM-40350 Query caused no records to be retrieved.
Table : EVENT
Block : EVENT
Field : EVENT.IDENT_EVENT

DESCRIPTION

Item #1 reads.....should be identifier as
should readshould be identified as

FIXED
11/30/92
✓

Programmer's Bug Report

Transaction: LE2110U1 Requirement Definition

=====

Comment ID : 615 Priority: LOW Date: 29-NOV-92

=====

Status : ASSIGNED Date: 30-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 92000009153 Oracle ROWID: 00000571.0010.0007

Last Error : -0

Last Msg : FRM-40100 At first record.

Table : LEMIS_STATUS

Block : EVESTA

Field : EVESTA.TYPE_STATUS

DESCRIPTION

If record is deletable it won't delete. It will delete on screen but when you commit it states no records to be committed. Run query entries reappear.

COULDN'T DUPLICATE ERROR

11/30/92

✓

1.4 LE2110U2 PROJECT DEFINITION

LEMIS Module Test Checklist Report

Name: PROJECT DEFINITION
 Short Name: LE2110U2
 Purpose: Maintain Project Information.

Initial Test Date: 11/25/92 *LC*

Verification Date: 12/1/92 *JA* O.K. (*JEFF VALIDATED*)

Help	Disp Fkeys	Security			
		Ins	Del	Query	Upd
()	()	()	()	()	()

Block: EVENT
 Title: Project

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
TITLE_EVENT	CHAR	40	()	()	()	()	()
REV_EVENT	NUMBER	3	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: EVENT

Title: Requirement

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
TITLE_EVENT	CHAR	40	()	()	()	()	()
REV_EVENT	NUMBER	3	()	()	()	()	()

Block: EVENT_NEST

Title: Responsible Requirement

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: LEMIS_STATUS

Title: Project Status

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_STATUS	CHAR	4	()	()	()	()	()
DATE_STATUS	DATE	8	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: TYPE_LEMIS_STATUS

Title: Event Status Types

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

This module is used to define a Project, including:

- 1) Project definition
- 2) Project status maintenance

Module Description

A project is associated, not subordinate, to a TPA submilestone. A project may be associated with multiple submilestones and a many-many relationship exists.

Text may be associated with a project.

A project will be linked to the SITE level location with a relationship of REL. This will default and will not be displayed on the screen.

A project is associated with organizations and people:

1. Organizations. Use LOV link to organization list. Multiple organizations may be associated with specific relationship codes.
2. Persons. Use literal values. Multiple persons can be associated with specific relationship codes.

A project is NOT associated with methods and procedures.

The project status will default to CR (create) for the first entry. The default date will be the current date; this is expected to be changed by the operator. Text is associated with the project status. Multiple status lines are possible, display at least two.

Module Help

This screen supports the update and retrieval functions for the projects upon which the liquid effluent sampling activities are based. This screen is used to define the projects and requirements upon which the project depends. The status of a project may also be updated from this screen.

To store a new project:

1. Enter the project identifier. This identifier may be up to 20 characters in length. The identifier must be unique or an error will occur. The field is required.
2. TAB and enter the title of the project. The title may be from one to 40 characters in length and is required.
3. TAB to the REV field. This is the revision identifier of the project. The field is required. A default value of zero is displayed. To accept the default, press the ENTER key. To enter a different revision level, key the revision number over the default and press ENTER. (A number is required.)

The Text? flag indicates whether free-form text is associated with the requirement. If the flag is set to 'N', no text is present. To add text, press the TEXT key (F7). A popup screen will appear that will permit you to enter as much text as desired. Several different types

Module Help

of text may be available. Once the text has been stored, the text flag will be set to 'Y'. Additional text processing assistance can be obtained by pressing CTRL/K while the text popup window is being displayed.

5. Press the COMMIT key (F10) to store the data in the database.

To update an existing project:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the project to be updated. If you only know part of the identifier, enter that part followed by a percent sign (%). This will cause those projects that begin with the partial identifier to be queried.
3. Press the RUN QY key (Shift/F12). The project will be displayed on the screen; if more than one project was queried, the first one matching the query will be displayed. To locate the desired project:
 - a. Press the down arrow key until the desired project is displayed.
4. Press the TAB key until the cursor is at the field to be modified.
5. Enter the new data in the field and press ENTER.
6. If more than one field is to be updated, repeat the above process for each field.
7. Press the COMMIT key (F10) to store the changes in the database.

If the text associated with the project is to be modified:

1. Press the TEXT key (F7). The default description text will be displayed. If you wish to modify another text type:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type will be displayed.
2. Press the PAGE DOWN key to move the cursor to the text block.
3. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
4. Press the COMMIT key (F10) to update the database.

Projects may not be deleted if subordinate QAPPs are present. To delete a project:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the project to be deleted.
3. Press the RUN QY key (Shift/F12). The project will be displayed on the screen.

Module Help

-
4. Press the DELETE grey key. The screen should clear.
 5. Press the COMMIT key (F10) to update the database.
-

Each project may be assigned to a requirement. For example, a project is assigned to a TPA submilestone.

To associate a project with its responsible requirement:

1. Press the PAGE DOWN key to move the cursor to the REQ ID field of the responsible requirement block.
2. Enter the identifier of the requirement, if known. If not known, press LOV (F9) to display a list of available requirements:
 - a. Press the down arrow key until the cursor points to the desired requirement.
 - b. Press the ENTER key to select the requirement. The requirement and its information will be displayed on the screen.
3. Press the COMMIT key (F10) to store the association of the project to its responsible requirement in the database.

To remove the association between a project and its requirement:

1. Press the PAGE DOWN key to move the cursor to the REQ ID field of the responsible requirement block.
 2. If the requirement identifier displayed is the one whose relationship is to be deleted, press the DELETE key.
 3. Press the COMMIT key (F10) to complete the update of the database.
-

The status of each project is tracked in the Project Status block. Here you may identify the different types of status conditions and the dates on which the condition occurs. Unlimited text may be added to clarify the status condition. To add a new status condition:

1. Press the PAGE DOWN key to move the cursor to the first status line.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Press LOV to display the available valid status condition codes.
4. Use the up and down arrow keys to select the desired status condition.
5. Press the COMMIT (F10) key to move the status to the screen.
6. To add more statuses, press the down arrow key to the next blank space and repeat the selection process.

If all of the spaces are filled and more statuses are to be added, press the down arrow key and the already entered records will scroll upward, leaving space for an additional record to be added.

Module Help

7. Press the COMMIT (F10) key to store the selected status conditions in the database as associated with the project.

To store text expanding the description of a status condition:

1. Use the up and down arrow keys to move the cursor to the status condition line where text is to be stored.
 2. Press the TEXT key (F7). The text popup window will be displayed with cursor at the text type code. If a text type other than that displayed is desired:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type, if any, will be displayed.
 3. Press the PAGE DOWN key to move the cursor to the text block.
 4. If no text is present, enter the desired text. If text is present:
 - a. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
 - b. To add new text at the end, press the down arrow key until a blank line appears. Enter the text.
 5. Press the COMMIT key (F10) to store the text in the database.
 6. Press the ENTER key until the popup window disappears and the requirement screen reappears.
 7. If more text is to be added to other status condition entries, use the up and down arrow keys to move the cursor to the desired entry and repeat the above steps.
-

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed.

Module Test Plan

The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
- Developer _____ Date Complete _____
- (✓) 8 - Secondary testing complete, no problems found.

Tester <u>OC</u>	Date Complete <u>11/25/92</u>
------------------	-------------------------------

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off

Module Test Plan

the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Test status conditions are:

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.

Project Manager ZSJ Date Complete 12/1/92

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu

System Architect SA Date Complete 12/1/92

Module Test Case

(1) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

(1) Duplicate row insertion attempts result in error message

COM00102 [Duplicate value, enter new value or F1 for help]

(✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

(✓) Transaction help describes:

(✓) Purpose of the transaction

(✓) Functions performed by the transaction

(✓) Where multiple page, what happens on each page

(✓) Add a new record by using the insert key.

(✓) The insert key clears all blocks of the entry screen.

(✓) A new record can be successfully added and committed.

606 (✓) Add a new record by scrolling past the last entry using the down arrow key.

() Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.

() All fields of the transaction blank when the end of the string of retrieved records is reached.

(✓) A new data record can be successfully added.

(✓) Query a single record.

(✓) A single record is retrieved.

(✓) Pressing ~~SHIFT~~ F12 clears the record and permits entering a second query.

(✓) The second query successfully retrieves the desired data.

(✓) Query multiple records using wild cards.

(✓) Pressing F12 clears the screen and makes the transaction ready for a query.

(✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through

(✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.

(✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.

(✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

(✓) Change each field individually, committing after each change.

() Change each field, as per the field test plan, for both valid and invalid values.

- (✓) Delete an existing record.
 - () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - () All data fields are correctly stored in the database.

Block: EVENT
Title: Project

Block Description

The event table is a multi-use table. It identifies those activities that are defined to guide and control the liquid effluent program.

An event, as currently (10/9/92) defined may be:

The TPA itself.

A TPA milestone (milestone 17 for LEMIS and will be known as a project.

A submilestone within milestone 17 (will be known as a task).

The QAPP defining LEMIS.

The SAPs defining the waste stream processing.

The SAFs defining the actual sampling activities.

In the hierarchy of things:

The TPA is the top level.

TPA Milestone 17 is dependent on the TPA.

TPA submilestones of 17 depend on Milestone 17.

Project

On HEIS, a project is a funded program, such as 300-Area cleanup. For release one of LEMIS, all sampling will be part of the same

Block: EVENT
Title: Project

Block Description

project, which is Liquid Effluent Sampling, or Tri-Party Agreement milestone number 17.

A project is very tightly coupled to its QAPjP which describes many aspects of how the project will be done.

LEMIS will record the project for each sample that is recorded. During release one, project book keeping may be invisible to the user, and may be done automatically by LEMIS, since only one project will exist.

A project will have the following attributes:
Project name: "TPA Liquid Effluent Sampling " is suggested.
User Id: M-17-00 is suggested.

A project is related to:
One QAPjP at a time.
Many tasks.
Perhaps a high level location, such as Hanford.

QAPP/QAPjP (Quality Assurance Project Plan)

The QAPP, covering all 33 major streams, is dependent on TPA Milestone 17 (the project).

For Release One, the project will be related to one QAPjP at a time. However, a QAPjP may be revised and changed many times, so a program (project) may be related to many different QAPjPs (or at least many different versions) over the history of the project.

A QAPjP will have the following characteristics:
ID Number: The document number
Title: Title of the document
Rev Number: Revision number of the document

A QAPjP is related to one and only one project.

LEMIS will know the QAPjP title and rev number for each sample that is recorded.

Task

Each project consists of many tasks. On HEIS, a task is typically a number from a project document that itemizes tasks to be performed. For LEMIS, a task is typically a stream specific milestone, such as

Block: EVENT
Title: Project

Block Description

M-17-XX.

The task stream (which is a high level location), and Sampling Analysis Plan (SAP) are very tightly coupled. A task corresponds to one and only one stream and one and only one SAP at a given time. This allows the user to select a task by either the milestone number or the stream name. Only one SAP describes a task at a time, however the SAP may be revised and changed throughout a program. Also, only one SAF will be active for a task at a time.

For each sample taken, LEMIS will record the corresponding task, the applicable SAP, and the applicable SAF.

A task has the following attributes:

Task name: Descriptive title.
User Id: M-17-XX is suggested.

A task is related to:

Exactly one project
Exactly one stream location (a high level location)
One or many sampling locations (the actual sampling locations)
Exactly one SAP at a time
Zero or or active SAFs at a time (but many SAFs may apply over time).

LEMIS will know the task, SAP name and revision, and SAF number for each sample that is recorded.

SAP (Sample Analysis Plan)

The SAPs, one for each waste stream, are dependent on the QAPP. Each SAP is also related to the submilestone (task). At any given point in time, a SAP and task are on a one to one relationship. It is possible for there to be multiple SAPs over time related to a single task.

The SAFs occur as part of the sampling defined by the SAP. Several SAFs may occur for a SAP but it is unlikely that more than one will be "active" at one time. This does not include those pre-scheduled for future sampling. Each SAF has a subordinate Sample-Authorization table entry containing specific information related to the SAF.

Block Test Case

Block: EVENT
 Title: Project

Block Test Results Expected

Field Name: IDENT_EVENT

Field Description

This field uniquely identifies the event.

Field Test Case

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or

Block: EVENT
Title: Project

Field Name: IDENT_EVENT

Field Test Results Expected

- () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- 1/A () Store a series of records using the SHIFT/^{sc}F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- (/ Any alphanumeric characters may be entered into the field.
- (/ Attempt to enter data as lower case. ~~An error should result.~~
- (/ Delete an existing value.
- (/ For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (/ Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (/ Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT
Title: Project

Field Name: TITLE_EVENT

Field Description

Field Test Case

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key

Block: EVENT
Title: Project

Field Name: TITLE_EVENT

Field Test Results Expected

to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

(✓) Any alphanumeric characters may be entered into the field.

(N) Attempt to enter data as lower case. ~~An error should result.~~

(✓) Delete an existing value.

(✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Field Name: REV EVENT

Field Description

This field represents a revision to the event.

Field Test Case

Field Test Results Expected

No testing for REV

Block: EVENT
 Title: Project

Field Name: FLAG_TEXT

Field Description

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.

Block: EVENT
Title: Project

Field Name: FLAG_TEXT

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT_NEST
 Title: Responsible Requirement

Block Description

The table links events to each other. The most common linkage will be that of a series of events that are grouped together under another event definition. Another option is to link a series of events that support the same repetitive set of sampling.

Block Test Case

Block Test Results Expected

*no testing for
 Res 200*

Block: LEMIS_STATUS
 Title: Project Status

Block Description

Block Test Case

Block Test Results Expected

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- () Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

NA
 - () Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - () Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction

Should be block help

Block: LEMIS_STATUS
 Title: Project Status

Block Test Results Expected

- ready for a query.
- (/) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- (/) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- (/) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- (/) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- (/) Change each field individually, committing after each change.
- () Change each field, as per the field test plan, for both valid and invalid values.
- (/) Delete an existing record.
- () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- () All data fields are correctly stored in the database.

Field Name: TYPE_STATUS

Field Description

Block: LEMIS_STATUS
Title: Project Status

Field Name: TYPE_STATUS

Field Test Case

- ✓(✓) Accept the default.
- () Override the default with a valid entry value. *no update capability*
- NA* () Override the default with an invalid data value. *]*
- NA* () Begin field with a space.
- ✓(✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - ✓(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ✓(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

- ✓(✓) Clear an existing field and attempt to update the record. *N/A*

Field Test Results Expected

- ✓(✓) Display table/block help. Verify that the narrative adequately describes the purpose and functions of the table/block in end-user
 - ✓(✓) Table/block help describes:
 - ✓(✓) Purpose of the table/block
 - ✓(✓) Functions performed by the table/block
- ✓(✓) Add a new record by using the insert key.
 - ✓(✓) The insert key clears all fields of the entry block.
 - ✓(✓) A new record can be successfully added and committed.
- ✓(✓) Add a new record by scrolling past the last entry using the down arrow key.
 - ✓(✓) All fields of the table/block blank when the end of the string of retrieved records is reached.
 - ✓(✓) A new data record can be successfully added.
- ✓(✓) Query a single record.

Field help?

Block: LEMIS_STATUS
Title: Project Status

Field Name: TYPE_STATUS

Field Test Results Expected

- ✓(✓) A single record is retrieved.
- ✓(✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
- ✓(✓) The second query successfully retrieves the desired data.
- ✓(✓) Query multiple records using wild cards.
- ✓(✓) Pressing F12 clears the block and makes the block ready for a query.
- ✓(✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- ✓(✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- ✓(✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- ✓(✓) Change each field in an existing record. Verify that the change
- ✓(✓) Change each field individually, committing after each change.
- () Change each field, as per the field test plan, for both valid and invalid values.
- ✓(✓) Delete an existing record.
- (✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- ✓(✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- ✓(✓) Field help describes:
 - ✓(✓) Purpose of the field
 - ✓(✓) LOV reference if available
 - ✓(✓) Valid range of values where present

*can Add new,
but not Change
existing*

Block: LEMIS_STATUS
 Title: Project Status

Field Name: TYPE_STATUS

Field Test Results Expected

- ✓(✓) Required indicator when field is NOT NULL
- ✓(✓) Default value when appropriate

() Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

() Change the field, committing after each change.

() Change the field where a range of valid values exists:

() Value below valid range should display message COM00101 or COM00103 (if no LOV).

() Value above valid range should display message COM00101 or

() Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

() Modify each default value with a valid alternative.

() Modify each default value with an invalid alternative.

() Any alphanumeric characters may be entered into the field.

() Attempt to enter data as lower case. ~~An error should result?~~

() Delete an existing value.

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

Block: LEMIS_STATUS
Title: Project Status

Field Name: TYPE_STATUS

Field Test Results Expected

- ✓ (1) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- ✓ Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: LEMIS_STATUS
 Title: Project Status

Field Name: DATE_STATUS

Field Description

Field Test Case

- ☒ Accept the default.
- ☒ Override the default with a valid entry value.
- ☒ Override the default with an invalid data value.
- ☒ Clear an existing field and attempt to update the record.
- ☒ Attempt to store an invalid date:
 - ☒ Enter the day as 00.
 - ☒ Enter the day as 32.
 - ☒ Enter the day as 30 in February.
 - ☒ Enter the day as 31 September.

Field Test Results Expected

- ☒ Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- ☒ Field help describes:
 - ☐ Purpose of the field
 - ☐ LOV reference if available
 - ☐ Valid range of values where present
 - ☐ Required indicator when field is NOT NULL
 - ☐ Default value when appropriate
- ☒ Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- ☒ Change the field, committing after each change.
- ☐ Change the field where a range of valid values exists:
 - ☐ Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - ☐ Value above valid range should display message COM00101 or

Block: LEMIS_STATUS
 Title: Project Status

Field Name: DATE_STATUS

Field Test Results Expected

- (✓) Store a series of records using the SHIFT/~~P~~^{file} [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- (✓) Modify each default value with a valid alternative.
- (✓) Modify each default value with an invalid alternative.
- (✓) Enter an invalid date. The invalid date should be rejected.
- () Database record must contain the proper autogenerated value. Use SQL*Plus to verify contents.
- (✓) Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Programmer's Bug Report

Transaction: LE2110U2 Project Definition

```
=====
Comment ID : 581    Priority: LOW          Date: 24-NOV-92
=====
Status      : ASSIGNED    Date: 26-NOV-92    Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : LARRY TOWNER
System ID   : 92000008947      Oracle ROWID:
Last Error  : -0
Last Msg    : FRM-40407 Transaction complete -- posted records comm
Table       : EVENT_NEST
Block       : EVENST
Field       : EVENST.DSP_TYPE_EVENT
```

DESCRIPTION

The requirement type, as shown, is one character when it should be two. But why are we displaying it at all? It doesn't seem to serve any purpose.

NOT PROB
11/28/92
✓

Programmer's Bug Report

Transaction: LE2110U2 Project Definition

=====

Comment ID : 580 Priority: LOW Date: 24-NOV-92

=====

Status : ASSIGNED Date: 26-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : LARRY TOWNER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40202 Field must be entered.

Table : EVENT_NEST

Block : EVENST

Field : EVENST.DSP_IDENT_EVENT

DESCRIPTION

When page down to the related requirement block, the cursor starts at the title instead of the requirement id.

NOT PROB
11/29/92
✓

Programmer's Bug Report

Transaction: LE2110U2 Project Definition

=====

Comment ID : 607 Priority: LOW Date: 25-NOV-92

=====

Status : DOCCHNG Date: 26-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By : TEST USER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40209 Field must be of form MM/DD/YY.

Table : LEMIS_STATUS

Block : EVESTA

Field : EVESTA.TYPE_STATUS

DESCRIPTION

help states date format as dd-~~MMM~~-yy but hint text and field accept
mm-dd-yy only.

✓

11/29

1.5 LE2110U3 QAPP DEFINITION

LEMIS Module Test Checklist Report

Name: QAPP DEFINITION
 Short Name: LE2110U3
 Purpose: Maintain QAPP Information.

Initial Test Date: 11/29/92 lc

Verification Date: 12/1/92 lc

	Disp	Security			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: EVENT
 Title: Project

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
TITLE_EVENT	CHAR	40	()	()	()	()	()
REV_EVENT	NUMBER	3	()	()	()	()	()

Block: EVENT

Title: QAPP Definition

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
TITLE_EVENT	CHAR	40	()	()	()	()	()
REV_EVENT	NUMBER	3	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: EVENT_METHOD_J

Title: Methods In QAPP

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: EVENT_NEST

Title: Responsible Project

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: LEMIS_STATUS

Title: QAPP Status

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_STATUS	CHAR	4	()	()	()	()	()
DATE_STATUS	DATE	8	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: METHOD

Title: Laboratory Methods

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	30	()	()	()	()	()
TITLE_METHOD	CHAR	40	()	()	()	()	()

Block: TYPE_LEMIS_STATUS

Title: Event Status Types

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

This module is used to define a QAPP, including:

- 1) QAPP definition
- 2) QAPP Status maintenance
- 4) Analytical Methods linkage

A QAPP is SUBORDINATE to a project. A project must exist before a QAPP may be defined.

Text may be associated with a QAPP.

Person and organization association to a QAPP:

1. Organizations. Use LOV link to organization list. Multiple organizations may be associated with specific relationship codes.
2. Persons. Use literal values. Multiple persons can be associated with specific relationship codes.

Methods are assigned to a QAPP as a shopping list of all applicable methods, including alternates, that may be used by any subsequent SAP as the basis for sampling. The shopping list should display all analysis classes in the reference library. Selecting an analysis class will display the methods associated with the class.

The user must be able to select one or more items from the method list to be attached to the QAPP. This selection should allow multiple methods to be selected during the same popup display time.

Module Description

This shopping list will be the basis for the SAP method selection.

Procedures are not associated with the QAPP.

QAPP status information will default to CR (create), with a default date of the current date (the date will likely be changed by the operator). Text may be associated with the status. Multiple status lines are possible, display at least two.

A project will be linked to the SITE level location with a relationship of REL. This will default and will not be displayed on the screen.

Module Help

This screen supports the update and retrieval functions for the QAPPs upon which the liquid effluent sampling activities are based. This screen is used to define the QAPPs and the projects to which a QAPP is dependent. The status of a QAPP may also be updated from this screen.

To store a new QAPP:

1. Enter the QAPP identifier. This identifier may be up to 20 characters in length. The identifier is usually the QAPP document identifier. The identifier value must be unique or an error will occur. The field is required.
2. TAB and enter the title of the QAPP. The title may be from one to 40 characters in length and is required.
3. TAB to the REV field. This is the revision identifier of the requirement. The field is required. A default value of zero is displayed. To accept the default, press the ENTER key. To enter a different revision level, key the revision number over the default and press ENTER. (A number is required.)

The Text? flag indicates whether free-form text is associated with the requirement. If the flag is set to 'N', no text is present. To add text, press the TEXT key (F7). A popup screen will appear that will permit you to enter as much text as desired. Several different types of text may be available. Once the text has been stored, the text flag will be set to 'Y'. Additional text processing assistance can be obtained by pressing CTRL/K while the text popup window is being displayed.

5. Press the COMMIT key (F10) to store the data in the database.

To update an existing QAPP:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the QAPP to be updated.

Module Help

-
3. Press the RUN QY key (Shift/F12). The QAPP will be displayed on the screen.
 4. Press the TAB key until the cursor is at the field to be modified.
 5. Enter the new data in the field and press ENTER.
 6. If more than one field is to be updated, repeat the above process for each field.
 7. Press the COMMIT key (F10) to store the changes in the database.

If the text associated with the QAPP is to be modified:

1. Press the TEXT key (F7). The default description text will be displayed. If you wish to modify another text type:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type will be displayed.
2. Press the PAGE DOWN key to move the cursor to the text block.
3. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
4. Press the COMMIT key (F10) to update the database.

QAPPs may not be deleted if subordinate SAPs are present. To delete a QAPP:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the QAPP to be deleted.
3. Press the RUN QY key (Shift/F12). The QAPP will be displayed on the screen.
4. Press the DELETE grey key. The screen should clear.
5. Press the COMMIT key (F10) to update the database.

Each QAPP may be assigned to a project. For example, QAPP WHC-SD-WM-QAPP-011 could be assigned to project TPA-17.011-1.

To associate a QAPP with its project:

1. Press the PAGE DOWN key to move the cursor to the PROJ ID field of the Responsible Project block.
2. Enter the identifier of the project, if known. If not known, press LOV (F9) to display a list of available project:
 - a. Press the down arrow key until the cursor points to the desired project.
 - b. Press the ENTER key to select the project.
The project and its information will be displayed

Module Help

on the screen.

3. Press the COMMIT key (F10) to store the association of the QAPP to its project in the database.

To remove the association between a QAPP and its project:

1. Press the PAGE DOWN key to move the cursor to the PROJ ID field of the Responsible Project block.
2. If the project identifier displayed is the one whose relationship is to be deleted, press the DELETE key.
3. Press the COMMIT key (F10) to complete the update of the database.

The status of each QAPP is tracked in the QAPP Status block. Here you may identify the different types of status conditions and the dates on which the condition occurs. Unlimited text may be added to clarify the status condition. To add a new status condition:

1. Press the PAGE DOWN key to move the cursor to the first status line.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Press LOV to display the available valid status condition codes.
4. Use the up and down arrow keys to select the desired status condition.
5. Press the COMMIT (F10) key to move the status to the screen.
6. To add more statuses, press the down arrow key to the next blank space and repeat the selection process.

If all of the spaces are filled and more statuses are to be added, press the down arrow key and the already entered records will scroll upward, leaving space for an additional record to be added.

7. Press the COMMIT (F10) key to store the selected status conditions in the database as associated with the QAPP.

To store text expanding the description of a status condition:

1. Use the up and down arrow keys to move the cursor to the status condition line where text is to be stored.
2. Press the TEXT key (F7). The text popup window will be displayed with cursor at the text type code. If a text type other than that displayed is desired:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type, if any, will be displayed.
3. Press the PAGE DOWN key to move the cursor to the text block.
4. If no text is present, enter the desired text. If text is

Module Help

present:

- a. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
- b. To add new text at the end, press the down arrow key until a blank line appears. Enter the text.
5. Press the COMMIT key (F10) to store the text in the database.
6. Press the ENTER key until the popup window disappears and the requirement screen reappears.
7. If more text is to be added to other status condition entries, use the up and down arrow keys to move the cursor to the desired entry and repeat the above steps.

Each QAPP identifies special analysis classes, analysis classes, and analytical methods that are to be used to analyze the constituents at the waste locations identified by the QAPP. One or more items may be defined. NOTE: Items NOT identified to the QAPP cannot be selected for subordinate SAPs and SAFs.

To identify an analysis item being used by the QAPP:

1. Use the PAGE DOWN key to move the cursor to the METHOD ID field of the Methods In QAPP block.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Press LOV (F9) to display a list of methods, analysis classes, and special analysis classes.
4. Select the desired item by one of the following options:
 - a. Use the up and down arrow keys to move the cursor to the item to be selected. Press 'X' to make the selection.
 - b. Press ENTER QY (F12), TAB until the cursor is at the TYPE field, enter the desired type code, and press RUN QY (SHIFT/F12). This will reduce the display to only the items of that type. Press 'X' to select those items desired.
 - c. Press SHIFT/F5 to select all items in the list. NOTE: This option can be used when most of the items in the list will be selected. After loading the entire list, selectively remove those not desired.

A combination of the above can be used. The query (b) can be used to reduce the list and then SHIFT/F5 to select those remaining. The reduced list can be picked from selectively by using the individual selection option (a).

NOTE: The normal items to select are analysis classes (type AC) and special analysis classes (type AP). The special analysis classes are made up of an analysis class with the addition/subtraction of specific constituents. The methods associated with an analysis class may also be individually selected.

Module Help

5. Use the up and down arrow keys to move the cursor to other items to be selected, pressing 'X' each time to make the selection.
6. Press the COMMIT key (F10) to register the selections.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Module Test Plan

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
Developer _____ Date Complete _____
- ()/8 - Secondary testing complete, no problems found.

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

Module Test Plan

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

() Reviews tests results.

() Reviews problem reports to ensure all corrections done.

() Updates NOTES text to reflect testing results.

Project Manager Scj Date Complete 12/1/92

() 20 - The transaction is placed into the production environment.

() Software files in production directories.

() DMR 550 updated to add transaction.

() DMR 900 updated to add transaction.

() Transaction added to production usage menu.

System Architect AF Date Complete 12/1/92

Module Test Case

6/17
+ 12/2/92
() Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

() Duplicate row insertion attempts result in error message
COM00102 [Duplicate value, enter new value or F1 for help]

(✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

(✓) Transaction help describes:

(✓) Purpose of the transaction

() Functions performed by the transaction

() Where multiple page, what happens on each page

6/18
(✓) Add a new record by using the insert key.

(✓) The insert key clears all blocks of the entry screen.

(✓) A new record can be successfully added and committed.

(✓) Add a new record by scrolling past the last entry using the down arrow key.

() Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.

() All fields of the transaction blank when the end of the string of retrieved records is reached.

(✓) A new data record can be successfully added.

(✓) Query a single record.

(✓) A single record is retrieved.

() Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.

() The second query successfully retrieves the desired data.

(X) Query multiple records using wild cards.

(X) Pressing F12 clears the screen and makes the transaction ready for a query.

() A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.

(X) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.

(X) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through

(X) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

(X) Change each field individually, committing after each change.

() Change each field, as per the field test plan, for both valid and invalid values.

(X) Delete an existing record.

() The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.

() All data fields are correctly stored in the database.

Block: EVENT

Title: QAPP Definition

Block Description

The event table is a multi-use table. It identifies those activities

*unique fields
should not change*

Block: EVENT
Title: QAPP Definition

Block Description

that are defined to guide and control the liquid effluent program.

An event, as currently (10/9/92) defined may be:

The TPA itself.

A TPA milestone (milestone 17 for LEMIS and will be known as a project).

A submilestone within milestone 17 (will be known as a task).

The QAPP defining LEMIS.

The SAPs defining the waste stream processing.

The SAFs defining the actual sampling activities.

In the hierarchy of things:

The TPA is the top level.

TPA Milestone 17 is dependent on the TPA.

TPA submilestones of 17 depend on Milestone 17.

Project

On HEIS, a project is a funded program, such as 300-Area cleanup. For release one of LEMIS, all sampling will be part of the same project, which is Liquid Effluent Sampling, or Tri-Party Agreement milestone number 17.

A project is very tightly coupled to its QAPjP which describes many aspects of how the project will be done.

LEMIS will record the project for each sample that is recorded. During release one, project book keeping may be invisible to the user, and may be done automatically by LEMIS, since only one project will exist.

A project will have the following attributes:

Project name: "TPA Liquid Effluent Sampling " is suggested.

User Id: M-17-00 is suggested.

A project is related to:

One QAPjP at a time.

Many tasks.

Perhaps a high level location, such as Hanford.

Block: EVENT
 Title: QAPP Definition

Block Description

QAPP/QAPjP (Quality Assurance Project Plan)

The QAPP, covering all 33 major streams, is dependent on TPA Milestone 17 (the project).

For Release One, the project will be related to one QAPjP at a time. However, a QAPjP may be revised and changed many times, so a program (project) may be related to many different QAPjPs (or at least many different versions) over the history of the project.

A QAPjP will have the following characteristics:

ID Number: The document number
 Title: Title of the document
 Rev Number: Revision number of the document

A QAPjP is related to one and only one project.

LEMIS will know the QAPjP title and rev number for each sample that is recorded.

Task

Each project consists of many tasks. On HEIS, a task is typically a number from a project document that itemizes tasks to be performed. For LEMIS, a task is typically a stream specific milestone, such as M-17-XX.

The task stream (which is a high level location), and Sampling Analysis Plan (SAP) are very tightly coupled. A task corresponds to one and only one stream and one and only one SAP at a given time. This allows the user to select a task by either the milestone number or the stream name. Only one SAP describes a task at a time, however the SAP may be revised and changed throughout a program. Also, only one SAF will be active for a task at a time.

For each sample taken, LEMIS will record the corresponding task, the applicable SAP, and the applicable SAF.

A task has the following attributes:

Task name: Descriptive title.
 User Id: M-17-XX is suggested.

A task is related to:

Exactly one project
 Exactly one stream location (a high level location)
 One or many sampling locations (the actual sampling locations)

Block: EVENT
Title: QAPP Definition

Block Description

Exactly one SAP at a time
Zero or more active SAFs at a time (but many SAFs may apply over time).

LEMIS will know the task, SAP name and revision, and SAF number for each sample that is recorded.

SAP (Sample Analysis Plan)

The SAPs, one for each waste stream, are dependent on the QAPP. Each SAP is also related to the submilestone (task). At any given point in time, a SAP and task are on a one to one relationship. It is possible for there to be multiple SAPs over time related to a single task.

The SAFs occur as part of the sampling defined by the SAP. Several SAFs may occur for a SAP but it is unlikely that more than one will be "active" at one time. This does not include those pre-scheduled for future sampling. Each SAF has a subordinate Sample-Authorization table entry containing specific information related to the SAF.

Block Test Case

Block Test Results Expected

Field Name: IDENT_EVENT

Field Description

This field uniquely identifies the event.

Field Test Case

✓ Begin field with a space. *unique should not update*

619 () Enter data beyond length of field.
() If autoskip is in effect, cursor will skip to the next field.
() If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field. *gk B*

✓ Set keyboard on lower case and enter data. Data should be

Block: EVENT
 Title: QAPP Definition

Field Name: IDENT_EVENT

Field Test Case

converted to upper case.

- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

- () Field help describes:
 - (✓) Purpose of the field
 - () LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - () Default value when appropriate

- N) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

*should
not update*

- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

FL6

- n/A () Store a series of records using the SHIFT/PT [duplicate field] key to carry forward field values from the previously stored record.

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

- () Use SQL*Plus to verify all data fields are processed correctly.

Block: EVENT
Title: QAPP Definition

Field Name: IDENT_EVENT

Field Test Results Expected

- () All data fields are correctly stored in the database.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. An error should result.
- (✓) Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT
Title: QAPP Definition

Field Name: TITLE_EVENT

Field Description

Field Test Case

- (✓) Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

error recorded

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - () LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - () Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- (✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key

Block: EVENT
Title: QAPP Definition

Field Name: TITLE_EVENT

Field Test Results Expected

- to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~
- (✓) Delete an existing value.
 - (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Field Name: REV_EVENT

Field Description

This field represents a revision to the event.

Field Test Case

Field Test Results Expected

Block: EVENT
 Title: QAPP Definition

Field Name: FLAG_TEXT

Field Description

Field Test Case

- Not testable*
- () Accept the default.
 - () Override the default with a valid entry value.
 - () Override the default with an invalid data value.
 - () Begin field with a space.
 - () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
 - () Set keyboard on lower case and enter data. Data should be converted to upper case.
 - () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.

Block: EVENT
 Title: QAPP Definition

Field Name: FLAG_TEXT

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT_METHOD_J
 Title: Methods In QAPP

Block Description

Block Test Case

Block Test Results Expected

✓ Display table/block help. Verify that the narrative adequately describes the purpose and functions of the table/block in end-user terms.

✓ Table/block help describes:

- (✓) Purpose of the table/block
- (✓) Functions performed by the table/block

✓ () Add a new record by using the insert key.

- () The insert key clears all fields of the entry block.
- () A new record can be successfully added and committed.

✓ () Add a new record by scrolling past the last entry using the down arrow key.

- () All fields of the table/block blank when the end of the string of retrieved records is reached.
- () A new data record can be successfully added.

✓ () Query a single record.

- (✓) A single record is retrieved.
- (✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
- (✓) The second query successfully retrieves the desired data.

() Query multiple records using wild cards.

- (✓) Pressing F12 clears the block and makes the block ready for a query.
- (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
- (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through

✓ () Change each field in an existing record. Verify that the change

Block: EVENT_METHOD_J
Title: Methods In QAPP

Block Test Results Expected

occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

11/11/85 () Change each field individually, committing after each change.
() Change each field, as per the field test plan, for both valid and invalid values.

321 () Delete an existing record.

() The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.

() All data fields are correctly stored in the database.

Block: EVENT_NEST
Title: Responsible Project

Block Description

The table links events to each other. The most common linkage will be that of a series of events that are grouped together under another event definition. Another option is to link a series of events that support the same repetitive set of sampling.

Block Test Case

Block Test Results Expected

Block: LEMIS STATUS
 Title: QAPP Status

Block Description

Block Test Case

Block Test Results Expected

(✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

(✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]

(✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

(✓) Transaction help describes:

- (✓) Purpose of the transaction
- (✓) Functions performed by the transaction
- () Where multiple page, what happens on each page

(✓) Add a new record by using the insert key.

(✓) The insert key clears all blocks of the entry screen.

(✓) A new record can be successfully added and committed.

(✓) Add a new record by scrolling past the last entry using the down arrow key.

(✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.

(✓) All fields of the transaction blank when the end of the string of retrieved records is reached.

(✓) A new data record can be successfully added.

(✓) Query a single record.

(✓) A single record is retrieved.

(✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.

(✓) The second query successfully retrieves the desired data.

() Query multiple records using wild cards.

(✓) Pressing F12 clears the screen and makes the transaction

Block: LEMIS_STATUS
Title: QAPP Status

Block Test Results Expected

- ✓ ready for a query.
- ✓ (a) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- ✓ (b) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- ✓ (c) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- ✓ (d) Change each ^{updatable} field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- ✓ (e) Change each field individually, committing after each change.
- () Change each field, as per the field test plan, for both valid and invalid values.
- ✓ (f) Delete an existing record.
- () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- () All data fields are correctly stored in the database.

Field Name: TYPE_STATUS

Field Description

Block: LEMIS_STATUS
Title: QAPP Status

Field Name: TYPE_STATUS

Field Test Case

- ☒ Accept the default.
- ☒ Override the default with a valid entry value.
- n/A ☐ Override the default with an invalid data value.
- n/A ☐ Begin field with a space.
- ☒ Enter data beyond length of field.
 - ☐ If autoskip is in effect, cursor will skip to the next field.
 - ☒ If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ☒ Set keyboard on lower case and enter data. Data should be converted to upper case.
- ☒ Clear an existing field and attempt to update the record.

no
overlay
or autoskip

Field Test Results Expected

- ☐ Display table/block help. Verify that the narrative adequately describes the purpose and functions of the table/block in end-user
 - ☐ Table/block help describes:
 - ☐ Purpose of the table/block
 - ☐ Functions performed by the table/block
- ☐ Add a new record by using the insert key.
 - ☐ The insert key clears all fields of the entry block.
 - ☐ A new record can be successfully added and committed.
- ☐ Add a new record by scrolling past the last entry using the down arrow key.
 - ☐ All fields of the table/block blank when the end of the string of retrieved records is reached.
 - ☐ A new data record can be successfully added.
- ☐ Query a single record.

Due

208

Block: LEMIS_STATUS
 Title: QAPP Status

Field Name: TYPE_STATUS

Field Test Results Expected

- () A single record is retrieved.
- () Pressing SHIFT/F12 clears the record and permits entering a second query.
- () The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
- () Pressing F12 clears the block and makes the block ready for a query.
- () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.

DWR

- n/A* () Change each field in an existing record. Verify that the change
- () Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.

Not Updatable

- n/A* () Delete an existing record. *field*
- () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

- () Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date-updated, owner role, and any other autogenerated fields.

- ✓* () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

- (*✓*) Field help describes:
 - (*✓*) Purpose of the field
 - () LOV reference if available
 - (*✓*) Valid range of values where present

Block: LEMIS STATUS
 Title: QAPP Status

Field Name: TYPE STATUS

Field Test Results Expected

- ☒ Required indicator when field is NOT NULL
- ☐ Default value when appropriate

- N/A () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- N/A () Store a series of records using the SHIFT/^{FL} [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- N/A () Modify each default value with a valid alternative.
- N/A () Modify each default value with an invalid alternative.
- ☒ Any alphanumeric characters may be entered into the field.
- ☒ Attempt to enter data as lower case. ~~An error should result.~~
- N/A () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

Block: LEMIS STATUS
Title: QAPP Status

Field Name: TYPE_STATUS

Field Test Results Expected

- N/A {
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: LEMIS_STATUS
Title: QAPP Status

Field Name: DATE_STATUS

Field Description

Field Test Case

☒ Accept the default.

☒ Override the default with a valid entry value.

M/A ☐ Override the default with an invalid data value.

☒ Clear an existing field and attempt to update the record.

☒ Attempt to store an invalid date:

☒ Enter the day as 00.

☒ Enter the day as 32.

☒ Enter the day as 30 in February.

☒ Enter the day as 31 September.

Field Test Results Expected

☐ Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

OK 1/22 ☐ Field help describes:

☐ Purpose of the field

☐ LOV reference if available

☐ Valid range of values where present

☐ Required indicator when field is NOT NULL

☐ Default value when appropriate

☒ Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

☒ Change the field, committing after each change.

☒ Change the field where a range of valid values exists:

☐ Value below valid range should display message COM00101 or COM00103 (if no LOV).

☐ Value above valid range should display message COM00101 or

mm ldd/vy

Block: LEMIS_STATUS
Title: QAPP Status

Field Name: DATE_STATUS

Field Test Results Expected

- ☒ Store a series of records using the SHIFT/~~F7~~^{F6} [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- ☒ Modify each default value with a valid alternative.
- ☒ Modify each default value with an invalid alternative.
- ☒ Enter an invalid date. The invalid date should be rejected.
- ☒ () Database record must contain the proper autogenerated value. Use SQL*Plus to verify contents.
- ☒ () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- ☒ () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- ☒ () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: LEMIS STATUS
 Title: QAPP Status

*NO
 testing*

Field Name: FLAG_TEXT

Field Description

This field is a flag that identifies when supporting text is present for a data row. The valid values are either "Y" (yes) or "N" (no). When the user desires to add text in support of a data row, the flag is set to "Y". The system then brings up the general text form and permits the user to enter text of varying types for the data row. Where text is already present, the user places the cursor on the flag field and selects to retrieve the text data. Text data may be updated. Note that updating text data causes the same hierarchical cloning that takes place elsewhere when a data row is updated. This ensures that the text data is also properly audit trail covered.

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- () Field help describes:

Block: LEMIS STATUS
Title: QAPP Status

Field Name: FLAG_TEXT

Field Test Results Expected

- () Purpose of the field
- () LOV reference if available
- () Valid range of values where present
- () Required indicator when field is NOT NULL
- () Default value when appropriate

- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.

- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.

Block: LEMIS STATUS
Title: QAPP Status

Field Name: FLAG_TEXT

Field Test Results Expected

-
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2110U3 QAPP Definition

```
=====
Comment ID : 618      Priority: LOW      Date: 29-NOV-92
=====
Status      : NOTPROB      Date: 30-NOV-92      Class:
Type        : PROBLEM
Assigned To :      KEN ATKINS
Entered By  :      TEST USER
System ID   : 92000009129      Oracle ROWID: 00000753.000D.0007
Last Error  :      -0
Last Msg    : FRM-40400      Transaction complete -- 2 records posted an
Table       : EVENT_NEST
Block       : EVENST
Field       : EVENST.DSP_IDENT_EVENT
=====
```

DESCRIPTION

```
-----
When trying to insert a record in QAPP def commit was not allowed
until I paged down and selected a record from LOV. Are these two
blocks suppose to work separately or are they suppose to tie into each
other???
```

CORRECTIVE ACTION

```
-----
11/30/92 - This is the correct functionality.
```


Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2110U3 QAPP Definition

=====
Comment ID : 619 Priority: LOW Date: 29-NOV-92
=====

Status : FIXED Date: 30-NOV-92 Class:
Type : PROBLEM
Assigned To: KEN ATKINS
Entered By : TEST USER
System ID : 92000009157 Oracle ROWID: 000004EF.0015.0007
Last Error : -0
Last Msg : FRM-40400 Transaction complete -- 1 records posted an
Table : EVENT
Block : EVENT
Field : EVENT.IDENT_EVENT

DESCRIPTION

field is unique and should not update. Field autoskip does not tab nor
overlay characters.

CORRECTIVE ACTION

11/30/92 - Hint text is wrong. Fixed

ie OK

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2110U3 QAPP Definition

```
=====
Comment ID : 620    Priority: LOW                      Date: 29-NOV-92
=====
Status      : NOTPROB      Date: 30-NOV-92      Class:
Type        : PROBLEM
Assigned To :              KEN ATKINS
Entered By  :              TEST USER
System ID   : 92000009129              Oracle ROWID: 00000753.000E.0007
Last Error  :              -0
Last Msg    : FRM-40350    Query caused no records to be retrieved.
Table       : EVENT_NEST
Block       : EVENST
Field       : EVENST.DSP_IDENT_EVENT
```

DESCRIPTION

```
-----
Resp Proj - insert record key not working.  Also not able to down
arrow and enter a record either.
```

CORRECTIVE ACTION

```
-----
11/30/92 - This is the correct functionality.
```

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

ansaction: LE2110U3 QAPP Definition

=====

Comment ID : 621 Priority: LOW Date: 29-NOV-92

=====

Status : NOTPROB Date: 30-NOV-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TEST USER

System ID : 92000009129 Oracle ROWID: 00000753.000E.0007

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : EVENT_NEST

Block : EVENST

Field : EVENST.DSP_IDENT_EVENT

DESCRIPTION

other functions keys not responding del, F12

CORRECTIVE ACTION

11/30/92 - Correct functionality

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2110U3 QAPP Definition

=====
Comment ID : 622 Priority: LOW Date: 29-NOV-92
=====

Status : FIXED Date: 30-NOV-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TEST USER

System ID : 92000009157 Oracle ROWID: 00000571.0013.0007

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : LEMIS_STATUS

Block : EVESTA

Field : EVESTA.DATE_STATUS

DESCRIPTION

date format should be dd/mm/yy instead of whats stated in help

CORRECTIVE ACTION

11/30/92 - Fixed help.

1.6 LE2110U5 SAP DEFINITION

LEMIS Module Test Checklist Report

Name: SAP DEFINITION
 Short Name: LE2110U5
 Purpose: Maintain SAP Information.

Initial Test Date: 11/30/92 *de*

Verification Date: 12/2/92 *RJA*

	Disp	Security			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: EVENT
 Title: SAP

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
TITLE_EVENT	CHAR	40	()	()	()	()	()
REV_EVENT	NUMBER	3	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: EVENT
Title: QAPP

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
TITLE_EVENT	CHAR	40	()	()	()	()	()
REV_EVENT	NUMBER	3	()	()	()	()	()

Block: EVENT_LOCATION_J
Title: Locations For A SAP

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: EVENT_METHOD_J
Title: Methods In SAP

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: EVENT_NEST
Title: Responsible QAPP

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: EVENT_PERSON

Title: Persons Associated With SAP

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
NAME_PERSON	CHAR	25	()	()	()	()	()
TITLE_FUNC_UNIT	CHAR	20	()	()	()	()	()
ADDR_PHONE	CHAR	13	()	()	()	()	()
ADDR_MS	CHAR	7	()	()	()	()	()
RELATIONSHIP	CHAR	3	()	()	()	()	()

Block: LEMIS_LOCATION

Title: LEMIS Location

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_LOCATION	CHAR	20	()	()	()	()	()
TITLE_LOCATION	CHAR	40	()	()	()	()	()

Block: LEMIS_STATUS

Title: SAP Status

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_STATUS	CHAR	4	()	()	()	()	()
DATE_STATUS	DATE	8	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: METHOD

Title: Laboratory Methods

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	30	()	()	()	()	()
TITLE_METHOD	CHAR	40	()	()	()	()	()

Block: TYPE_LEMIS_STATUS
 Title: Event Status Types

Help ()	Scroll ()	Text ()	Navigate ()	Ins ()	Del ()	Upd ()	Qry ()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
-----	-----	---	---	---	-----	-----	---
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

This module is used to define a SAP, including:

- 1) SAP definition
- 2) SAP status maintenance
- 3) Person data
- 4) SAP Location linkage
- 5) Analytical Methods linkage

The SAP is subordinate to and cannot exist without a parent QAPP.

The SAP identifies the specific, within close proximity, the actual sampling locations on the target stream that will be sampled. S&ML will use text to further refine the physical sampling position if it differs any from the defined location.

Text may be associated with the SAP.

Locations will be identified by their description. Display of the unique identifier and WIDS ID, if any, is suppressed.

The SAP method list is extracted from the QAPP shopping list. In this case, the specific methods to be employed on this stream are pulled from the QAPP list. Should the testing to be performed on the samples from a location change in the future, the SAP itself will be updated.

The SAP cannot contain a method that is not also identified in the QAPP, even an alternate.

People and organizations are associated with the SAP:

1. Organizations. Use LOV link to organization list. Multiple organizations may be associated with specific relationship codes.
2. Persons. Use literal values. Multiple persons can be associated with specific relationship codes.

SAP status information will default to CR (create), with a default date of the current date (the date will likely be changed by the operator). Text may be associated with the status. Multiple status lines are possible, display at least two.

Module Help

 This screen supports the update and retrieval functions for the Sample Analysis Plan (SAP) upon which the individual liquid effluent sampling activities are based. The screen is used to define a SAP and its association to a QAPP. The status of the SAP may also be updated from this screen.

 To store a new SAP:

1. Enter the SAP identifier. This identifier may be up to 20 characters in length. The identifier must be unique or an error will occur. The field is required.
2. TAB and enter the title of the SAP. The title may be from one to 40 characters in length and is required.
3. TAB to the REV field. This is the revision identifier of the SAP. The field is required. A default value of zero is displayed. To accept the default, press the ENTER key. To enter a different revision level, key the revision number over the default and press ENTER. (A number is required.)

The Text? flag indicates whether free-form text is associated with the requirement. If the flag is set to 'N', no text is present. To add text, press the TEXT key (F7). A popup screen will appear that will permit you to enter as much text as desired. Several different types of text may be available. Once the text has been stored, the text flag will be set to 'Y'. Additional text processing assistance can be obtained by pressing CTRL/K while the text popup window is being displayed.

4. Press the COMMIT key (F10) to store the data in the database.

To update an existing SAP:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the SAP to be updated. If you only know part of the identifier, enter that part followed by a percent sign (%). This will cause those SAPs that begin with the partial identifier to be queried.
3. Press the RUN QY key (Shift/F12). The SAP will be displayed on the screen; if more than one SAP was queried, the first one matching the query will be displayed. To locate the desired SAP:
 - a. Press the down arrow key until the desired SAP is displayed.
4. Press the TAB key until the cursor is at the field to be modified.
5. Enter the new data in the field and press ENTER.
6. If more than one field is to be updated, repeat the above process for each field.
7. Press the COMMIT key (F10) to store the changes in the database.

Module Help

If the text associated with the SAP is to be modified:

1. Press the TEXT key (F7). The default description text will be displayed. If you wish to modify another text type:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type will be displayed.
2. Press the PAGE DOWN key to move the cursor to the text block.
3. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
4. Press the COMMIT key (F10) to update the database.

SAPs may not be deleted if subordinate sample schedules or Sample Analysis Forms (SAF) are present. To delete a SAF:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the SAF to be deleted.
3. Press the RUN QY key (Shift/F12). The SAF will be displayed on the screen.
4. Press the DELETE grey key. The screen should clear.
5. Press the COMMIT key (F10) to update the database.

Each SAP may be assigned to a QAPP. For example, SAP WHC-SD-FF-PLN-002 may be assigned to QAPP WHC-SD-WM-QAPP-011.

To associate a SAP with its responsible QAPP:

1. Press the PAGE DOWN key to move the cursor to the QAPP ID field of the Responsible QAPP block.
2. Enter the identifier of the QAPP, if known. If not known, press LOV (F9) to display a list of available QAPPs:
 - a. Press the down arrow key until the cursor points to the desired QAPP.
 - b. Press the ENTER key to select the QAPP. The QAPP and its information will be displayed on the screen.
3. Press the COMMIT key (F10) to store the association of the SAP to its QAPP in the database.

To remove the association between a SAP and its QAPP:

1. Press the PAGE DOWN key to move the cursor to the QAPP ID field of the Responsible QAPP block.
2. If the QAPP identifier displayed is the one whose relationship is to be deleted, press the DELETE key.

Module Help

-
3. Press the COMMIT key (F10) to complete the update of the database.
-

The status of each SAP is tracked in the SAP Status block. Here you may identify the different types of status conditions and the dates on which the condition occurs. Unlimited text may be added to clarify the status condition. To add a new status condition:

1. Press the PAGE DOWN key to move the cursor to the first status line.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Press LOV to display the available valid status condition codes.
4. Use the up and down arrow keys to select the desired status condition.
5. Press the COMMIT (F10) key to move the status to the screen.
6. To add more statuses, press the down arrow key to the next blank space and repeat the selection process.

If all of the spaces are filled and more statuses are to be added, press the down arrow key and the already entered records will scroll upward, leaving space for an additional record to be added.

7. Press the COMMIT (F10) key to store the selected status conditions in the database as associated with the SAP.

To store text expanding the description of a status condition:

1. Use the up and down arrow keys to move the cursor to the status condition line where text is to be stored.
2. Press the TEXT key (F7). The text popup window will be displayed with cursor at the text type code. If a text type other than that displayed is desired:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type, if any, will be displayed.
3. Press the PAGE DOWN key to move the cursor to the text block.
4. If no text is present, enter the desired text. If text is present:
 - a. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
 - b. To add new text at the end, press the down arrow key until a blank line appears. Enter the text.
5. Press the COMMIT key (F10) to store the text in the database.
6. Press the ENTER key until the popup window disappears and the requirement screen reappears.

Module Help

7. If more text is to be added to other status condition entries, use the up and down arrow keys to move the cursor to the desired entry and repeat the above steps.

Each SAP identifies special analysis classes, analysis classes, and analytical methods that are to be used to analyze the constituents at the waste locations identified by the SAP. One or more items may be defined. NOTE: Only the items identified in the QAPP are allowed to be defined in the SAP. The LOV will display only this restricted list.

To identify an analysis item being used by the SAP:

1. Press the PAGE DOWN key until the cursor is at the METHOD ID field of the Methods In SAP block.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Press LOV (F9) to display a list of methods, analysis classes, and special analysis classes that may be used by the SAP.
4. Select the desired item by one of the following options:
 - a. Use the up and down arrow keys to move the cursor to the item to be selected. Press 'X' to make the selection.
 - b. Press ENTER QY (F12), TAB until the cursor is at the TYPE field, enter the desired type code, and press the RUN QY (SHIFT/F12). this will reduce the display to only the items of that type. Press 'X' to select those items desired.
 - c. Press SHIFT/F5 to select all items in the list. NOTE: This option can be used when most of the items in the list will be selected. After loading the entire list, selectively remove those not desired.

A combination of the above can be used. The query (b) can be used to reduce the list and then SHIFT/F5 to select those remaining. The reduced list can be picked from selectively by using the individual selection option (a).

5. Use the up and down arrow keys to move the cursor to other items to be selected, pressing 'X' each time to make the selection.
6. Press the COMMIT key (F10) to register the selections.

To remove an existing item:

1. Press the PAGE DOWN key until the cursor is at the METHOD ID field of the Methods In SAP block.
2. Press the down arrow key, if required, to move the cursor to the item to be removed.
3. Press the DELETE grey key. The item should disappear from the screen.
4. Press the COMMIT key (F10) to update the database.

Module Help

Each SAP identifies one or more specific locations where sampling will take place. Locations to be selected must have previously been entered into the LEMIS location list.

To add a location to a SAP:

1. Press the PAGE DOWN key until the cursor is at the LOCATION ID field in the Locations For A SAP block.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Press LOV (F9) to display a list of available locations.
4. Use the up and down arrow keys to move the cursor to the location to be selected.
5. Press the COMMIT key (F10) to move the selected location to the main screen.
6. If additional locations are desired, repeat the above process.

If all of the spaces are filled and more locations are to be added, press the down arrow key to the bottom entry and then continue to press the down arrow until a blank entry appears. The already entered locations will scroll upward, leaving space for an additional location to be entered.

7. Press the COMMIT key (F10) to store the data in the database.

To remove an existing location:

1. Press the PAGE DOWN key until the cursor is at the LOCATION ID field of the Locations For A SAP block.
2. Press the down arrow key, if required, until the cursor points to the location to be removed.
3. Press the DELETE grey key. The location should disappear from the screen line.
4. Press the COMMIT key (F10) to update the database.

Persons may be associated with a SAP. This information is used by the subsequently created SAFs to identify the default customer and OSM coordinators. Normally only two persons are identified.

To store person data with the SAP:

1. Press the PAGE DOWN key until the cursor is located at the NAME field of the Persons Associated With SAP block.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Enter the name of the person. Use format of first name or initial, middle initial, and last name.
4. TAB to the title field and enter a 1-40 character title of the person. The available space on the screen line is less than the

Module Help

allowable 40 characters. If the title is longer than the space available on the screen, the title will scroll to the left to permit entering all of the title.

5. TAB to the phone number field and enter the phone number of the person.
6. TAB to the mail stop field and enter the person's mail stop.
7. TAB to the (rel)ationship field. Enter CUS if identifying the customer; enter OSM if identifying the OSM coordinator.
8. If additional persons are to be added, repeat the above process beginning at step 2.

If all of the spaces are filled and more persons are to be added, press the down arrow key to the bottom entry and then continue to press the down arrow until a blank entry appears. The already entered persons will scroll upward, leaving space for an additional person to be entered.

9. Press the COMMIT key (F10) to store the data in the database.

To modify an existing person:

1. Press the PAGE DOWN key until the cursor is at the NAME field of the Persons Associated With SAP block.
2. Press the down arrow key, if required, until the cursor points to the person to be modified.
3. TAB to the field to be changed and modify the field contents.
4. If more than one field is to be changed, TAB to each field and modify the contents.
5. Press the COMMIT key (F10) to update the database.

To remove a person from a SAP:

1. Press the PAGE DOWN key until the cursor is at the NAME field of the Persons Associated with SAP block.
2. Press the down arrow key, if required, until the cursor points to the person to be removed.
3. Press the DELETE grey key. The person should disappear from the display.
4. Press the COMMIT key (F10) to update the database.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.

Module Test Plan

- 3. System testing.
- 4. End user testing.
- 5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
 - Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
 - Developer _____ Date Complete _____
- () 8 - Secondary testing complete, no problems found.

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the

Module Test Plan

corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.

4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Test status conditions are:

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.
Tester _____ Date Complete _____

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.

Project Manager [Signature] Date Complete 12/2/92

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.

System Architect [Signature] Date Complete 12/2/92

Module Test Case

- 63 ✓ Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
- () Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- 7(✓ Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
- (✓ Transaction help describes:
- (✓ Purpose of the transaction
 - (✓ Functions performed by the transaction
 - (✓ Where multiple page, what happens on each page
- (✓ Add a new record by using the insert key.
- (✓ The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- (✓ Add a new record by scrolling past the last entry using the down arrow key.
- () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓ A new data record can be successfully added.
- (✓ Query a single record.
- (✓ A single record is retrieved.
 - (✓ Pressing ~~SHIFT~~ F12 clears the record and permits entering a second query.
 - (✓ The second query successfully retrieves the desired data.
- (✓ Query multiple records using wild cards.
- (✓ Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓ A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - (✓ A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (✓ A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through
- 643 ✓ Change each field in an existing record. Verify that the change

occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- () Change each field individually, committing after each change.
- () Change each field, as per the field test plan, for both valid and invalid values.
- (/) Delete an existing record.
 - (/) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - () All data fields are correctly stored in the database.

Block: EVENT
Title: SAP

Block Description

The event table is a multi-use table. It identifies those activities that are defined to guide and control the liquid effluent program.

An event, as currently (10/9/92) defined may be:

The TPA itself.

A TPA milestone (milestone 17 for LEMIS and will be known as a project.

A submilestone within milestone 17 (will be known as a task).

The QAPP defining LEMIS.

The SAPs defining the waste stream processing.

The SAFs defining the actual sampling activities.

In the hierarchy of things:

The TPA is the top level.

TPA Milestone 17 is dependent on the TPA.

TPA submilestones of 17 depend on Milestone 17.

Block: EVENT
 Title: SAP

Block Description

Project

On HEIS, a project is a funded program, such as 300-Area cleanup. For release one of LEMIS, all sampling will be part of the same project, which is Liquid Effluent Sampling, or Tri-Party Agreement milestone number 17.

A project is very tightly coupled to its QAPjP which describes many aspects of how the project will be done.

LEMIS will record the project for each sample that is recorded. During release one, project book keeping may be invisible to the user, and may be done automatically by LEMIS, since only one project will exist.

A project will have the following attributes:

Project name: "TPA Liquid Effluent Sampling " is suggested.
 User Id: M-17-00 is suggested.

A project is related to:

One QAPjP at a time.
 Many tasks.
 Perhaps a high level location, such as Hanford.

QAPP/QAPjP (Quality Assurance Project Plan)

The QAPP, covering all 33 major streams, is dependent on TPA Milestone 17 (the project).

For Release One, the project will be related to one QAPjP at a time. However, a QAPjP may be revised and changed many times, so a program (project) may be related to many different QAPjPs (or at least many different versions) over the history of the project.

A QAPjP will have the following characteristics:

ID Number: The document number
 Title: Title of the document
 Rev Number: Revision number of the document

A QAPjP is related to one and only one project.

Block: EVENT
Title: SAP

Block Description

LEMIS will know the QAPjP title and rev number for each sample that is recorded.

Task

Each project consists of many tasks. On HEIS, a task is typically a number from a project document that itemizes tasks to be performed. For LEMIS, a task is typically a stream specific milestone, such as M-17-XX.

The task stream (which is a high level location), and Sampling Analysis Plan (SAP) are very tightly coupled. A task corresponds to one and only one stream and one and only one SAP at a given time. This allows the user to select a task by either the milestone number or the stream name. Only one SAP describes a task at a time, however the SAP may be revised and changed throughout a program. Also, only one SAF will be active for a task at a time.

For each sample taken, LEMIS will record the corresponding task, the applicable SAP, and the applicable SAF.

A task has the following attributes:

Task name: Descriptive title.
User Id: M-17-XX is suggested.

A task is related to:

Exactly one project
Exactly one stream location (a high level location)
One or many sampling locations (the actual sampling locations)
Exactly one SAP at a time
Zero or or active SAFs at a time (but many SAFs may apply over time).

LEMIS will know the task, SAP name and revision, and SAF number for each sample that is recorded.

SAP (Sample Analysis Plan)

The SAPs, one for each waste stream, are dependent on the QAPP. Each SAP is also related to the submilestone (task). At any given point in time, a SAP and task are on a one to one relationship. It is possible for there to be multiple SAPs over time related to a single task.

The SAFs occur as part of the sampling defined by the SAP. Several SAFs may occur for a SAP but it is unlikely that more than one will be

Block: EVENT
Title: SAP

Block Description

"active" at one time. This does not include those pre-scheduled for future sampling. Each SAF has a subordinate Sample-Authorization table entry containing specific information related to the SAF.

Block Test Case

Block Test Results Expected

Field Name: IDENT_EVENT

Field Description

This field uniquely identifies the event.

Field Test Case

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms,
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - () LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - () Default value when appropriate

Block: EVENT
Title: SAP

Field Name: IDENT_EVENT

Field Test Results Expected

- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
 - (✓) ~~Attempt~~ ^{file} Store a series of records using the SHIFT/~~F7~~ [duplicate field] key to carry forward field values from the previously stored record.
 - () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
 - (✓) Any alphanumeric characters may be entered into the field.
 - (✓) Attempt to enter data as lower case. ~~An error should result.~~
 - (✓) Delete an existing value.
 - (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT
Title: SAP

Field Name: TITLE_EVENT

Field Description

Field Test Case

- (/) Begin field with a space.
- (/) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (/) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (/) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (/) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (/) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (/) Field help describes:
 - (/) Purpose of the field
 - () LOV reference if available
 - (/) Valid range of values where present
 - (/) Required indicator when field is NOT NULL
 - () Default value when appropriate
- (/) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (/) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- (/) Store a series of records using the SHIFT/EF [duplicate field] key

Block: EVENT
Title: SAP

Field Name: TITLE_EVENT

Field Test Results Expected

-
- to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
 - (✓) Any alphanumeric characters may be entered into the field.
 - (✓) Attempt to enter data as lower case. ~~An error should result.~~
 - (✓) Delete an existing value.
 - (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Field Name: REV_EVENT

Field Description

This field represents a revision to the event.

Field Test Case

Field Test Results Expected

Block: EVENT
Title: SAP

Field Name: FLAG_TEXT

Field Description

Field Test Case

- NO testing*
- () Accept the default.
 - () Override the default with a valid entry value.
 - () Override the default with an invalid data value.
 - () Begin field with a space.
 - () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
 - () Set keyboard on lower case and enter data. Data should be converted to upper case.
 - () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.

Block: EVENT
Title: SAP

Field Name: FLAG_TEXT

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT_LOCATION_J
 Title: Locations For A SAP

Block Description

Block Test Case

Block Test Results Expected

- (✓) Display table/block help. Verify that the narrative adequately describes the purpose and functions of the table/block in end-user terms.
 - (✓) Table/block help describes:
 - (✓) Purpose of the table/block
 - (✓) Functions performed by the table/block
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all fields of the entry block.
 - (✓) A new record can be successfully added and committed.
- () Add a new record by scrolling past the last entry using the down arrow key.
 - (✓) All fields of the table/block blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
 - (✓) Pressing F12 clears the block and makes the block ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through
- () Change each field in an existing record. Verify that the change

Block: EVENT_LOCATION_J
 Title: Locations For A SAP

Block Test Results Expected

occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- () Change each field individually, committing after each change.
- () Change each field, as per the field test plan, for both valid and invalid values.

✓ Delete an existing record.

- ✓ () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.

- () All data fields are correctly stored in the database.

Block: EVENT_METHOD_J.
Title: Methods In SAP

Block Description

Block Test Case

Block Test Results Expected

- (/) Display table/block help. Verify that the narrative adequately describes the purpose and functions of the table/block in end-user terms.
 - (/) Table/block help describes:
 - (/) Purpose of the table/block
 - (/) Functions performed by the table/block
- (/) Add a new record by using the insert key.
 - () The insert key clears all fields of the entry block.
 - (/) A new record can be successfully added and committed.
- (/) Add a new record by scrolling past the last entry using the down arrow key.
 - () All fields of the table/block blank when the end of the string of retrieved records is reached.
 - () A new data record can be successfully added.
- (/) Query a single record.
 - (/) A single record is retrieved.
 - (/) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - (/) The second query successfully retrieves the desired data.
- (/) Query multiple records using wild cards.
 - (/) Pressing F12 clears the block and makes the block ready for a query.
 - (/) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - (/) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (/) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through
- () Change each field in an existing record. Verify that the change

N/A

Block: EVENT_METHOD_J
 Title: Methods In SAP

Block Test Results Expected

occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- () Change each field individually, committing after each change.
- () Change each field, as per the field test plan, for both valid and invalid values.
- ✓ () Delete an existing record.
 - ✓ () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date-updated, owner role, and any other autogenerated fields.
- () All data fields are correctly stored in the database.

Block: EVENT_NEST
 Title: Responsible QAPP

= QAPP

Block Description

The table links events to each other. The most common linkage will be that of a series of events that are grouped together under another event definition. Another option is to link a series of events that support the same repetitive set of sampling.

Block Test Case

Block Test Results Expected

- ✓ () Use F9 (Loc) to Select QAPP. (for New SAP)
- ✓ () Try to Delete The Rec - s/not allow
- ✓ () Try to Clear The field & Commit - s/b mand.
- ✓ () Try using F9 to change a Committed Record
- ✓ () Try inserting New Record - s/not allow
- ✓ () Try entering invalid Value on New Record

Block: EVENT_PERSON
 Title: Persons Associated With SAP

Block Description

This table links an event with the person that requests the event. This linkage normally will take place at the highest level of event definition but may occur at all levels. A person may have multiple associations with an event:

REQ - requestor
 RES - responsible for managing
 AUT - author of event description
 SAM - sampling participant
 SAC - sampling coordinator/manager

Block Test Case

Block Test Results Expected

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - () Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- 685 (✓) Add a new record by using the insert key.
 - () The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- 685 (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - () A new data record can be successfully added.

Block: EVENT_PERSON
Title: Persons Associated With SAP

Block Test Results Expected

- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- (✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
 - (✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.

Block: EVENT_PERSON
Title: Persons Associated With SAP

Block Test Results Expected

() All data fields are correctly stored in the database.

Field Name: NAME_PERSON

Field Description

Field Test Case

Field Test Results Expected

Field Name: TITLE_FUNC_UNIT

Field Description

Field Test Case

Field Test Results Expected

Field Name: ADDR_PHONE

Field Description

Field Test Case

Field Test Results Expected

Field Name: ADDR_MS

Field Description

Field Test Case

Field Test Results Expected

no
testing

Block: EVENT_PERSON
Title: Persons Associated With SAP

Field Name: RELATIONSHIP

Field Description

Field Test Case

- (/) Accept the default.
- (/) Override the default with a valid entry value.
- 2(A) () Override the default with an invalid data value.
- (/) Begin field with a space.
- (/) Enter data beyond length of field.
 - (/) If autoskip is in effect, cursor will skip to the next field.
 - (/) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (/) Set keyboard on lower case and enter data. Data should be converted to upper case.
- n(A) () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- 650 () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- 1(A) () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Change the field, committing after each change.

Block: EVENT_PERSON
 Title: Persons Associated With SAP

Field Name: RELATIONSHIP

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or ~~COM00103~~
- n/A* () Store a series of records using the SHIFT/~~F7~~ [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- N/A* () Modify each default value with a valid alternative.
- N/A* () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. ~~An error should result.~~
- () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: LEMIS_STATUS
Title: SAP Status

Block Description

Block Test Case

Block Test Results Expected

- ✓ (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- ✓ (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - () Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- ✓ (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- ✓ (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - (✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - (✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
 - () A new data record can be successfully added.
- ✓ (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing ~~Shift~~/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- ✓ (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction

Block: LEMIS STATUS
 Title: SAP Status

Block Test Results Expected

- ready for a query.
- (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
 - (✓) Change each ^{updatable} field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
 - (✓) Delete an existing record.
 - (✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
 - () Display the _JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - () All data fields are correctly stored in the database.

Field Name: TYPE_STATUS

Field Description

Block: LEMIS_STATUS
Title: SAP Status

Field Name: TYPE_STATUS

Field Test Case

- ☒ Accept the default.
- ☒ Override the default with a valid entry value.
- ☒ Override the default with an invalid data value.
- ☒ Begin field with a space.
- ☒ Enter data beyond length of field.
 - ☒ If autoskip is in effect, cursor will skip to the next field.
 - ☒ If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ☒ Set keyboard on lower case and enter data. Data should be converted to upper case.
- ☒ Clear an existing field and attempt to update the record.

Field Test Results Expected

- ☒ Display table/block help. Verify that the narrative adequately describes the purpose and functions of the table/block in end-user
 - ☒ Table/block help describes:
 - ☒ Purpose of the table/block
 - ☒ Functions performed by the table/block
- ☒ Add a new record by using the insert key.
 - ☒ The insert key clears all fields of the entry block.
 - ☒ A new record can be successfully added and committed.
- ☒ Add a new record by scrolling past the last entry using the down arrow key.
 - ☒ All fields of the table/block blank when the end of the string of retrieved records is reached.
 - ☒ A new data record can be successfully added.
- ☒ Query a single record.

Block: LEMIS_STATUS
 Title: SAP Status

Field Name: TYPE_STATUS

Field Test Results Expected

- ☒ A single record is retrieved.
- ☒ Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
- ☒ The second query successfully retrieves the desired data.
- ☒ Query multiple records using wild cards.
- ☒ Pressing F12 clears the block and makes the block ready for a query.
- ☒ A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- ☒ A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- ☒ A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- () Change each field in an existing record. Verify that the change *not updatable*
 - () Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
- () Delete an existing record.
 - () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record. *not updatable*
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- ☒ Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- ☒ Field help describes:
 - ☒ Purpose of the field
 - ☒ LOV reference if available
 - () Valid range of values where present

Block: LEMIS_STATUS
Title: SAP Status

Field Name: TYPE_STATUS

Field Test Results Expected

- () Required indicator when field is NOT NULL
- () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made. Dup
- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- n/A () Store a series of records using the SHIFT/^{F6} [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- n/A () Modify each default value with a valid alternative.
- n/A () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. ~~An error should result.~~
- () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

Block: LEMIS_STATUS
Title: SAP Status

Field Name: TYPE_STATUS

Field Test Results Expected

N/A () Attempt to bypass the field. An error message should tell you that the field data must be entered.

N/A () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

has Default - will
not allow blank
field

Block: LEMIS_STATUS
Title: SAP Status

Field Name: DATE_STATUS

Field Description

Field Test Case

(☒) Accept the default.

(☒) Override the default with a valid entry value.

N/A (☐) Override the default with an invalid data value.

(☐) Clear an existing field and attempt to update the record.

64 (☒) Attempt to store an invalid date:

(☒) Enter the day as 00.

(☒) Enter the day as 32.

(☒) Enter the day as 30 in February.

(☒) Enter the day as 31 September.

Fixed
A

Field Test Results Expected

(☒) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

(☒) Field help describes:

(☒) Purpose of the field

(☒) LOV reference if available

(☒) Valid range of values where present

(☒) Required indicator when field is NOT NULL

(☒) Default value when appropriate

(☒) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

(☒) Change the field, committing after each change.

(☐) Change the field where a range of valid values exists:

(☐) Value below valid range should display message COM00101 or COM00103 (if no LOV).

(☐) Value above valid range should display message COM00101 or

Block: LEMIS_STATUS
 Title: SAP Status

Field Name: DATE_STATUS

Field Test Results Expected

-
- (A) Store a series of records using the SHIFT/~~F7~~^{F6} [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- (A) Modify each default value with a valid alternative.
- (A) Modify each default value with an invalid alternative.
- (A) Enter an invalid date. The invalid date should be rejected.
- () Database record must contain the proper autogenerated value. Use SQL*Plus to verify contents.
- (A) Delete an existing value.
- (A) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (A) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (A) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: LEMIS_STATUS
Title: SAP Status

no
testing

Field Name: FLAG_TEXT

Field Description

This field is a flag that identifies when supporting text is present for a data row. The valid values are either "Y" (yes) or "N" (no). When the user desires to add text in support of a data row, the flag is set to "Y". The system then brings up the general text form and permits the user to enter text of varying types for the data row. Where text is already present, the user places the cursor on the flag field and selects to retrieve the text data. Text data may be updated. Note that updating text data causes the same hierarchical cloning that takes place elsewhere when a data row is updated. This ensures that the text data is also properly audit trail covered.

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:

Block: LEMIS_STATUS
Title: SAP Status

Field Name: FLAG_TEXT

Field Test Results Expected

- () Purpose of the field
- () LOV reference if available
- () Valid range of values where present
- () Required indicator when field is NOT NULL
- () Default value when appropriate

- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.

- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.

Block: LEMIS_STATUS
Title: SAP Status

Field Name: FLAG_TEXT

Field Test Results Expected

-
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2110U5 SAP Definition

=====

Comment ID : 647 Priority: LOW Date: 30-NOV-92

=====

Status : FIXED Date: 30-NOV-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TEST USER

System ID : 92000009237 Oracle ROWID:

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : LEMIS_STATUS

Block : EVESTA

Field : EVESTA.TYPE_STATUS

DESCRIPTION

date format should use slashes. Hint text should use slashes. Help
text should use mm/dd/yy format.F-700 11/30/92

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2110U5 SAP Definition

=====
Comment ID : 644 Priority: LOW Date: 30-NOV-92
=====

Status : FIXED Date: 30-NOV-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TEST USER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40350 Query caused no records to be retrieved.

Table : EVENT_NEST

Block : EVENST

Field : EVENST.DSP_IDENT_EVENT

DESCRIPTION

allowed an invalid entry.

CORRECTIVE ACTION

11/30/92 - Fixed

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2110U5 SAP Definition

=====

Comment ID : 611 Priority: LOW Date: 27-NOV-92

=====

Status : ASSIGNED Date: 28-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By : KEN ATKINS

System ID : Oracle ROWID: 0000079A.0007.0007

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : EVENT_METHOD_J

Block : EVLABP

Field : EVLABP.DSP_IDENT_METHOD

DESCRIPTION

The Help for this item states that the user should use the down arrow key to navigate to the first blank record, then use LOV to select a method. If there are 20 or 30 methods this could take awhile. An easier method is to press 'Insert' to create a blank record.

✓

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2110U5 SAP Definition

```
=====
Comment ID : 612    Priority: LOW                    Date: 27-NOV-92
=====
Status      : ASSIGNED    Date: 28-NOV-92    Class:
Type        : PROBLEM
Assigned To :            LARRY TOWNER
Entered By  :            KEN ATKINS
System ID   :                                  Oracle ROWID: 0000075D.0012.0007
Last Error  :            -0
Last Msg    : FRM-40400   Transaction complete -- 2 records posted an
Table       : EVENT_PERSON
Block       : EVEPER
Field       : EVEPER.NAME_PERSON
```

DESCRIPTION

There is no help entered for the EVENT_PERSON fields or block.

11/29

OK
jt

1.7 LE2121U1 SCHEDULE SAMPLES

LEMIS Module Test Checklist Report

Name: SCHEDULE SAMPLES
 Short Name: LE2121U1
 Purpose: Sample Schedule Definition.

Initial Test Date: 10/30/92 LH

Verification Date: 12/2/92 RJA

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: EVENT

Title: Sample Analysis Plan

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
TITLE_EVENT	CHAR	40	()	()	()	()	()

Block: EVENT

Title: Sample Schedule

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_EVENT	CHAR	20	()	()	()	()	()
LEVEL_DATA_VALID	NUMBER	1	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: EVENT_NEST

Title: Sample Analysis Plan

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: EVENT_ORGAN_J

Title: Organizations Associated With Event

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: EVENT_PERSON

Title: Persons Associated With Events

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
NAME_PERSON	CHAR	25	()	()	()	()	()
ADDR_PHONE	CHAR	18	()	()	()	()	()
ADDR_MS	CHAR	10	()	()	()	()	()

Block: EVENT_PERSON

Title: Person Requesting Sample

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
NAME_PERSON	CHAR	20	()	()	()	()	()
ADDR_PHONE	CHAR	13	()	()	()	()	()
ADDR_MS	CHAR	7	()	()	()	()	()

Block: LEMIS_STATUS

Title: Scheduled Date

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DATE_STATUS	DATE	8	()	()	()	()	()

Block: ORGANIZATION

Title: Organization

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_ORGAN	CHAR	12	()	()	()	()	()
TITLE	CHAR	40	()	()	()	()	()

Block: TYPE_DATA_VALID_CODE

Title: Data Validation Levels

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Module Description

This form provides the ETP operator with the ability to schedule a future sampling activity. The form contains a base minimum of information:

Module Description

- 1) Customer name, org, phone, msin (optional [see below]).
These fields will be in uppercase.
- a) Requested sampling date (may be changed).
- 3) Parent SAP (once stored may not be changed).
- 4) Unique identifier (may not be altered by operator).

Under the covers, a SAF is created. It has none of the details of the SAF. It can have two person associations only:

1. Persons. Use literal values. Multiple persons can be associated with specific relationship codes:
 - a. Customer, with relationship code CUS.
 - b. OSM coordinator, with relationship code OSM.

The person information is derived from the SAP as a default that may be changed by the operator.

Module Help

This screen supports the update and retrieval functions for the scheduling of samples that will be taken in support of the liquid effluent monitoring activities. This screen is used to schedule a sampling event. The schedule is associated with a SAP. Modifications to the schedule may be handled from this screen.

To store a new sample schedule:

1. Enter the SAP identifier. This identifier may be up to 20 characters in length. The identifier must be unique or an error will occur. The field is required. If the SAP identifier is not known, press LOV (F9) to display a list of available SAPs under which this schedule may be established:
 - a. Press the down arrow key until the cursor points to the desired SAP.
 - b. Press the ENTER key to select the desired SAP. The SAP identifier and title will be displayed.
2. The default generated schedule identifier will be displayed. If the default is acceptable (it is a combination of the SAP identifier and a unique numeric suffix), press ENTER. Otherwise enter the schedule identifier.
3. Enter the date on which the sampling event is requested to occur. The date is entered mm/dd/yy.
4. The customer and OSM coordinator information is extracted automatically from the SAP. If this default information is correct, proceed to the next step. Otherwise:
 - a. TAB the cursor to a field that must be changed from the default.
 - b. Enter the desired data value.
 - c. TAB to the next field to be changed and repeat the process\

687
WRONG

Module Help

The Text? flag indicates whether free-form text is associated with the requirement. If the flag is set to 'N', no text is present. To add text, press the TEXT key (F7). A popup screen will appear that will permit you to enter as much text as desired. Several different types of text may be available. Once the text has been stored, the text flag will be set to 'Y'. Additional text processing assistance can be obtained by pressing CTRL/K while the text popup window is being displayed.

5. Press the COMMIT key (F10) to store the data in the database.

To update an existing sample schedule:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the sample schedule to be updated. If you only know part of the identifier, enter that part followed by a percent sign (%). This will cause schedules that begin with the partial identifier to be queried.
3. Press the RUN QY key (Shift/F12). The schedule will be displayed on the screen; if more than one schedule was queried, the first one matching the query will be displayed. To locate the desired schedule:
 - a. Press the down arrow key until the desired schedule is displayed.
4. Press the TAB key until the cursor is at the field to be modified.
5. Enter the new data in the field and press ENTER.
6. If more than one field is to be updated, repeat the above process for each field.
7. Press the COMMIT key (F10) to store the changes in the database.

If the text associated with the schedule is to be modified:

1. Press the TEXT key (F7). The default description text will be displayed. If you wish to modify another text type:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type will be displayed.
2. Press the PAGE DOWN key to move the cursor to the text block.
3. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
4. Press the COMMIT key (F10) to update the database.

Sample schedules may not be deleted if a SAF has been created in support of the schedule. To delete a schedule:

Module Help

1. Press the ENTER QY key (F12).
2. Enter the identifier of the schedule to be deleted.
3. Press the RUN QY key (Shift/F12). The schedule will be displayed on the screen.
4. Press the DELETE grey key. The screen should clear.
5. Press the COMMIT key (F10) to update the database.

Module Test Plan

Module Test Case

Block: EVENT
Title: Sample Schedule

Block Description

The event table is a multi-use table. It identifies those activities that are defined to guide and control the liquid effluent program.

An event, as currently (10/9/92) defined may be:

The TPA itself.

A TPA milestone (milestone 17 for LEMIS and will be known as a project.

A submilestone within milestone 17 (will be known as a task).

The QAPP defining LEMIS.

The SAPs defining the waste stream processing.

The SAFs defining the actual sampling activities.

In the hierarchy of things:

The TPA is the top level.

TPA Milestone 17 is dependent on the TPA.

TPA submilestones of 17 depend on Milestone 17.

Project

On HEIS, a project is a funded program, such as 300-Area cleanup. For release one of LEMIS, all sampling will be part of the same project, which is Liquid Effluent Sampling, or Tri-Party Agreement milestone number 17.

Block: EVENT
Title: Sample Schedule

Block Description

A project is very tightly coupled to its QAPjP which describes many aspects of how the project will be done.

LEMIS will record the project for each sample that is recorded. During release one, project book keeping may be invisible to the user, and may be done automatically by LEMIS, since only one project will exist.

A project will have the following attributes:

Project name: "TPA Liquid Effluent Sampling " is suggested.
User Id: M-17-00 is suggested.

A project is related to:

One QAPjP at a time.
Many tasks.
Perhaps a high level location, such as Hanford.

QAPP/QAPjP (Quality Assurance Project Plan)

The QAPP, covering all 33 major streams, is dependent on TPA Milestone 17 (the project).

For Release One, the project will be related to one QAPjP at a time. However, a QAPjP may be revised and changed many times, so a program (project) may be related to many different QAPjPs (or at least many different versions) over the history of the project.

A QAPjP will have the following characteristics:

ID Number: The document number
Title: Title of the document
Rev Number: Revision number of the document

A QAPjP is related to one and only one project.

LEMIS will know the QAPjP title and rev number for each sample that is recorded.

Task

Each project consists of many tasks. On HEIS, a task is typically a number from a project document that itemizes tasks to be performed. For LEMIS, a task is typically a stream specific milestone, such as M-17-XX.

Handwritten:
Project
M-17-XX

Block: EVENT
 Title: Sample Schedule

Block Description

The task stream (which is a high level location), and Sampling Analysis Plan (SAP) are very tightly coupled. A task corresponds to one and only one stream and one and only one SAP at a given time. This allows the user to select a task by either the milestone number or the stream name. Only one SAP describes a task at a time, however the SAP may be revised and changed throughout a program. Also, only one SAF will be active for a task at a time.

For each sample taken, LEMIS will record the corresponding task, the applicable SAP, and the applicable SAF.

A task has the following attributes:

Task name: Descriptive title.
 User Id: M-17-XX is suggested.

A task is related to:

Exactly one project
 Exactly one stream location (a high level location)
 One or many sampling locations (the actual sampling locations)
 Exactly one SAP at a time
 Zero or or active SAFs at a time (but many SAFs may apply over time).

LEMIS will know the task, SAP name and revision, and SAF number for each sample that is recorded.

SAP (Sample Analysis Plan)

The SAPs, one for each waste stream, are dependent on the QAPP. Each SAP is also related to the submilestone (task). At any given point in time, a SAP and task are on a one to one relationship. It is possible for there to be multiple SAPs over time related to a single task.

The SAFs occur as part of the sampling defined by the SAP. Several SAFs may occur for a SAP but it is unlikely that more than one will be "active" at one time. This does not include those pre-scheduled for future sampling. Each SAF has a subordinate Sample-Authorization table entry containing specific information related to the SAF.

Block Test Case

Block Test Results Expected

check - 11/19

Block: EVENT
Title: Sample Schedule

Field Name: IDENT_EVENT *SAP ID*

Field Description

This field uniquely identifies the event.

Field Test Case

☒ Begin field with a space.

☒ Enter data beyond length of field.

() If autoskip is in effect, cursor will skip to the next field.

☒ If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

☒ Set keyboard on lower case and enter data. Data should be converted to upper case.

☒ Clear an existing field and attempt to update the record.

Field Test Results Expected

() Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

☒ Field help describes:

☒ Purpose of the field

☒ LOV reference if available

☒ Valid range of values where present

☒ Required indicator when field is NOT NULL

☒ Default value when appropriate

() Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

() Change the field, committing after each change.

() Change the field where a range of valid values exists:

() Value below valid range should display message COM00101 or

() Value above valid range should display message COM00101 or COM00103 (if no LOV).

6/27 Fred
6/26
Can't change to SAP
linked to SAP

Block: EVENT
Title: Sample Schedule

Field Name: IDENT_EVENT

Field Test Results Expected

~~NA~~ Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

~~NA~~ Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

~~NA~~ Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

(~~1~~) Any alphanumeric characters may be entered into the field.

(~~1~~) Attempt to enter data as lower case. An error should result.

(~~1~~) Delete an existing value.

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

(~~1~~) Attempt to bypass the field. An error message should tell you that the field data must be entered.

(~~1~~) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Handwritten: Change series

Handwritten: requires mnt of to valid

Block: EVENT
Title: Sample Schedule

Field Name: LEVEL_DATA_VALID

Field Description

This field contains the current validation level of the data row. The valid values range from 0 through 9; the actual valid values are determined by the code table LEVEL_DATA_VALID_CODE. The default value is zero, indicating that the data is unvalidated. The user may update this value as the validation condition changes.

Field Test Case

- (~~1~~) Accept the default.
- (~~1~~) Override the default with a valid entry value.
- (~~1~~) Override the default with an invalid data value.
- (~~1~~) Clear an existing field and attempt to update the record.
- (~~1~~) Attempt to enter alpha data.
- (~~1~~) Attempt to enter longer value than allowed.
- (~~1~~) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- () Attempt to enter negative value. No error should occur.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - (~~1~~) Purpose of the field
 - (~~1~~) LOV reference if available
 - (~~1~~) Valid range of values where present
 - (~~1~~) Required indicator when field is NOT NULL
 - (~~1~~) Default value when appropriate
- (~~1~~) Change the field in an existing record. Verify that the change

Handwritten notes:
 Change the field in an existing record. Verify that the change

Handwritten notes:
 191

Handwritten notes:
 accepted during
 628
 accepted neg
 on screen but not in DB

Block: EVENT
Title: Sample Schedule

Field Name: LEVEL_DATA_VALID

Field Test Results Expected

occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- (✓) Change the field, committing after each change.
- (✓) Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

WJ () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

WJ () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

(✓) Use SQL*Plus to verify all data fields are processed correctly.

- () All data fields are correctly stored in the database.

(✓) Modify ~~each~~ ^{DL} default value with a valid alternative.

(✓) Modify ~~each~~ ^{DL} default value with an invalid alternative.

Ryb () Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.

(✓) Begin field entry with a space. An error should occur.

(✓) Delete an existing value.

- (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

Block: EVENT
Title: Sample Schedule

Field Name: LEVEL_DATA_VALID

Field Test Results Expected

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT
Title: Sample Schedule

Field Name: FLAG_TEXT

Field Description

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.

Block: EVENT
Title: Sample Schedule

Field Name: FLAG_TEXT ,

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: EVENT_NEST
Title: Sample Analysis Plan

Block Description

The table links events to each other. The most common linkage will be that of a series of events that are grouped together under another event definition. Another option is to link a series of events that support the same repetitive set of sampling.

Block Test Case

Block Test Results Expected

Block: EVENT_ORGAN_J
 Title: Organizations Associated With Event

Block Description

This table links an event with the organization that requests the event. This linkage normally will take place at the highest level of event definition but may occur at all levels. An organization may have multiple associations with an event:

REQ - requestor
 RES - responsible for managing
 AUT - author of event description
 SAM - sampling participant
 SAC - sampling coordinator/manager

*CVS - Customer
 oom -*

632

Block Test Case

Block Test Results Expected

- () Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - () Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- () Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - () Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- () Add a new record by using the insert key.
 - () The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- () Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - () A new data record can be successfully added.

631

*632
 Customer data*

Block: EVENT_ORGAN_J
 Title: Organizations Associated With Event

Block Test Results Expected

- () Query a single record.
 - () A single record is retrieved.
 - () Pressing SHIFT/F12 clears the record and permits entering a second query.
 - () The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
 - () Pressing F12 clears the screen and makes the transaction ready for a query.
 - () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- () Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
- () Delete an existing record.
 - () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date-updated, owner role, and any other autogenerated fields.

Block: EVENT_ORGAN_J

Title: Organizations Associated With Event

Block Test Results Expected

() All data fields are correctly stored in the database.

Block: EVENT_PERSON

Title: Persons Associated With Events

Block Description

This table links an event with the person that requests the event. This linkage normally will take place at the highest level of event definition but may occur at all levels. A person may have multiple associations with an event:

REQ - requestor

RES - responsible for managing

AUT - author of event description

SAM - sampling participant

SAC - sampling coordinator/manager

Block Test Case

Block Test Results Expected

- () Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - () Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- () Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - () Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- () Add a new record by using the insert key.
 - () The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- () Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - () A new data record can be successfully added.

Block: EVENT_PERSON

Title: Persons Associated With Events

Block Test Results Expected

- () Query a single record.
 - () A single record is retrieved.
 - () Pressing SHIFT/F12 clears the record and permits entering a second query.
 - () The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
 - () Pressing F12 clears the screen and makes the transaction ready for a query.
 - () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- () Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
- () Delete an existing record.
 - () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date-updated, owner role, and any other autogenerated fields.

Block: EVENT_PERSON

Title: Persons Associated With Events

Block Test Results Expected

() All data fields are correctly stored in the database.

Field Name: NAME_PERSON

Field Description

Field Test Case

Field Test Results Expected

Field Name: ADDR_PHONE

Field Description

Field Test Case

Field Test Results Expected

Field Name: ADDR_MS

Field Description

Field Test Case

Field Test Results Expected

Block: EVENT_PERSON
 Title: Person Requesting Sample

Block Description

This table links an event with the person that requests the event. This linkage normally will take place at the highest level of event definition but may occur at all levels. A person may have multiple associations with an event:

REQ - requestor
 RES - responsible for managing
 AUT - author of event description
 SAM - sampling participant
 SAC - sampling coordinator/manager

Block Test Case

Block Test Results Expected

- () Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - () Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- () Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - () Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- () Add a new record by using the insert key.
 - () The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- () Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - () A new data record can be successfully added.

Block: EVENT_PERSON
Title: Person Requesting Sample

Block Test Results Expected

- () Query a single record.
 - () A single record is retrieved.
 - () Pressing SHIFT/F12 clears the record and permits entering a second query.
 - () The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
 - () Pressing F12 clears the screen and makes the transaction ready for a query.
 - () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- () Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
- () Delete an existing record.
 - () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.

Block: EVENT_PERSON
Title: Person Requesting Sample

Block Test Results Expected`

() All data fields are correctly stored in the database.

Field Name: NAME_PERSON

Field Description

Field Test Case

Field Test Results Expected

Field Name: ADDR_PHONE

Field Description

Field Test Case

Field Test Results Expected

Field Name: ADDR_MS

Field Description

Field Test Case

Field Test Results Expected

Block: LEMIS_STATUS
 Title: Scheduled Date

Block Description

Block Test Case

Block Test Results Expected

- () Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
- () Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- () Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - () Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- () Add a new record by using the insert key.
 - () The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- () Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - () A new data record can be successfully added.
- () Query a single record.
 - () A single record is retrieved.
 - () Pressing SHIFT/F12 clears the record and permits entering a second query.
 - () The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
 - () Pressing F12 clears the screen and makes the transaction

Block: LEMIS_STATUS
Title: Scheduled Date

Block Test Results Expected

- ready for a query.
- () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- () Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
- () Delete an existing record.
 - () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - () All data fields are correctly stored in the database.

Field Name: DATE_STATUS

Field Description

Block: LEMIS_STATUS
Title: Scheduled Date

Field Name: DATE_STATUS

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Clear an existing field and attempt to update the record.
- () Attempt to store an invalid date:
 - () Enter the day as 00.
 - () Enter the day as 32.
 - () Enter the day as 30 in February.
 - () Enter the day as 31 September.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

Block: LEMIS_STATUS
Title: Scheduled Date

Field Name: DATE_STATUS

Field Test Results Expected

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Enter an invalid date. The invalid date should be rejected.
- () Database record must contain the proper autogenerated value. Use SQL*Plus to verify contents.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

30-NOV-92

WHC-SD-WM-TP-163 REV 0

Page 1 of 1

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2121U1 Schedule Samples

```
=====
Comment ID : 627      Priority: LOW      Date: 29-NOV-92
=====
Status      : FIXED      Date: 30-NOV-92      Class:
Type        : PROBLEM
Assigned To:              KEN ATKINS
Entered By  :              LARRY TOWNER
System ID   :              Oracle ROWID:
Last Error  :              -0
Last Msg    : FRM-40350    Query caused no records to be retrieved.
Table       : EVENT_NEST
Block       : EVENST
Field       : EVENST.DSP_IDENT_EVENT
=====
```

DESCRIPTION

Element help indicates that a variety of requirements may go in the field when only SAP is valid. Help indicates that entering an existing value will cause an error when an existing value must be entered or an error will occur.

CORRECTIVE ACTION

11/30/92 - Fixed.

*Verified
RGA
12/2/92*

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

ansaction: LE2121U1 Schedule Samples

=====
Comment ID : 687 Priority: LOW Date: 02-DEC-92
=====

Status : FIXED Date: 02-DEC-92 Class:
Type : PROBLEM
Assigned To: KEN ATKINS
Entered By : TODD ADAMS
System ID : Oracle ROWID:
Last Error : -0
Last Msg : FRM-40102 Record must be entered or deleted first.
Table : LEMIS_STATUS
Block : EVESTA
Field : EVESTA.DATE_STATUS

DESCRIPTION

Module help text is not correct. Under, To store a new sample schedule:, #2. It says when you come back from the LOV, the schedule identifier will be displayed. It it not displayed until the user hits tab or enter to move out of the SAP ID field. It also says the user can press enter and accept the schedule ID or enter another one. I could not find a way into that field to enter a different ID. ???

CORRECTIVE ACTION

12/2/92 - Fixed the help.

1.8 LE4041U1 LEMIS LOCATION MAINTENANCE

LEMIS Module Test Checklist Report

Name: LEMIS LOCATION MAINTENANCE
 Short Name: LE4041U1
 Purpose: Maintain LEMIS location data.

Initial Test Date: 11/30/92 *RA*

Verification Date: 12/1/92 *RA*

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: LEMIS_LOCATION
 Title: LEMIS Location

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_LOCATION	CHAR	20	()	()	()	()	()
TITLE_LOCATION	CHAR	40	()	()	()	()	()
LEVEL_LOCATION	NUMBER	2	()	()	()	()	()
TYPE_LOCATION	CHAR	1	()	()	()	()	()
TYPE_SITE	NUMBER	2	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: LEMIS_LOCATION
Title: LEMIS Location

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_LOCATION	CHAR	20	()	()	()	()	()
TITLE_LOCATION	CHAR	40	()	()	()	()	()
LEVEL_LOCATION	NUMBER	2	()	()	()	()	()

Block: LEMIS_LOCATION_NEST
Title: LEMIS Location Hierarchy

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
RELATIONSHIP	CHAR	3	()	()	()	()	()

Module Description

This module provides the ability to add and modify but not delete a location.

A link to other locations is permitted. Locations may link to higher level locations as a subordinate; to lower level locations as a parent.

A link to organizations is handled through the locat_organ_j junction table. The relationship codes are: REL (related, for organizations that are associated with the location), MGR (manager, for the organization managing the location).

The data validation code defaults to 9. The ident_owner defaults to 'library'.

The type of site flag is supported by LOV to Type_site_code code table.

Module Help

This screen supports the update and retrieval functions for the LEMIS locations where the liquid effluent sampling activities take place. This screen is used to define the sampling locations and related locatins. This screen may also be used to update location data.

To store a new location:

1. Enter the location identifier. This identifier may be up to

Module Help

- 20 characters in length. The identifier must be unique or an error will occur. The field is required.
2. TAB and enter the title of the location. The title may be from one to 40 characters in length and is required.
 3. TAB to the LOC LEVEL field. Use LOV to select the appropriate level code.
 - a. Press the down arrow key until the cursor points to the desired code.
 - b. Press the ENTER key to select the desired code onto the screen.
 - c. The desired code and its description will be displayed.
 4. TAB to the LOC TYPE field. Press LOV (F9) to display the valid location types. The field is required.
 5. Enter the SITE TYPE. Press LOV (F9) to display the valid site type codes. The field is required.

The Text? flag indicates whether free-form text is associated with the requirement. If the flag is set to 'N', no text is present. To add text, press the TEXT key (F7). A popup screen will appear that will permit you to enter as much text as desired. Several different types of text may be available. Once the text has been stored, the text flag will be set to 'Y'. Additional text processing assistance can be obtained by pressing the HELP key (F1) while the text popup window is being displayed.

5. Press the COMMIT key (F10) to store the data in the database.

To update an existing location:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the location to be updated. If you only know part of the identifier, enter that part followed by a percent sign (%). This will cause those locations that begin with the partial identifier to be queried.
3. Press the RUN QY key (Shift/F12). The location will be displayed on the screen; if more than one location was queried, the first one matching the query will be displayed. To locate the desired location:
 - a. Press the down arrow key until the desired location is displayed.
4. Press the TAB key until the cursor is at the field to be modified.
5. Enter the new data in the field and press ENTER.
6. If more than one field is to be updated, repeat the above process for each field.
7. Press the COMMIT key (F10) to store the changes in the database.

If the text associated with the location is to be modified:

1. Press the TEXT key (F7). The default description text will be displayed. If you wish to modify another text type:

Module Help

- a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type will be displayed.
2. Press the PAGE DOWN key to move the cursor to the text block.
3. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
4. Press the COMMIT key (F10) to update the database.

Locations may not be deleted if other sampling information that uses location is present. To delete a location:

1. Press the ENTER QY key (F12).
2. Enter the identifier of the location to be deleted.
3. Press the RUN QY key (Shift/F12). The location will be displayed on the screen.
4. Press the DELETE grey key. The screen should clear.
5. Press the COMMIT key (F10) to update the database.

Each location may be assigned to a location hierarchy. For example, a stream sampling location is associated with a waste stream.

To associate a location with another location:

1. Press the PAGE DOWN key to move the cursor to the LOC ID field of the LEMIS Location Hierarchy block.
2. Enter the identifier of the associated location, if known. If not known, press LOV (F9) to display a list of available locations:
 - a. Press the down arrow key until the cursor points to the desired location.
 - b. Press the ENTER key to select the location. The location and its title will be displayed on the screen.
3. Press the COMMIT key (F10) to store the association of the location to its related location in the database.

To remove the association between a location and its associated location:

1. Press the PAGE DOWN key to move the cursor to the LOC ID field of the LEMIS Location Hierarchy block.
2. If the location identifier displayed is the one whose relationship is to be deleted, press the DELETE key.
3. Press the COMMIT key (F10) to complete the update of the database.

Module Help

The status of each xxxxxxxxxxxx is tracked in the xxxxxxxxxxxx Status block. Here you may identify the different types of status conditions and the dates on which the condition occurs. Unlimited text may be added to clarify the status condition. To add a new status condition:

1. Press the PAGE DOWN key to move the cursor to the first status line.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Press LOV to display the available valid status condition codes.
4. Use the up and down arrow keys to select the desired status condition.
5. Press the COMMIT (F10) key to move the status to the screen.
6. To add more statuses, press the down arrow key to the next blank space and repeat the selection process.

If all of the spaces are filled and more statuses are to be added, press the down arrow key and the already entered records will scroll upward, leaving space for an additional record to be added.

7. Press the COMMIT (F10) key to store the selected status conditions in the database as associated with the xxxxxxxxxxxx.

To store text expanding the description of a status condition:

1. Use the up and down arrow keys to move the cursor to the status condition line where text is to be stored.
2. Press the TEXT key (F7). The text popup window will be displayed with cursor at the text type code. If a text type other than that displayed is desired:
 - a. Press LOV (F9) to display the other text options.
 - b. Use the down arrow key to move the cursor to the desired text type.
 - c. Press ENTER to select the desired text type. The text for that text type, if any, will be displayed.
3. Press the PAGE DOWN key to move the cursor to the text block.
4. If no text is present, enter the desired text. If text is present:
 - a. Use the up and down arrow keys to move the cursor to the line to be modified. Change the text as required. If special editing of text is necessary, use the KEY HELP (Shift/F1) to display the editing keys available.
 - b. To add new text at the end, press the down arrow key until a blank line appears. Enter the text.
5. Press the COMMIT key (F10) to store the text in the database.
6. Press the ENTER key until the popup window disappears and the requirement screen reappears.
7. If more text is to be added to other status condition entries, use the up and down arrow keys to move the cursor to the desired

Module Help

entry and repeat the above steps.

Module Test PlanModule Test Case

70
test

Block: LEMIS_LOCATION

Title: LEMIS Location

Block DescriptionBlock Test CaseBlock Test Results Expected

Field Name: IDENT_LOCATION

Field DescriptionField Test Case

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
- () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user

Block: LEMIS_LOCATION
 Title: LEMIS Location

Field Name: IDENT_LOCATION

Field Test Results Expected

terms.

- () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database. *SKIP*
- () Use SQL*Plus to verify all data fields are processed correctly. *SKIP*
 - () All data fields are correctly stored in the database.
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you

Block: LEMIS_LOCATION
Title: LEMIS Location

Field Name: IDENT_LOCATION

Field Test Results Expected

that the field data must be entered.

- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- () Attempt to enter data as lower case. An error should result.

Block: LEMIS_LOCATION
Title: LEMIS Location

Field Name: TITLE_LOCATION

Field Description

The field is a 1-40 character free-form text title of the data item.

Field Test Case

- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key

Block: LEMIS_LOCATION
 Title: LEMIS Location

Field Name: TITLE_LOCATION

Field Test Results Expected

to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database. *SKIP*

() Use SQL*Plus to verify all data fields are processed correctly. *SKIP*

() All data fields are correctly stored in the database.

(✓) Any alphanumeric characters may be entered into the field.

(✓) Delete an existing value.

(✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

(✓) Attempt to enter data as lower case. ~~An error should result~~ *dup*

Block: LEMIS_LOCATION
Title: LEMIS Location

Field Name: LEVEL_LOCATION

Field Description

Field Test Case

- ☒ Accept the default.
- ☒ Override the default with a valid entry value.
- ☒ Override the default with an invalid data value.
- ☒ Clear an existing field and attempt to update the record. *Non-upd.*
- ☒ Attempt to enter alpha data.
- ☒ Attempt to enter longer value than allowed.
- ☒ Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- ☐ Attempt to enter negative value. No error should occur.

Field Test Results Expected

- ☒ Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

- 661* ☒ Field help describes:
- ☒ Purpose of the field
 - ☒ LOV reference if available
 - ☒ Valid range of values where present
 - ☒ Required indicator when field is NOT NULL
 - ☒ Default value when appropriate

- ☐ Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - ☐ Change the field, committing after each change.
 - ☐ Change the field where a range of valid values exists:
- Non-update*

Block: LEMIS_LOCATION
 Title: LEMIS Location

Field Name: LEVEL_LOCATION

Field Test Results Expected

- () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly. *okup*
- () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly. *okup*
- () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative. *DUP*
- () Modify each default value with an invalid alternative. *DUP*
- () Delete an existing value. *DUP*
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- (✓) Numeric data (0-9), ~~comma (,), plus (+), minus (-), and period (.)~~ are permitted.
- (✓) ~~Begin field entry with a space. An error should occur.~~

will test justify

Block: LEMIS_LOCATION
Title: LEMIS Location

Field Name: TYPE_LOCATION

Field Description

Field Test Case

- N/A for (char F12)
- () Begin field with a space.
 - () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
 - () Set keyboard on lower case and enter data. Data should be converted to upper case.
 - () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

Block: LEMIS_LOCATION
 Title: LEMIS Location

Field Name: TYPE_LOCATION

Field Test Results Expected

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database. *SKIP*

() Use SQL*Plus to verify all data fields are processed correctly. *SKIP*

() All data fields are correctly stored in the database.

☒ Any alphanumeric characters may be entered into the field.

☒ Delete an existing value. *Done*

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

☒ Attempt to bypass the field. An error message should tell you that the field data must be entered.

☒ Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

☒ Attempt to enter data as lower case. An error should result.

Field Name: TYPE_SITE

Field Description

Field Test Case

Field Test Results Expected

Block: LEMIS_LOCATION
Title: LEMIS Location

Field Name: FLAG_TEXT

Field Description

Field Test Case

Field Test Results Expected

Block: LEMIS_LOCATION_NEST
 Title: LEMIS Location Hierarchy

Block Description

Block Test Case

Block Test Results Expected

Field Name: RELATIONSHIP

Field Description

The relationship identifies the nature of the linkage between two occurrences of the table parents. A default value is typically assigned. Other values are found in a LOV supported by TYPE_RELATIONSHIP_CODE.

Field Test Case

- (/ Accept the default.
- 60 (/ Override the default with a valid entry value.
- (/ Override the default with an invalid data value.
- (/ Begin field with a space.
- (/ Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (/ Set keyboard on lower case and enter data. Data should be converted to upper case.
- (/ Clear an existing field and attempt to update the record.

Field Test Results Expected

- (/ Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

670 Dup Rec

Block: LEMIS_LOCATION_NEST
Title: LEMIS Location Hierarchy

Field Name: RELATIONSHIP

Field Test Results Expected

- () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made. *DUP*
- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly. *skip*
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly. *skip*
- () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative. *DUP*
- () Modify each default value with an invalid alternative. *DUP*
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

Block: LEMIS_LOCATION_NEST
Title: LEMIS Location Hierarchy

Field Name: RELATIONSHIP

Field Test Results Expected

- (/) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (/) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- (/) Attempt to enter data as lower case. An error should result.

Ident location
671,

Programmer's Bug Report

Transaction: LE4041U1 Enter Lemis Locations

```
=====
Comment ID : 624    Priority: LOW          Date: 29-NOV-92
=====
Status      : FIXED      Date: 30-NOV-92   Class:
Type        : PROBLEM
Assigned To :          KEN ATKINS
Entered By  :          TODD ADAMS
System ID   : 10000001228      Oracle ROWID: 00000022.0011.0018
Last Error  :      -0
Last Msg    : FRM-40400 Transaction complete -- 2 records posted an
Table       : LOCATION
Block       : LOCATE
Field       : LOCATE.TYPE_LOCATION
```

DESCRIPTION

```
-----
Location Type 'L - Logical Location' does not show up in the
LOV, even though I added it to CG_REF_CODES. Ken??
```

CORRECTIVE ACTION

```
-----
11/30/92 - For some reason the form pointed to the
LOCATION.TYPE_LOCATION domain. I changed it in the form to the
LEMIS_LOCATION.TYPE_LOCATION domain.
```

Programmer's Bug Report

Transaction: LE4041U1 Enter Lemis Locations

=====

Comment ID : 666 Priority: LOW Date: 30-NOV-92

=====

Status : ASSIGNED Date: 30-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : KEN ATKINS

System ID : 92000009276 Oracle ROWID: 0000056A.0015.0007

Last Error : -0

Last Msg : FRM-40202 Field must be entered.

Table : LEMIS_LOCATION

Block : LEMLOC

Field : LEMLOC.TYPE_SITE

DESCRIPTION

List of values for site type should have description.

FIXED
12/11/92
✓

12/11/92 A

Programmer's Bug Report

Transaction: LE4041U1 Enter Lemis Locations

```
=====
Comment ID : 669      Priority: LOW      Date: 30-NOV-92
=====
Status      : ASSIGNED   Date: 30-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : KEN ATKINS
System ID   : 10000001434      Oracle ROWID: 0000030F.0001.0007
Last Error  : -0
Last Msg    : FRM-40400 Transaction complete -- 1 records posted an
Table       : LEMIS_LOCATION_NEST
Block       : LMLOCN
Field       : LMLOCN.RELATIONSHIP
```

DESCRIPTION

No validation or LOV on this field.

FIXED -

*REMOVED
REL*

*FROM DWPLM -
DEFAULT TO SUB.*

12/1/92

ver 12/1/92

Transaction: LE4041U1 Enter Lemis Locations

=====
Comment ID : 667 Priority: LOW Date: 30-NOV-92
=====

Status : ASSIGNED Date: 30-NOV-92 Class:
Type : PROBLEM
Assigned To: TODD ADAMS
Entered By: KEN ATKINS
System ID : 92000009276 Oracle ROWID: 0000056A.0015.0007
Last Error : -0
Last Msg : FRM-40407 Transaction complete -- posted records comm
Table : LEMIS_LOCATION
Block : LEMLOC
Field : LEMLOC.TYPE_SITE

DESCRIPTION

This field allows blank but does not have 'OPTIONAL - ' in its hint.
either fix the hint, or fix the field.

FIXED

12/1/92

✓

✓ verified
on 1/1/93 AS

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE4041U1 Enter Lemis Locations

```
=====
Comment ID : 662      Priority: LOW      Date: 30-NOV-92
=====
Status      : ASSIGNED      Date: 30-NOV-92      Class:
Type        : COMMENT
Assigned To : TODD ADAMS
Entered By  : KEN ATKINS
System ID   : 92000009211      Oracle ROWID: 00000567.0000.0007
Last Error  : -0
Last Msg    : FRM-40350 Query caused no records to be retrieved.
Table       : LEMIS_LOCATION
Block       : LEMLOC
Field       : LEMLOC.IDENT_LOCATION
```

DESCRIPTION

You should display more than two related locations since there is
enough room.

*FIXED**12/1/92**✓**verified
12/1/92* *Ⓟ*

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE4041U1 Enter Lemis Locations

=====
Comment ID : 677 Priority: LOW Date: 01-DEC-92
=====

Status : ASSIGNED Date: 01-DEC-92 Class:
Type : PROBLEM
Assigned To: TODD ADAMS
Entered By : KEN ATKINS
System ID : 10000001434 Oracle ROWID: 00000567.0013.0007
Last Error : -0
Last Msg : FRM-40400 Transaction complete -- 1 records posted an
Table : LEMIS_LOCATION
Block : LEMLOC
Field : LEMLOC.TYPE_SITE

DESCRIPTION

The site type description should be displayed on the form.

FIXED

12/1/92

✓

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE4041U1 Enter Lemis Locations

=====
Comment ID : 686 Priority: LOW Date: 01-DEC-92
=====Status : FIXED Date: 01-DEC-92 Class:
Type : PROBLEM
Assigned To: KEN ATKINS
Entered By : TEST USER
System ID : Oracle ROWID:
Last Error : -0
Last Msg : FRM-50016 Legal characters are 0-9 - + E .
Table : LEMIS_LOCATION
Block : LEMLOC
Field : LEMLOC.TYPE_SITEDESCRIPTION

lower case letters allowed to be entered into field.

CORRECTIVE ACTION

12/1/92 - Fixed

2.0 GENERIC

2.1 GE1410Q1 PERSON LOV

There is no test booklet for this module because testing for PERSON LOV was tested in GE2410U1 ASSIGN ROLE.

2.2 GE2130Q1 DISPLAY MENU

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

2.3 GE2130R1 MENU STRUCTURE REPORT

Initial Test Date: 6/25/93 LC Verification Date: 6/25/93 LC

- (✓) Verify that the report is accessible from the menu.
- (✓) Verify that the report is run correctly when requested.
- (✓) Verify that the report format is correct.
- (✓) Verify that the correct data is listed in the report.

2.4 GE2320Q1 SELECT ROLE

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

2.5 GE2410U1 ASSIGN ROLE

LEMIS Module Test Checklist Report

Name: ASSIGN ROLE
 Short Name: GE2410U1
 Purpose: To assign users to roles.

Initial Test Date: 11/30/92 RJA

Verification Date: 12/1/92 RJA

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: APPLICATION
 Title: Application

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
APPLICATION_ACRONYM	CHAR	10	()	()	()	()	()

Block: PERSON

Title: Person

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
NAME_LAST	CHAR	15	()	()	()	()	()
NAME_FIRST	CHAR	10	()	()	()	()	()
INITIALS	CHAR	4	()	()	()	()	()
TITLE	CHAR	20	()	()	()	()	()

Block: ROLE

Title: Role

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_APPLICATION	CHAR	2	()	()	()	()	()
IDENT_ROLE	CHAR	12	()	()	()	()	()
TITLE_ROLE	CHAR	28	()	()	()	()	()

Block: ROLE_FUNC_UNIT_J

Title: People for this role

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Module Description

Module Help

This transaction is used to assign people to specific access roles. A person's access role determines which transactions they will have access to and what type of access they will have for each transaction (ie allow update, query only etc). The list of transactions and type of access has been set up previously using the 'Role Maintenance' transaction.

The first block in this transaction is a query only block that displays the applications and their roles.

Module Help

The second block is a list of all of the people assigned to the displayed role.

To give a person a particular role:

- 1) Make sure the cursor is in the 'Role' block.
- 2) Press <F12> (enter query) and enter the application ID and the desired role (if known). If the role is not entered, all roles for the application will be queried. List of values is available for these two fields.
- 3) Press <Shift-F12> to execute the query.
- 4) If role was not entered, press the down arrow key until the desired role appears.
- 5) Press the <Page Down> key to move the cursor to the 'People for this role' block.
- 6) Press the <Insert> key to add a new person to the list.

The 'LEMIS Person Directory' popup window will appear. This window functions in a similar fashion to the HLAN POPFON directory.

- 7) Start entering the last name of the person to be added to the list. As you type the name, the highlighted row will move through the list of people, showing the first person whose last name starts with the letters you have entered.
- 8) When the person you want to add is highlighted, press <F10> to add them to the role. If you do not want to add anyone, press <Esc> or <F4>. If you want to erase a character that you have entered, use the <Shift-Tab> key.

The person's name should now appear on the screen.

- 9) Use <F10> to save this addition to the database.

To remove a person from a role, perform steps 1)->5) above and then:

- 6) Use the down arrow key to move the cursor to the person's name you wish to remove.
- 7) Press the <Delete> key to remove their name from the list.
- 8) Press the <F10> key to save this change to the database.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.

Module Test Plan

2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

Test status conditions are:

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
 - Developer _____ Date Complete _____
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
- () 8 - Secondary testing complete, no problems found.
 - Tester _____ Date Complete _____

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of

Module Test Plan

the transaction for correction of the problems.

3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.

4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.
- Project Manager LSJ Date Complete 4/25/93

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.
- System Architect LA Date Complete 1/25/93

Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - (✓) Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - (✓) A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - (✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - (✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (○) Query a single record.
 - (✓) A single record is retrieved.
 - () Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - () The second query successfully retrieves the desired data.
- (✓) Query multiple records using wild cards.
 - () Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through
- (✓) Change each field in an existing record. Verify that the change

occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- (✓) Change each field individually, committing after each change.
- (✓) Change each field, as per the field test plan, for both valid and invalid values.

(✓) Delete an existing record.

- (✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date-updated, owner role, and any other autogenerated fields.

- () All data fields are correctly stored in the database.

Block: ROLE_FUNC_UNIT_J
Title: People for this role

Block Description

Block Test Case

Block Test Results Expected

2.6 GE2620U1 ASSIGN DATA OWNERSHIP

LEMIS Module Test Checklist Report

Name: ASSIGN DATA OWNERSHIP
 Short Name: GE2620U1
 Purpose: To assign Data Ownership to people

Initial Test Date: 11/30/92 RDA

Verification Date: 12/1/92 RDA

Help	Disp Fkeys	Security			
		Ins	Del	Query	Upd
()	()	()	()	()	()

Block: APPLICATION

Title: Application

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
APPLICATION_ACRONYM	CHAR	10	()	()	()	()	()

Block: DATA_OWNER

Title: Data Ownership

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_APPLICATION	CHAR	2	()	()	()	()	()
IDENT_OWNER	CHAR	12	()	()	()	()	()
DESCR_OWNER	CHAR	26	()	()	()	()	()

Block: DATA_OWNER_PERSON_J

Title: People with this Data Ownership

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
RELATIONSHIP	CHAR	3	()	()	()	()	()

Block: PERSON

Title: Person

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
NAME_LAST	CHAR	15	()	()	()	()	()
NAME_FIRST	CHAR	10	()	()	()	()	()
INITIALS	CHAR	4	()	()	()	()	()
TITLE	CHAR	20	()	()	()	()	()

Module Description

Module Help

This transaction is used to assign people to specific data ownerships. It can also be used to see which people are currently assigned to a particular data ownership. The data ownership itself cannot be updated from this transaction.

The first block in this transaction is a query only block that displays the applications and data ownership identifiers.

The second block is a list of all of the people assigned to the displayed data ownership.

To give a person a particular data ownership:

- 1) Make sure the cursor is in the 'Data Ownership' block.
- 2) Press <F12> (enter query) and enter the application ID and the desired data ownership (if known). If the data ownership is not entered, all data ownerships for the application will be queried. List of Values is available for these two fields.
- 3) Press <Shift-F12> to execute the query.
- 4) If data ownership was not entered, press the down arrow key until the desired data ownership appears.
- 5) Press the <Page Down> key to move the cursor to the 'People with

Module Help

this Data Ownership' block.

- 6) Press the <Insert> key to add a new person to the list.

The 'LEMIS Person Directory' popup window will appear. This window functions in a similar fashion to the HLAN POPFON directory.

- 7) Start entering the last name of the person to be added to the list. As you type the name, the highlighted row will move through the list of people, showing the first person whose last name starts with the letters you have entered.
- 8) When the person you want to add is highlighted, press <F10> to add them to the data ownership. If you do not want to add anyone, press <Esc> or <F4>. If you want to erase a character that you have entered, use the <Shift-Tab> key.

The person's name should now appear on the screen.

- 9) Use <F10> to save this addition to the database.

To remove a person from a data ownership perform steps 1)->5) above and then:

- 6) Use the down arrow key to move the cursor the the person's name you wish to remove.
- 7) Press the <Delete> key to remove their name from the list.
- 8) Press the <F10> key to save this change to the database.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary are the transaction progresses through the development and test steps.

Developer Testing

Module Test Plan

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

Test status conditions are:

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
 Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
 Developer _____ Date Complete _____
- () 8 - Secondary testing complete, no problems found.

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

Module Test Plan

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.

Project Manager Z. J. Date Complete 12/1/92

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.

System Architect [Signature] Date Complete 12/1/92

Module Test Case

(✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

() Duplicate row insertion attempts result in error message
COM00102 [Duplicate value, enter new value or F1 for help]

(✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

[Signature]

#673 ↓

- () Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - (✓) A new record can be successfully added and committed.
- () Add a new record by scrolling past the last entry using the down arrow key.
 - (✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - (✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing ~~Shift~~/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through
- (✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change each field individually, committing after each change.
 - (✓) Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
 - (✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date updated, owner role, and any other autogenerated fields.
- () All data fields are correctly stored in the database.

Block: DATA_OWNER_PERSON_J

Title: People with this Data Ownership

Block Description

Block Test Case

Block Test Results Expected

Field Name: RELATIONSHIP

Field Description

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be

Block: DATA_OWNER_PERSON_J
 Title: People with this Data Ownership

Field Name: RELATIONSHIP

Field Test Case

 converted to upper case.

() Clear an existing field and attempt to update the record.

Field Test Results Expected

 () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

- () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate

() Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).

() Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

- () All data fields are correctly stored in the database.

Block: DATA_OWNER_PERSON_J
 Title: People with this Data Ownership

Field Name: RELATIONSHIP

Field Test Results Expected

-
- () Modify each default value with a valid alternative.
 - () Modify each default value with an invalid alternative.
 - () Any alphanumeric characters may be entered into the field.
 - () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
 - () Attempt to enter data as lower case. An error should result.

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE2620U1 Assign People to Data Ownership

```
=====
Comment ID : 675      Priority: LOW      Date: 01-DEC-92
=====
Status      : ASSIGNED      Date: 01-DEC-92      Class: OPERATIONAL
Type        : COMMENT
Assigned To : KEN ATKINS
Entered By  : KEN ATKINS
System ID   :      Oracle ROWID:
Last Error  : -
Last Msg    : -
Table       :
Block       :
Field       :
```

DESCRIPTION

Hint text said Title is optional when it is needed to enter the person correctly.

CORRECTIVE ACTION

12/1/92 - Title really IS optional. Set field to non enterable (to save confusion). Also people should only be entered via the person LOV. When a new person was entered, the form allowed their name to be updated, which caused an error. I modified the form to automatically commit a person when entered.

✓ 12/1/92

2.7 GE3120Q1 DISPLAY FUNCTION KEY HELP

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

2.8 GE3130Q1 DISPLAY FUNCTION KEY TEMPLATE

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

2.9 GE3210U1 ENTER A COMMENT OR BUG

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

2.10 GE3250R1 PROGRAMMER'S BUG REPORT

Initial Test Date: JK 6/28/93 Verification Date: JK 6/28/93

- (✓) Verify that the report is accessible from the menu.
- (✓) Verify that the report is run correctly when requested.
- (✓) Verify that the report format is correct.
- (✓) Verify that the correct data is listed in the report.

2.11 GE3420Q1 RELATED DATA POPUP

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

2.12 GE3520Q1 APPLICATION STARTUP SCREEN

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

2.13 GE3721R1 ACCESS LOG REPORT

Initial Test Date: JK 6/28/93 Verification Date: JK 6/28/93

- (✓) Verify that the report is accessible from the menu.
- (✓) Verify that the report is run correctly when requested.
- (✓) Verify that the report format is correct.
- (✓) Verify that the correct data is listed in the report.

2.14 GE3830R1 ERROR MESSAGES BY CODE

Initial Test Date: 6/28/93 Verification Date: 6/28/93

- (X) Verify that the report is accessible from the menu.
- (X) Verify that the report is run correctly when requested.
- (X) Verify that the report format is correct.
- (X) Verify that the correct data is listed in the report.

2.15 GE4130G1 PICKLIST TEMPLATE FORM

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

2.16 GE6014Q1 ONLINE DOCUMENT DISPLAY

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

2.17 GETEXTU1 TEXTUAL DATA ENTRY

This is a support module for the generic application management system. Testing for this module was performed during the testing of all of the other modules in the system.

3.0 LIBRARY

3.1 LI1004U1 MAINTAIN UNIT MEASURE CODE

LEMIS Module Test Checklist Report

Name: TYPE UNIT MEASURE CODE
 Short Name: LI1004U1
 Purpose: To maintain unit of measure codes.

Initial Test Date: 11/19/92 JC

Verification Date: 11-25-92 RJA

Help	Disp	Security			
()	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: TYPE_UNIT_MEASURE_CODE
 Title: Valid Units of Measure

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry	
()	()	()	()	()	()	()	()	
Data Element	Datatype	Dis	Len	Help	LOV	Val/	Not	Ins
SOURCE_UNIT_MEASURE	CHAR	5	()	()	()	()	()	()
DESCRIPTION	CHAR	30	()	()	()	()	()	()
CONVERSION_MODIFIER	NUMBER	18	()	()	()	()	()	()
OBJECT_UNIT_MEASURE	CHAR	5	()	()	()	()	()	()

Block: TYPE_UNIT_MEASURE_CODE
 Title: Valid Units of Measure

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry	
()	()	()	()	()	()	()	()	
Data Element	Datatype	Dis	Len	Help	LOV	Val/	Not	Ins

Module Description

This module is used to add, update or delete valid unit of measure codes and their descriptions.

Each entry contains a source unit of measure, object unit of measure, and conversion formula.

The source unit of measure is used for validation purposes. When a database volumetric column has a default unit of measure other than that of the source, the data is converted to the proper unit of measure.

Display 15 lines of data. The source, object, conversion formula, and description are displayed.

Code table entries may not be deleted while being used elsewhere in the system.

Module Help

This screen supports the maintenance and retrieval of unit of measure code information used by the system. The table contains a list of valid unit of measure codes.

The table contains two unit of measure codes:

1. The source code. This is the code that is validated. It identifies the unit of measure of source data, usually input data being entered into the database. An example source unit measure code would be MILLILITERS (ML).
2. The object code. This is the default unit of measure that is stored in the database. An example object unit measure that corresponds with our source example is LITERS.

In the above example, our database will carry all volume data in liters. Data entered as milliliters must be converted to liters. The table contains a conversion factor field. In our example, the conversion factor value would be .001. This converts 5 milliliters to .005 liters.

To store a new unit of measure code:

1. Use the down arrow key to move the cursor to a blank line.

If all of the spaces are filled and more units of measure are to be added, press the down arrow key to the bottom entry and then continue to press the down arrow key until a blank entry appears. The already entered codes will scroll upward, leaving space for an additional code to be entered.

2. Enter the appropriate source unit of measure.
3. Enter a 1-30 character description of the code.
4. Enter the conversion factor.

Module Help

5. Enter the object unit of measure.
6. to add more codes, press the down arrow key to the next blank space and repeat the entry process.
7. Press the COMMIT key (F10) to store the codes in the database.

To update an existing code:

1. Press the down arrow key until the cursor displays the entry to be updated.
2. The source unit of measure cannot be changed.
3. If the description is to be changed, TAB to the description field and make the change.
4. The conversion factor and object unit of measure cannot be changed.
5. Press the COMMIT key (F10) to update the selected codes in the database.

To delete an existing code:

1. Press the down arrow key until the cursor is located at the code to be deleted.
2. Press the DELETE key. NOTE: If the code is being used by other data in the database, it cannot be deleted. An error message will be displayed.
3. If no error occurs, press the COMMIT key (F10) to delete the code from the database.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

Module Test Plan

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
 - () 5 - Secondary testing complete, problems found.
 - () 6 - Problem resolution in progress.
 - () 7 - Problem resolution complete.
Developer		Date Complete	
- () 8 - Secondary testing complete, no problems found.
Tester	<i>ja</i>	Date Complete	<i>11/19/92</i>

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

Module Test Plan

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

- () 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.
 - () Reviews tests results.
 - () Reviews problem reports to ensure all corrections done.
 - () Updates NOTES text to reflect testing results.
 - Project Manager AA Date Complete 11/21/92
- () 20 - The transaction is placed into the production environment.
 - () Software files in production directories.
 - () DMR 550 updated to add transaction.
 - () DMR 900 updated to add transaction.
 - () Transaction added to production usage menu.
 - System Architect AA Date Complete 11/25/92

Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
- (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

- (✓) Transaction help describes:
 - (✓) Purpose of the transaction
 - (✓) Functions performed by the transaction
 - W/A () Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - (✓) A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing ~~Shift~~/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- () Change each ^{update the} field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
 - (✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

- (/) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- (/) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- (/) Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- (/) All data fields are correctly stored in the database.

Block: TYPE_UNIT_MEASURE_CODE
Title: Valid Units of Measure

Block Description

This table is a code table containing the valid units of measure codes for the system.

Block Test Case

Block Test Results Expected

Field Name: SOURCE_UNIT_MEASURE

Field Description

The field contains an identifier that specifies the unit of measure for the volume field that accompanies this field.

Field Test Case

- () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (/) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- () Accept the default.

N/A

not null fields

Block: TYPE_UNIT_MEASURE_CODE
Title: Valid Units of Measure

Field Name: SOURCE_UNIT_MEASURE

Field Test Case

- ☐ Override the default with a valid entry value.
- ☐ Override the default with an invalid data value.
- ☒ Begin field with a space.
- ☒ Enter data beyond length of field.
 - ☐ If autoskip is in effect, cursor will skip to the next field.
 - ☒ If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ☒ Set keyboard on lower case and enter data. Data should be converted to upper case.
- ☒ Clear an existing field and attempt to update the record.
- ☐ Attempt to store an invalid date:
 - ☐ Enter the day as 00.
 - ☐ Enter the day as 32.
 - ☐ Enter the day as 30 in February.
 - ☐ Enter the day as 31 September.
- ☒ Attempt to enter alpha data.
- ☒ Attempt to enter longer value than allowed.
- ☒ Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- ☒ Attempt to enter negative value. No error should occur.

Field Test Results Expected

- ☒ Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - ☒ Field help describes:
 - ☒ Purpose of the field
 - ☒ LOV reference if available

Block: TYPE_UNIT_MEASURE_CODE
Title: Valid Units of Measure

Field Name: SOURCE_UNIT_MEASURE

Field Test Results Expected

- ☒ () Valid range of values where present
- ☒ () Required indicator when field is NOT NULL
- ☐ () Default value when appropriate

- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

*non
updateable
field*

- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or

N/A

- ☒ () Store a series of records using the SHIFT/^{FB} [duplicate field] key to carry forward field values from the previously stored record.

N/A

*unique
field*

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

- () Use SQL*Plus to verify all data fields are processed correctly.

- () Bypass the field by using the TAB key.
- () No error should occur.
- () The database contains a null value for the field.

*not null
field*

- () Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- () Delete an existing value. The field should be cleared and stored as a null value.

N/A

- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.

N/A

Block: TYPE_UNIT_MEASURE_CODE
Title: Valid Units of Measure

Field Name: SOURCE_UNIT_MEASURE

Field Test Results Expected

- (✓) Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. *FRM 40200 Field prohibits! against update*
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- N/A () Enter an invalid date. The invalid date should be rejected.
- (✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
- (✓) Begin field entry with a space. An error should occur.
- N/A () Database record must contain the proper autogenerated value. Use SQL*Plus to verify contents.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~
Auto All CAPS

Block: TYPE_UNIT_MEASURE_CODE
Title: Valid Units of Measure

Field Name: DESCRIPTION

Field Description

This field contains a 1-30 character value that describes the unit of measure code.

Field Test Case

- not null field*
- () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
 - N/A* () Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
 - () Accept the default.
 - N/A* () Override the default with a valid entry value.
 - () Override the default with an invalid data value.
 - (✓) Begin field with a space.
 - (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
 - (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
 - (✓) Clear an existing field and attempt to update the record.
 - N/A* () Attempt to store an invalid date:
 - () Enter the day as 00.
 - () Enter the day as 32.
 - () Enter the day as 30 in February.
 - () Enter the day as 31 September.
 - (✓) Attempt to enter alpha data.
 - (✓) Attempt to enter longer value than allowed.

Block: TYPE UNIT MEASURE CODE
 Title: Valid Units of Measure

Field Name: DESCRIPTION

Field Test Case

- (✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- (✓) Attempt to enter negative value. No error should occur.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - N/A () LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/^{F6} [duplicate field] key to carry forward field values from the previously stored record.
- (✓) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - (✓) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

Block: TYPE UNIT MEASURE CODE
 Title: Valid Units of Measure

Field Name: DESCRIPTION

Field Test Results Expected

(✓) Use SQL*Plus to verify all data fields are processed correctly.

() Bypass the field by using the TAB key.

() No error should occur.

() The database contains a null value for the field.

() Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

() Delete an existing value. The field should be cleared and stored as a null value.

() Modify each default value with a valid alternative.

() Modify each default value with an invalid alternative.

(✓) Any alphanumeric characters may be entered into the field.

(✓) Delete an existing value.

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. *FRM 10302 Field must be entered*

(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

n/A () Enter an invalid date. The invalid date should be rejected.

(✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.

5A3 (✓) Begin field entry with a space. An error should occur.

n/A () Database record must contain the proper autogenerated value. Use SQL*Plus to verify contents.

533 (✓) Attempt to enter data as lower case. An error should result.

*n/A
not null
field*

Block: TYPE_UNIT_MEASURE_CODE
Title: Valid Units of Measure

Field Name: DESCRIPTION

Field Test Results Expected

Block: TYPE_UNIT_MEASURE_CODE
 Title: Valid Units of Measure

Field Name: CONVERSION_MODIFIER

Field Description

Field Test Case

- N/A* {
- () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field. *not null field*
 - () Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field. *not null field*
 - (✓) Accept the default.
 - (✓) Override the default with a valid entry value.
 - 534 () Override the default with an invalid data value.
 - 535 () Begin field with a space.
 - (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
 - N/A* () Set keyboard on lower case and enter data. Data should be converted to upper case.
 - 536 () Clear an existing field and attempt to update the record.
 - N/A* {
 - () Attempt to store an invalid date:
 - () Enter the day as 00.
 - () Enter the day as 32.
 - () Enter the day as 30 in February.
 - () Enter the day as 31 September.
 - ~~537~~ (✓) Attempt to enter alpha data.
 - (✓) Attempt to enter longer value than allowed.
 - (✓) Attempt to enter decimal value. An error will occur if no

Block: TYPE_UNIT_MEASURE_CODE
 Title: Valid Units of Measure

Field Name: CONVERSION_MODIFIER

Field Test Case

provision has been made for decimal data in the element definition of the table.

(✓) Attempt to enter negative value. No error should occur.

Field Test Results Expected

() Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

() Field help describes:

(✓) Purpose of the field

N/A () LOV reference if available

(✓) Valid range of values where present

(✓) Required indicator when field is NOT NULL

() Default value when appropriate

545 (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

845 (✓) Change the field, committing after each change.

N/A () Change the field where a range of valid values exists:

() Value below valid range should display message COM00101 or COM00103 (if no LOV).

() Value above valid range should display message COM00101 or

537 (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

(✓) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

(✓) The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

(✓) Use SQL*Plus to verify all data fields are processed correctly.

Block: TYPE_UNIT_MEASURE_CODE
 Title: Valid Units of Measure

Field Name: CONVERSION_MODIFIER

Field Test Results Expected

- n/A* {
- () Bypass the field by using the TAB key.
 - () No error should occur.
 - () The database contains a null value for the field.
 - () Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
 - () Delete an existing value. The field should be cleared and stored as a null value.
 - (✓) Modify each default value with a valid alternative.
 - (✓) *Attempt to* Modify each default value with an invalid alternative.
 - 538 () Any alphanumeric characters may be entered into the field.
 - () Delete an existing value.
 - 539 () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - 540 {
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
 - n/A* () Enter an invalid date. The invalid date should be rejected.
 - 541 (✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
 - () ~~Begin field entry with a space. An error should occur.~~ *Do*
 - n/A* () Database record must contain the proper autogenerated value. Use SQL*Plus to verify contents.
 - n/A* () Attempt to enter data as lower case. An error should result.

not null field

Block: TYPE_UNIT_MEASURE_CODE
Title: Valid Units of Measure

Field Name: OBJECT_UNIT_MEASURE

Field Description

Field Test Case

- N/A* ☒ TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- N/A* ☒ Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- N/A* () Accept the default.
- N/A* () Override the default with a valid entry value.
- N/A* () Override the default with an invalid data value.
- 546* () Begin field with a space.
- ☒ Enter data beyond length of field.
- () If autoskip is in effect, cursor will skip to the next field.
- ☒ If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ☒ Set keyboard on lower case and enter data. Data should be converted to upper case.
- 531* ☒ Clear an existing field and attempt to update the record.
- N/A* { () Attempt to store an invalid date:
- () Enter the day as 00.
- () Enter the day as 32.
- () Enter the day as 30 in February.
- () Enter the day as 31 September.
- ☒ Attempt to enter alpha data.
- ☒ Attempt to enter longer value than allowed.
- ☒ Attempt to enter decimal value. An error will occur if no

not null field

Block: TYPE_UNIT_MEASURE_CODE
 Title: Valid Units of Measure

Field Name: OBJECT_UNIT_MEASURE

Field Test Case

provision has been made for decimal data in the element definition of the table.

547 () Attempt to enter negative value. No error should occur.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- () Store a series of records using the SHIFT/^{F6}F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.

Block: TYPE_UNIT_MEASURE_CODE
Title: Valid Units of Measure

Field Name: OBJECT_UNIT_MEASURE

Field Test Results Expected

- () Bypass the field by using the TAB key. } not null field
- () No error should occur.
- () The database contains a null value for the field.
- N/A () Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value. } not null
- () Delete an existing value. The field should be cleared and stored as a null value. } not null
- N/A () Modify each default value with a valid alternative.
- N/A () Modify each default value with an invalid alternative.
- 549 () Any alphanumeric characters may be entered into the field.
- () Delete an existing value. field not updatable
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- 550 () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- N/A () Enter an invalid date. The invalid date should be rejected.
- N/A () Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
- () Begin field entry with a space. An error should occur.
- N/A () Database record must contain the proper autogenerated value. Use SQL*Plus to verify contents.
- () Attempt to enter data as lower case. An error should result. ALL CAPS

Programmer's Bug Report

Application: LIBR Library functions

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 527 Priority: LOW Date: 19-NOV-92

=====

Status : ASSIGNED Date: 19-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : Oracle ROWID: 0000011F.0014.000F

Last Error : -0

Last Msg : FRM-40350 Query caused no records to be retrieved.

Table : TRANSACTION_NEST

Block : TRNNST

Field : TRNNST.PTR

DESCRIPTION

MENU NAME AND TEST PRINTOUT NAME DIFFERENT

Transaction: LI1004U1 Maintain Unit of Measure Codes

```
=====
Comment ID : 531   Priority: LOW   Date: 19-NOV-92
=====
Status      : ASSIGNED   Date: 19-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : TEST USER
System ID   : 10000000681   Oracle ROWID: 00000067.0004.0018
Last Error  : -0
Last Msg    : FRM-40102   Record must be entered or deleted first.
Table       : TYPE_UNIT_MEASURE_CODE
Block       : UNIMEA
Field       : UNIMEA.OBJECT_UNIT_MEASUR
             E
=====
```

DESCRIPTION

FIELD ALLOWED UPDATING OF RECORD - FIELD IS EMPTY SHOULD BE NONNULL.

NO: PROB
✓
TH

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 535 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40407 Transaction complete -- posted records comm

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.CONVERSION_MODIFIE

R

DESCRIPTION

Conv_mod; cannot begin with a space. No error is given. If space added then tab to next field space disappears. Field does have a default

✓
NOT PROB
HA

Programmer's Bug Report

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 536 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000000647 Oracle ROWID: 00000067.0002.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.CONVERSION_MODIFIE

R

DESCRIPTION

This field must be a null field with a default. I was able to clear the field and update data.

A handwritten checkmark is positioned above the words "Not from" which are written in a cursive, handwritten style.

Programmer's Bug Report

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 538 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By: TEST USER

System ID : 10000000681 Oracle ROWID: 00000067.0004.0018

Last Error : -0

Last Msg : FRM-50016 Legal characters are 0-9 - + E .

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.SOURCE_UNIT_MEASUR

E

DESCRIPTION

field is numeric only.*not prob*

Programmer's Bug Report

Transaction: LI1004U1 Maintain Unit of Measure Codes.

=====

Comment ID : 539 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000000627 Oracle ROWID: .00000067.0000.0018

Last Error : -0

Last Msg : FRM-40407 Transaction complete -- posted records comm

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.OBJECT_UNIT_MEASUR

E

DESCRIPTION

Field should not have allowed deletion.*✓ NOT PROB*

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 540 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000000628 Oracle ROWID: 00000067.0001.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.CONVERSION_MODIFIE

R

DESCRIPTION

No error resulted.

Programmer's Bug Report

Transaction: LI1004U1 Maintain Unit of Measure Codes

```
=====
Comment ID : 541   Priority: LOW           Date: 20-NOV-92
=====
Status      : ASSIGNED   Date: 20-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : TEST USER
System ID   : 10000000628      Oracle ROWID: 00000067.0001.0018
Last Error  : -0
Last Msg    : FRM-50016  Legal characters are 0-9 - + E .
Table       : TYPE_UNIT_MEASURE_CODE
Block       : UNIMEA
Field       : UNIMEA.CONVERSION_MODIFIE
              R
=====
```

DESCRIPTION

```
-----
comma not allowed in numeric field
-----
```

✓
NOT PROB

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 543 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000000647 Oracle ROWID: 00000067.0002.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 2 records posted an

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.DESCRPTION

DESCRIPTION

NO ERROR SHOULD OCCUR WHEN BEGINNING THIS FIELD WITH A SPACE A handwritten checkmark is followed by a signature that appears to be "NOT Y 10/20".

Programmer's Bug Report

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 545 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000000695 Oracle ROWID: 00000067.0009.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.CONVERSION_MODIFIE

R

DESCRIPTION

SHOULD NOT BE ABLE TO UPDATE THIS FIELD A handwritten checkmark is drawn above the words "not prob", which are written in a cursive, handwritten style.

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 546 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By: TEST USER

System ID : 10000000647 Oracle ROWID: 00000067.0002.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 2 records posted an

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.OBJECT_UNIT_MEASUR

E

DESCRIPTION

FIELD WILL NOT ALLOW BEGINNING ENTRY WITH A SPACE. FIELD IS UNIQUE.✓
not
prod

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====
Comment ID : 547 Priority: LOW Date: 20-NOV-92
=====

Status : ASSIGNED Date: 20-NOV-92 Class:
Type : PROBLEM
Assigned To: TODD ADAMS
Entered By : TEST USER
System ID : 10000000647 Oracle ROWID: 00000067.0002.0018
Last Error : -0
Last Msg : FRM-40400 Transaction complete -- 2 records posted an
Table : TYPE_UNIT_MEASURE_CODE
Block : UNIMEA
Field : UNIMEA.OBJECT_UNIT_MEASUR
E

DESCRIPTION

Error did occur when attempting to enter negative value into a LOV
field. COM 00101

✓
not
prob

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 548 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By: TEST USER

System ID : 10000000647 Oracle ROWID: 00000067.0002.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 2 records posted an

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.OBJECT_UNIT_MEASUR

E

DESCRIPTION

No mention of LOV in field help. Field should not be updatable because it is a unique field

*Not
prob*

Transaction: LI1004U1 Maintain Unit of Measure Codes

```
=====
Comment ID : 549   Priority: LOW           Date: 20-NOV-92
=====
Status      : ASSIGNED   Date: 20-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : TEST USER
System ID   : 10000000676      Oracle ROWID: 00000067.0003.0018
Last Error  : -0
Last Msg    : FRM-40400 Transaction complete -- 4 records posted an
Table       : TYPE_UNIT_MEASURE_CODE
Block       : UNIMEA
Field       : UNIMEA.OBJECT_UNIT_MEASUR
              E
```

DESCRIPTION

field is a alpha field only

✓
11/20/92
f1603

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 550 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By: TEST USER

System ID: 10000000628 Oracle ROWID: 00000067.0001.0018

Last Error: -0

Last Msg: FRM-40400 Transaction complete -- 1 records posted an

Table: TYPE_UNIT_MEASURE_CODE

Block: UNIMEA

Field: UNIMEA.SOURCE_UNIT_MEASUR

E

DESCRIPTION

field should give an error but doesn't. Field cannot be blank✓
NOT
prob

Transaction: LI1004U1 Maintain Unit of Measure Codes

```
=====
Comment ID : 542   Priority: LOW   Date: 20-NOV-92
=====
Status      : ASSIGNED   Date: 20-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS Larry
Entered By  : TEST USER
System ID   : 10000000647   Oracle ROWID: 00000067.0002.0018
Last Error  : -0
Last Msg    : FRM-40202   Field must be entered.
Table       : TYPE_UNIT_MEASURE_CODE
Block       : UNIMEA
Field       : UNIMEA.SOURCE_UNIT_MEASUR
              E
```

DESCRIPTION

✓ SPELLING ERROR IN TRANSACTION HELP "ALREADYT"

AS
1/24

Transaction: LI1004U1 Maintain Unit of Measure Codes

=====

Comment ID : 544 Priority: LOW Date: 20-NOV-92

=====

Status : ASSIGNED Date: 20-NOV-92 Class:

Type : PROBLEM

Assigned To: ~~TODD ADAMS~~ *Carry*

Entered By : TEST USER

System ID : 10000000647 Oracle ROWID: 00000067.0002.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 2 records posted an

Table : TYPE_UNIT_MEASURE_CODE

Block : UNIMEA

Field : UNIMEA.CONVERSION_MODIFIE

R

DESCRIPTION

A DEFAULT OF 1 IS NOT MENTIONED IN HELP

✓

3.2 LI1006U1 TYPE CONSTITUENT CODE

LEMIS Module Test Checklist Report

Name: TYPE CONSTITUENT CODE
 Short Name: LI1006U1
 Purpose: Maintain constituent codes

Initial Test Date: 11/25/72 JK

Verification Date: 11/29/72 LRP

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: TYPE_CONSTITUENT_CODE
 Title: Types of Constituent Data Records

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

		Dis		Val/	Not	
Data Element	Datatype	Len	Help	LOV	Range	Null
-----	-----	----	----	----	-----	----
TYPE_CONSTITUENT	CHAR	2	()	()	()	()
DESCRIPTION	CHAR	30	()	()	()	()

Module Description

This module is used to add, update or delete constituent codes and their descriptions.

Display 12 lines of data. The constituent code and description are the only data items displayed.

Module Help

This module supports the maintenance and retrieval of constituent type

Module Help

codes. This table maintains a list of valid codes that describe the types of constituents that may occur.

To store a new constituent code:

1. Use the down arrow key to move the cursor to a blank line.

If all of the spaces are filled and more constituent codes are to be added, press the down arrow key to the bottom entry and then continue to press the down arrow key until a blank entry appears. The already entered codes will scroll upward, leaving space for an additional code to be entered.

2. Enter the appropriate constituent code.
3. Enter a 1-30 character description of the code.
4. To add more codes, press the down arrow key to the next blank space and repeat the entry process.
5. Press the COMMIT key (F10) to store the selected codes in the database.

To update an existing code:

1. Press the down arrow key until the cursor displays the entry to be updated.
2. If the code is to be changed, enter the replacement code over the existing code. CAUTION: Changing a constituent code will cause the replacement value to be entered everywhere in the database where the original value was used.
3. If the description is to be changed, TAB to the description field and make the change.
4. Press the COMMIT key (F10) to update the selected codes in the

To delete an existing code:

1. Press the down arrow key until the cursor is located at the code to be deleted.
2. Press the DELETE key. NOTE: If the code is being used by other data in the database, it cannot be deleted. An error message will be displayed.
3. If no error occurs, press the COMMIT key (F10) to delete the code from the database.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

Module Test Plan

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
- ✓ 8 - Secondary testing complete, no problems found.

Developer _____	Date Complete _____
Tester <u>Ly</u>	Date Complete <u>11/29</u>

When problems are found:

1. The tester passes the checklist to the project manager for

Module Test Plan

review and action.

2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.

3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.

4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.

Project Manager [Signature] Date Complete 11/29/92

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.

System Architect [Signature] Date Complete 11/29/92

Module Test Plan

Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - (✓) Transaction help describes:
 - (✓) Purpose of the transaction
 - (✓) Functions performed by the transaction
 - NA Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - (✓) A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - NA Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - (✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - (✓) A query using a prefix % wildcard retrieves the correct data. ~~Verify by running the same query under SQL*Plus.~~
 - (✓) A query using both prefix and suffix % wildcards retrieves the

yes, but displays default of 'CO'

de Fault of 'CO'

correct data. ~~Verify by running the same query through SQL*Plus.~~

- (✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
- (✓) Change each field individually, committing after each change.
- NK (✓) Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
 - (✓) The selected record is deleted from the database. ~~Verify by using SQL*Plus to attempt to retrieve the record.~~
 - () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - NK () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - () All data fields are correctly stored in the database.

Block: TYPE_CONSTITUENT_CODE
Title: Types of Constituent Data Records

Block Description

Block Test Case

Block Test Results Expected

Field Name: TYPE_CONSTITUENT

Field Description

Block: TYPE_CONSTITUENT_CODE
 Title: Types of Constituent Data Records

Field Name: TYPE_CONSTITUENT

Field Test Case

- (✓) Accept the default.
- (✓) Override the default with a valid entry value.
- (✓) Override the default with an invalid data value. *what's invalid?*
- (✓) Begin field with a space. 0<
- (✓) Enter data beyond length of field.
 - ✗ If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record. *won't allow*

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).

NA

Block: TYPE_CONSTITUENT_CODE
 Title: Types of Constituent Data Records

Field Name: TYPE_CONSTITUENT

Field Test Results Expected

- () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/~~E~~^S [duplicate field] key to carry forward field values from the previously stored record. *won't allow*
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- NA () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- (✓) Modify each default value with a valid alternative.
- (✓) Modify each default value with an invalid alternative. *only invalid is blank*
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. *diff. msg*
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered. *yes*
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~
Converts to upper case

Block: TYPE_CONSTITUENT_CODE
Title: Types of Constituent Data Records

Field Name: DESCRIPTION

Field Description

Field Test Case

- () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- () Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - NA If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

won't allow blank

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - NA LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - NA Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and~~

won't allow blanks

Block: TYPE_CONSTITUENT_CODE
 Title: Types of Constituent Data Records

Field Name: DESCRIPTION

Field Test Results Expected

~~after the changes were made.~~

- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - NA () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/^S [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- NA () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- (✓) Bypass the field by using the TAB key.
- NA () No error should occur.
- () The database contains a null value for the field.
- () Enter data into the field, backspace the data out of the field and NA TAB to the next field. The database should contain a null value.
- () Delete an existing value. The field should be cleared and stored NA as a null value.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. OK
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered. YES

Block: TYPE_CONSTITUENT_CODE
Title: Types of Constituent Data Records

Field Name: DESCRIPTION

Field Test Results Expected

- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- (✓) Attempt to enter data as lower case. An error should result.

Transaction: LI1006U1 Maintain Constituent Type Codes

=====

Comment ID : 469 Priority: LOW Date: 04-NOV-92

=====

Status : ASSIGNED Date: 04-NOV-92 Class:

Type : COMMENT

Assigned To: TODD ADAMS

Entered By : MARY KORMAN

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-41800 List of values not available for this field

Table : TYPE_CONSTITUENT_CODE

Block : TYPCON

Field : TYPCON.TYPE_CONSTITUENT

DESCRIPTION

Ken -

Given LI1006U1 [Maintain Constituent Codes], yellow-black message said "Enter the two-character constituent type code; see LOV ..". Did LOV [list of values] function key. Received error INFO-FRM-41800 : List of values not available.....???

?1 : Who's responsibility is it to key in all LOV values ?

?2 : Who's responsibility is it to keep list of all LEMIS fields with LOV functionality ?

?3 : Users might find above answers + current list of LEMIS fields with LOV functionality with LOV values "friendly" appendix in their Users Guide. Okay if I query users ?

Thanks -

Mary

CORRECTIVE ACTION

11/04/92 - Todd: There should not be a default here. The hint message needs to have the LOV stuff removed.✓
FIXED11-22-92
✓

OK

11/29

Lm

3.3 LI1008U1 BOTTLE CODE TABLE MAINTENANCE

LEMIS Module Test Checklist Report

Name: BOTTLE CODE TABLE MAINTENANCE
 Short Name: LI1008U1
 Purpose: Bottle/container code table maintenance
 Initial Test Date: 11/30/92 JK
 Verification Date: 11/30/92 RDA (no bugs)

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: TYPE_BOTTLE_CODE
 Title: Container Type Code Table Maintenance

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_BOTTLE	CHAR	5	()	()	()	()	()
DESCRIPTION	CHAR	30	()	()	()	()	()
SIZE_BOTTLE	NUMBER	5	()	()	()	()	()
UNIT_MEASURE	CHAR	5	()	()	()	()	()
TYPE_BOTTLE_COMPOSITIO	CHAR	2	()	()	()	()	()

Module Description

The transaction displays the table code and the code description. The date added and update person, where present, are not displayed. These fields are set by the software. The operator may add new entries. Entries may be modified or deleted as long as the entry being modified does not exist elsewhere in the database.

Module Description

Display 12 lines of data. The bottle code and description are the only data items displayed.

Module Help

This module supports the maintenance and retrieval of bottle type codes. This table maintains a list of valid codes that describe the types of bottles that may occur.

To store a new bottle type code:

1. Use the down arrow key to move the cursor to a blank line.

If all of the spaces are filled and more bottle type codes are to be added, press the down arrow key to the bottom entry and then continue to press the down arrow key until a blank entry appears. The already entered codes will scroll upward, leaving space for an additional code to be entered.

2. Enter the appropriate bottle type code.
3. Enter a 1-30 character description of the code.
4. To add more codes, press the down arrow key to the next blank space and repeat the entry process.
5. Press the COMMIT key (F10) to store the selected codes in the database.

To update an existing code:

1. Press the down arrow key until the cursor displays the entry to be updated.
2. If the code is to be changed, enter the replacement code over the existing code. CAUTION: Changing a bottle type code will cause the replacement value to be entered everywhere in the database where the original value was used.
3. If the description is to be changed, TAB to the description field and make the change.
4. Press the COMMIT key (F10) to update the selected codes in the

To delete an existing code:

1. Press the down arrow key until the cursor is located at the code to be deleted.
2. Press the DELETE key. NOTE: If the code is being used by other data in the database, it cannot be deleted. An error message will be displayed.
3. If no error occurs, press the COMMIT key (F10) to delete the code from the database.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.

Developer _____ Date Complete _____

Module Test Plan

- () 8 - Secondary testing complete, no problems found.
 Tester _____ Date Complete _____

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.
 () 10 - System testing complete, problems found.
 Tester _____ Date Complete _____
 () 11 - Problem resolution in progress.
 () 12 - Problem resolution complete.
 Developer _____ Date Complete _____
 () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

- () 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.
 () Reviews tests results.
 () Reviews problem reports to ensure all corrections done.
 () Updates NOTES text to reflect testing results.
 Project Manager SAJ Date Complete 11/30/92
 () 20 - The transaction is placed into the production

Module Test Plan

environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.

System Architect AS Date Complete 11/30/92

Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
- (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
- 626 (✓) Transaction help describes:
 - (✓) Purpose of the transaction
 - (✓) Functions performed by the transaction
 - NR (✓) Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - (✓) A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
- 638 (✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- () Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing SHIFT/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction

- (✓) ready for a query.
- (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
- (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- (✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
- (✓) Change each field individually, committing after each change.
- (NR) Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
 - () The selected record is deleted from the database. ~~Verify by using SQL*Plus to attempt to retrieve the record.~~
 - () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - () All data fields are correctly stored in the database.

Block: TYPE_BOTTLE_CODE

Title: Container Type Code Table Maintenance

Block Description

This table provides a validation of the data item TYPE_BOTTLE wherever it appears in the database.

Block Test Case

Block Test Results Expected

Block: TYPE_BOTTLE_CODE
Title: Container Type Code Table Maintenance

Field Name: TYPE_BOTTLE

Field Description

This is a code used in the container record to define the type of container.

Field Test Case

- () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- () Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.

Wrong Test

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - (✗) If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

- (✓) Clear an existing field and attempt to update the record.

wait allow

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✗) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✗) Default value when appropriate

- (✓) Change the field in an existing record. Verify that the change

Block: TYPE_BOTTLE_CODE

Title: Container Type Code Table Maintenance

Field Name: TYPE_BOTTLE

Field Test Results Expected

occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or

- (✓) Store a series of records using the SHIFT/~~PT~~^G [duplicate field] key to carry forward field values from the previously stored record.

duplicates not allowed

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

- () Use SQL*Plus to verify all data fields are processed correctly.

- () Bypass the field by using the TAB key.
 - () No error should occur.
 - () The database contains a null value for the field.

- () Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

- () Delete an existing value. The field should be cleared and stored as a null value.

Wrong Test

- (✓) Any alphanumeric characters may be entered into the field.

- (✓) Attempt to enter data as lower case. An error should result.

Block: TYPE_BOTTLE_CODE

Title: Container Type Code Table Maintenance

Field Name: TYPE_BOTTLE

Field Test Results Expected

- (✓) Delete an existing value.
- (✓) For NOT NULL fields, the COM00107 message *diff. msg* [field value may not be deleted] is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: TYPE_BOTTLE_CODE

Title: Container Type Code Table Maintenance

Field Name: DESCRIPTION

Field DescriptionField Test Case

() TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.

() Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.

Wrong Test

(✓) Begin field with a space.

(✓) Enter data beyond length of field.

(✓) If autoskip is in effect, cursor will skip to the next field.

(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

(✓) Clear an existing field and attempt to update the record. won't allow

Field Test Results Expected

(✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

(✓) Field help describes:

(✓) Purpose of the field

(✓) LOV reference if available

(✓) Valid range of values where present

(✓) Required indicator when field is NOT NULL

(✓) Default value when appropriate

(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and~~

Block: TYPE_BOTTLE_CODE

Title: Container Type Code Table Maintenance

Field Name: DESCRIPTION

Field Test Results Expected

after the changes were made.

- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - NA () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
 - () Bypass the field by using the TAB key.
 - () No error should occur.
 - () The database contains a null value for the field.
 - () Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
 - () Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Delete an existing value.
- (✓) Attempt to enter data as lower case. An error should result.

Wrong Test

Block: TYPE_BOTTLE_CODE

Title: Container Type Code Table Maintenance

Field Name: DESCRIPTION

Field Test Results Expected

- (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: TYPE_BOTTLE_CODE
Title: Container Type Code Table Maintenance

Field Name: SIZE_BOTTLE

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Clear an existing field and attempt to update the record.
- (✓) Attempt to enter alpha data.
- (✓) Attempt to enter longer value than allowed.
- 639 OK (✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- (✓) Attempt to enter negative value. No error should occur.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (NA) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (NA) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~

Block: TYPE_BOTTLE_CODE
 Title: Container Type Code Table Maintenance

Field Name: SIZE_BOTTLE

Field Test Results Expected

- (✓) Change the field, committing after each change.
- (✓) Change the field where a range of valid values exists:
 - NA () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- NA () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
- (✓) Begin field entry with a space. An error should occur. → NO, but it left justifies number
- (✓) Delete an existing value.
- NA For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you

Block: TYPE_BOTTLE_CODE

Title: Container Type Code Table Maintenance

Field Name: SIZE_BOTTLE

Field Test Results Expected

that the field data must be entered.

() Enter data into the field, backspace the data out of the field and
attempt to TAB to the next field. An error should result.

NA

Block: TYPE_BOTTLE_CODE
 Title: Container Type Code Table Maintenance

Field Name: UNIT_MEASURE

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Accept the default.
- (✓) Override the default with a valid entry value.
- (✓) Override the default with an invalid data value. *NO, won't allow*
- (✓) Begin field with a space. *won't allow*
- (✓) Enter data beyond length of field.
 - (✓) If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record. *allows blank*

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- () Field help describes:
 - (✓) Purpose of the field
 - () LOV reference if available
 - (✓) Valid range of values where present

Block: TYPE_BOTTLE_CODE
Title: Container Type Code Table Maintenance

Field Name: UNIT_MEASURE

Field Test Results Expected

- (✓) Required indicator when field is NOT NULL
- (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Modify each default value with a valid alternative.
- (✓) Modify each default value with an invalid alternative.

Block: TYPE_BOTTLE_CODE
 Title: Container Type Code Table Maintenance

Field Name: UNIT_MEASURE

Field Test Results Expected

- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. An error should result.
- (✓) Delete an existing value.

Wt For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

*-no must match
 low*

*wrong
 test*

Block: TYPE_BOTTLE_CODE
Title: Container Type Code Table Maintenance

Field Name: TYPE_BOTTLE_COMPOSITION

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Begin field with a space. *invalid value*
- (✓) Enter data beyond length of field.
 - (✓) If autoskip is in effect, cursor will skip to the next field.
 - ~~(NA)~~ If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - ~~(NA)~~ Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

*must match LOV
no mention of LOV*

641

Block: TYPE_BOTTLE_CODE

Title: Container Type Code Table Maintenance

Field Name: TYPE_BOTTLE_COMPOSITION

Field Test Results Expected

- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- (✓) Store a series of records using the SHIFT/F⁹ [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- see 641 (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. An error should result.

Transaction: LI1008U1 Bottle Code Table Maintenance

=====
Comment ID : 638 Priority: LOW Date: 30-NOV-92
=====

Status : ASSIGNED Date: 30-NOV-92 Class:
Type : PROBLEM
Assigned To: TODD ADAMS
Entered By : LARRY TOWNER
System ID : Oracle ROWID:
Last Error : -0
Last Msg : FRM-40400 Transaction complete -- 1 records posted an
Table : TYPE_BOTTLE_CODE
Block : BOTTYP
Field : BOTTYP.TYPE_BOTTLE

DESCRIPTION

unit of measure field defaults to 'liter' when opening up a new line.

NOT PROB

u/30/92

✓

Application: LIBR Library functions

Transaction: LI1008U1 Bottle Code Table Maintenance

=====
Comment ID : 639 Priority: LOW Date: 30-NOV-92
=====

Status : ASSIGNED Date: 30-NOV-92 Class:
Type : COMMENT
Assigned To: TODD ADAMS
Entered By : LARRY TOWNER
System ID : 10000000631 Oracle ROWID: 0000003F.0000.0018
Last Error : -0
Last Msg : FRM-40400 Transaction complete -- 1 records posted an
Table : TYPE_BOTTLE_CODE
Block : BOTTP
Field : BOTTP.TYPE_BOTTLE

DESCRIPTION

Size field: will accept a decimal value eg 1.25 but when record is
retrieved from table, the value is rounded to nearest whole number.

NOT prob

✓

11/30/92

3.4 LI1009U1 PRESERVATIVE CODE TABLE MAINTENANCE

LEMIS Module Test Checklist Report

Name: PRESERVATIVE_CODE TABLE MAINTENANCE
 Short Name: LI1009U1
 Purpose: The module maintains the preservative code table
 Initial Test Date: 11-17-92 ~~11-17-92~~ LC
 Verification Date: 11-17-92 LC/TR/mk

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: TYPE PRESERVATIVE_CODE
 Title: List of Preservatives

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
PRESERVATIVE	CHAR	9	()	()	()	()	()
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

The transaction displays the table code and the code description. The date added and update person, where present, are not displayed. These fields are set by the software. The operator may add new entries, update existing entries, but may not delete any entry.

Display 12 lines of data. The preservative code and description are the only data items displayed.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

Test status conditions are:

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
 - Developer _____ Date Complete _____
- () 3 - Initial unit testing complete.
 - Developer _____ Date Complete _____

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.

Module Test Plan

Tester _____ Date Complete _____
 () 6 - Problem resolution in progress.
 () 7 - Problem resolution complete.
 Developer _____ Date Complete _____
 () 8 - Secondary testing complete, no problems found.
 Tester fe Date Complete 11/17/92

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Test status conditions are:

() 9 - System testing in progress.
 () 10 - System testing complete, problems found.
 Tester _____ Date Complete _____
 () 11 - Problem resolution in progress.
 () 12 - Problem resolution complete.
 Developer _____ Date Complete _____
 () 13 - System testing complete, no problems found.
 Tester _____ Date Complete _____

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and

Module Test Plan

determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.

Project Manager ZJS Date Complete 11/17/92

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.

System Architect JS Date Complete 11/17/92

Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - (✓) Transaction help describes:
 - (✓) Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - ~~NR~~ The insert key clears all blocks of the entry screen. N/A
 - (✓) A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - ~~NR~~ Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - ~~NR~~ All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing ~~SHIFT~~/F12 clears the record and permits entering a

- (✓) second query.
- (✓) The second query successfully retrieves the desired data.
- (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- (✓) Change each ~~field~~ ^{OR MAX VALUE} in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change each field individually, committing after each change.
 - (✓) Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
 - (✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
 - (✓) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - (✓) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - (✓) Use SQL*Plus to verify all data fields are stored correctly, including ID_keys, date-updated, owner role, and any other autogenerated fields.
 - (✓) All data fields are correctly stored in the database.

* SKIP SQL FOR SMALL TABLES

* DO SHRT - FIZ INSTEAD TO VERIFY

SQLPLUS

* COPE TABLES - JUST CHECK JOURNAL TABLE FOR INSERTS, UPDATES, AND MATCH # OF DELETES.

* ON COPE TABLES JUST VERIFY DATE & FIELDS FROM SCREEN

Block: TYPE_PRESERVATIVE_CODE
Title: List of Preservatives

Block Description

Block: TYPE PRESERVATIVE_CODE
 Title: List of Preservatives

Block Test Case

Block Test Results Expected

Field Name: PRESERVATIVE

Field Description

Field Test Plan

- (✓) Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- (✓) Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- (✓) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- (✓) Store a series of records using the SHIFT/F6 [duplicate field] key to carry forward field values from the previously stored record.
- (✓) Modify each default value with a valid alternative.
- (✓) Modify each default value with an invalid alternative.
- (✓) Test LOV where that capability is defined.
 - () Select the first entry in the LOV.
 - () Select the last entry in the LOV.
 - () Use the FIND option and select an entry.
- (✓) Enter data beyond the length of the field.
- (✓) Set the keyboard on lower case and enter data. Data should be converted to upper case.
- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.

DVR

Block: TYPE_PRESERVATIVE_CODE
Title: List of Preservatives

Field Name: PRESERVATIVE

Field Test Plan

- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- () ~~Attempt to enter data as lower case. DVP~~
- () ~~Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL. DVP~~
- () ~~Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur. DVP~~
- (✓) Skip the field using the TAB key.
- () ~~Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value. DVP~~
- () ~~Delete an existing value. The field should be cleared and stored. DVP~~

Field Test Case

- ~~10/18~~ TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field. N/A TO NOT NULL FIELD
- ~~12/18~~ Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field. N/A TO NOT NULL FIELD
- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - (✓) If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () ~~Set keyboard on lower case and enter data. Data should be converted to upper case. DVP~~
- () Clear an existing field and attempt to update the record. N/A FOR NOT NULL FIELDS

Block: TYPE_PRESERVATIVE_CODE
 Title: List of Preservatives

Field Name: PRESERVATIVE

Field Test Results Expected

() Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

() Field help describes:

- () Purpose of the field
- () LOV reference if available
- ~~() Valid range of values where present~~
- () Required indicator when field is NOT NULL
- ~~() Default value when appropriate~~

() Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

() Change the field, committing after each change.

() Change the field where a range of valid values exists:

- () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- () Value above valid range should display message COM00101 or COM00103 (if no LOV).

~~() Store a series of records using the SHIFT/EO [duplicate field] key to carry forward field values from the previously stored record~~

~~() Display the JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.~~

~~() The JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.~~

~~() Use SQL*Plus to verify all data fields are processed correctly.~~

() Bypass the field by using the TAB key.

- () No error should occur.
- () The database contains a null value for the field.

Block: TYPE_PRESERVATIVE_CODE
Title: List of Preservatives

Field Name: PRESERVATIVE

Field Test Results Expected

- () Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- () Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) ~~Delete~~ ^{CLERK} an existing value.
 - () For NOT NULL fields, the ~~COM00107 message [field value may not be deleted]~~ ^{FIELD MUST BE ENTERED} is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- ~~() Attempt to enter data as lower case. An error should result.~~
- ~~() Delete an existing value.~~
 - ~~() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.~~
- ~~() Attempt to bypass the field. An error message should tell you that the field data must be entered.~~
- ~~() Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.~~

N/A for NOT NULL

N/A for NOT NULL

DUP

DUP

DUPS

Block: TYPE_PRESERVATIVE_CODE
Title: List of Preservatives

Field Name: DESCRIPTION

Field Description

* Warning: This module is developed from function SOE53
* which only has some of its etities implemented as tables
* This suggests that part of the module is a manual process

Field Test Plan

- (✓) Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- (✓) Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made. SEE P116 R 5- DVP
- (✓) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL. → F8 + commit
- (✓) Store a series of records using the SHIFT/F6 [duplicate field] key to carry forward field values from the previously stored record.
- N/A Modify each default value with a valid alternative.
- N/A Modify each default value with an invalid alternative.
- () Test LOV where that capability is defined.
- N/A () Select the first entry in the LOV.
- N/A () Select the last entry in the LOV.
- () Use the FIND option and select an entry.
- (✓) Enter data beyond the length of the field.
- (✓) Set the keyboard on lower case and enter data. Data should be converted to upper case.
- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL. DVP

Block: TYPE_PRESERVATIVE_CODE
Title: List of Preservatives

Field Name: DESCRIPTION

Field Test Plan

- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur. *FOR NOT NULL*
- () ~~Attempt to enter data as lower case.~~ *DUPS*
- () ~~Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.~~ *DUP*
- () ~~Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.~~
- (✓) Skip the field using the TAB key.
- () Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value. *FOR NULL FIELDS*
- () Delete an existing value. The field should be cleared and stored. *FOR NULL VALUES.*

Field Test Case

- () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field. *FOR NULLS ONLY*
- () Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field. *FOR NULLS ONLY*
- (✓) Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field. *DUP*
- () ~~Set keyboard on lower case and enter data. Data should be converted to upper case.~~ *DUP*
- () ~~Clear an existing field and attempt to update the record.~~

Block: TYPE_PRESERVATIVE_CODE
Title: List of Preservatives

Field Name: DESCRIPTION

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- (✓) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - (✓) The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- (✓) Use SQL*Plus to verify all data fields are processed correctly.
 - (✓) All data fields are correctly stored in the database.
- () Bypass the field by using the TAB key.
 - () No error should occur.
 - () The database contains a null value for the field.

Block: TYPE_PRESERVATIVE_CODE
Title: List of Preservatives

Field Name: DESCRIPTION

Field Test Results Expected

- ☒ Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- ☐ Delete an existing value. The field should be cleared and stored as a null value. *for NULL fields only*
- ☒ Any alphanumeric characters may be entered into the field.
- ☒ Delete an existing value.
 - ☐ For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- ☒ Attempt to bypass the field. An error message should tell you that the field data must be entered.
- ☒ Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- ☒ Attempt to enter data as lower case. An error should result. *DUP*
- ☐ Delete an existing value.
 - ☐ For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- ☐ Attempt to bypass the field. An error message should tell you that the field data must be entered. *DUP*
- ☒ Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

3.5 LI1011U1 MAINTAIN TYPE SAMPLE MATRIX CODE

LEMIS Module Test Checklist Report

Name: MAINTAIN TYPE SAMPLE MATRIX CO
 Short Name: LI1011U1
 Purpose: Maintain the sample matrix type codes.

Initial Test Date: 11/30/92 JK

Verification Date: 11/30/92 RJA (No Bugs)

Help	Disp Fkeys	Security			
		Ins	Del	Query	Upd
()	()	()	()	()	()

Block: TYPE_SAMPLE_MATRIX_CODE
 Title: Sample Matrix

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_SAMPLE_MATRIX	CHAR	2	()	()	()	()	()
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

This form is used to add, update or delete sample matrix codes.
 You may not update or delete codes that are being used elsewhere
 in the database.

Display 12 lines of data. The matrix code and description are the
 only data items displayed.

Module Help

This module supports the maintenance and retrieval of sample matrix codes. This table maintains a list of valid codes that describe the types of sample matrix that may occur.

To store a new sample matrix code:

1. Use the down arrow key to move the cursor to a blank line.

If all of the spaces are filled and more sample matrix codes are to be added, press the down arrow key to the bottom entry and then continue to press the down arrow key until a blank entry appears. The already entered codes will scroll upward, leaving space for an additional code to be entered.

2. Enter the appropriate sample matrix code.
3. Enter a 1-30 character description of the code.
4. To add more codes, press the down arrow key to the next blank space and repeat the entry process.
5. Press the COMMIT key (F10) to store the selected codes in the database.

To update an existing code:

1. Press the down arrow key until the cursor displays the entry to be updated.
2. If the code is to be changed, enter the replacement code over the existing code. CAUTION: Changing a sample matrix code will cause the replacement value to be entered everywhere in the database where the original value was used.
3. If the description is to be changed, TAB to the description field and make the change.
4. Press the COMMIT key (F10) to update the selected codes in the

To delete an existing code:

1. Press the down arrow key until the cursor is located at the code to be deleted.
2. Press the DELETE key. NOTE: If the code is being used by other data in the database, it cannot be deleted. An error message will be displayed.
3. If no error occurs, press the COMMIT key (F10) to delete the code from the database.

Module Test Plan

TEST PLAN

Module Test Plan

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.

Developer _____	Date Complete _____
-----------------	---------------------
- () 8 - Secondary testing complete, no problems found.

Tester _____	Date Complete _____
--------------	---------------------

When problems are found:

Module Test Plan

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

- () 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.
 - () Reviews tests results.
 - () Reviews problem reports to ensure all corrections done.
 - () Updates NOTES text to reflect testing results.
 - Project Manager LHV Date Complete 11/30/92
- () 20 - The transaction is placed into the production environment.
 - () Software files in production directories.
 - () DMR 550 updated to add transaction.
 - () DMR 900 updated to add transaction.
 - () Transaction added to production usage menu.

Module Test PlanSystem Architect ADate Complete 11/30/92Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
- (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - (✓) Transaction help describes:
 - (✓) Purpose of the transaction
 - (✓) Functions performed by the transaction
 - (✓) Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - (✓) A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - (✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - (✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing SHIFT/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.

- (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through ~~SQL*Plus.~~
- (✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
- (✓) Change each field individually, committing after each change.
- (N/A) Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
- (N/A) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- () All data fields are correctly stored in the database.

Block: TYPE_SAMPLE_MATRIX_CODE
Title: Sample Matrix

Block Description

Block Test Case

Block Test Results Expected

- () Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
- () Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]

dup test

Block: TYPE_SAMPLE_MATRIX_CODE
 Title: Sample Matrix

Block Test Results Expected

- () Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - () Transaction help describes:
 - () Purpose of the transaction
 - () Functions performed by the transaction
 - () Where multiple page, what happens on each page
- () Add a new record by using the insert key.
 - () The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- () Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - () A new data record can be successfully added.
- () Query a single record.
 - () A single record is retrieved.
 - () Pressing SHIFT/F12 clears the record and permits entering a second query.
 - () The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
 - () Pressing F12 clears the screen and makes the transaction ready for a query.
 - () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through
- () Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

dup test

Block: TYPE_SAMPLE_MATRIX_CODE

Title: Sample Matrix

Block Test Results Expected

- () Change each field individually, committing after each change.
- () Change each field, as per the field test plan, for both valid and invalid values.
- () Delete an existing record.
 - () The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- () All data fields are correctly stored in the database.

dup test

Field Name: TYPE_SAMPLE_MATRIX

Field Description

This is a codes table identifying the types of sample "matrix" that may occur with a sample.

Field Test Case

- () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- () Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Accept the default.

Wrong test

Block: TYPE_SAMPLE_MATRIX_CODE
 Title: Sample Matrix

Field Name: TYPE_SAMPLE_MATRIX

Field Test Case

- (✓) Override the default with a valid entry value.
- (✓) Override the default with an invalid data value. *blank is only invalid*
- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - (NA) If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record. *won't allow*

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (NA) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
 - (✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or

Block: TYPE_SAMPLE_MATRIX_CODE
Title: Sample Matrix

Field Name: TYPE_SAMPLE_MATRIX

Field Test Results Expected

(✓) Store a series of records using the SHIFT/^{F6}P1 [duplicate field] key to carry forward field values from the previously stored record.

*won't work -
dups not allowed
in Key
Field*

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

NA () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

() Bypass the field by using the TAB key.

() No error should occur.

() The database contains a null value for the field.

*wrong
Test*

() Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

() Delete an existing value. The field should be cleared and stored as a null value.

(✓) Modify each default value with a valid alternative.

(✓) Modify each default value with an invalid alternative. *only invalid is blank*

(✓) Any alphanumeric characters may be entered into the field.

(✓) Delete an existing value.

(NA) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

(✓) Attempt to enter data as lower case. An error should result.

Block: TYPE_SAMPLE_MATRIX_CODE
Title: Sample Matrix

Field Name: TYPE_SAMPLE_MATRIX

Field Test Results Expected

Block: TYPE_SAMPLE_MATRIX_CODE
Title: Sample Matrix

Field Name: DESCRIPTION

Field Description

This is a codes table identifying the types of sample "matrix" that may occur with a sample.

Field Test Case

- () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- () Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.

wrong test

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - (✓) If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record. *won't allow*

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change

Block: TYPE_SAMPLE_MATRIX_CODE

Title: Sample Matrix

Field Name: DESCRIPTION

Field Test Results Expected

occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- NA () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
 - () Bypass the field by using the TAB key.
 - () No error should occur.
 - () The database contains a null value for the field.
 - () Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
 - () Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Delete an existing value.
 - (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you

wrong
Test

Block: TYPE_SAMPLE_MATRIX_CODE
Title: Sample Matrix

Field Name: DESCRIPTION

Field Test Results Expected

that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

(✓) Attempt to enter data as lower case. An error should result.

Transaction: LI1011U1 Maintain Type Sample Matrix Codes

=====

Comment ID : 642 Priority: LOW Date: 30-NOV-92

=====

Status : ASSIGNED Date: 30-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : LARRY TOWNER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40102 Record must be entered or deleted first.

Table : TYPE_SAMPLE_MATRIX_CODE

Block : TYPSMC

Field : TYPSMC.TYPE_SAMPLE_MATRIX

DESCRIPTION

Type_sample_matrix field defaults to WA when attempting to enter a new record. Is this correct? In addition, if you try to commit the record with 'WA', program tell you it's a duplicate value.*NOT PROBLEM**11/30/92*

3.6 LI5051Q1 CONSTITUENT PICK LIST

Testing for this module was performed during the testing of
LI5055U1 ASSIGN CONSTITUENTS TO METHODS.

3.7 LI5051U1 MAINTAIN CONSTITUTENTS

LEMIS Module Test Checklist Report

Name: MAINTAIN CONSTITUENTS
 Short Name: LI5051U1
 Purpose: Maintain the library of constituents
 Initial Test Date: 11/22/92 *LC*
 Verification Date: 11/29/92 *F4*

	Disp	Security			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: CONSTITUENT
 Title: Constituents

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_CONSTITUENT	CHAR	30	()	()	()	()	()
NAME_CON_LONG	CHAR	60	()	()	()	()	()
TYPE_CONSTITUENT	CHAR	2	()	()	()	()	()

Block: CONSTITUENT
Title: Constituents

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_CONSTITUENT	CHAR	30	()	()	()	()	()
TYPE_CONSTITUENT	CHAR	2	()	()	()	()	()
NAME_CON_LONG	CHAR	60	()	()	()	()	()
MOLECULAR_FORMULA	CHAR	50	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: CONSTITUENT_NEST
Title: Associated Constituents

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
RELATIONSHIP	CHAR	3	()	()	()	()	()

Block: METHOD
Title: Laboratory Methods

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	30	()	()	()	()	()
TITLE_METHOD	CHAR	40	()	()	()	()	()

Block: METHOD_CONSTITUENT
Title: Methods Testing for Constituent

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Block: TYPE_CONSTITUENT_CODE

Title:

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DESCRIPTION	CHAR	30	()	()	()	()	()

Block: TYPE_CONSTITUENT_CODE

Title:

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

This module provides the ability to store constituents in the database and associate them with one or more analytical methods that are used to detect the constituent.

A constituent may be associated with other constituents through the nest table. The default relationship is SUB (subordinate) for constituents that are composed of other constituents. Also permitted are ALT (alternate).

A constituent will be associated with one or more methods. The constituent may be associated only with those methods of type ME and AP. Where associated with the method type AP, the relationship code will be either PLS (plus) or MIN (minus). When method type ME is associated with a constituent, the relationship code is REL (related).

The data validation level defaults to 9. The ident-owner value defaults to 'library'.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

Module Test Plan

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

Test status conditions are:

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- Developer AA Date Complete 11/25/92
- () 3 - Initial unit testing complete.
 - Developer AA Date Complete 11/25/92

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
 - Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
 - Developer _____ Date Complete _____
- () 8 - Secondary testing complete, no problems found.
 - Tester JC Date Complete 11/22/92

Module Test Plan

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Test status conditions are:

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.
Tester _____ Date Complete _____

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

- () 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.
 - () Reviews tests results.
 - () Reviews problem reports to ensure all corrections done.
 - () Updates NOTES text to reflect testing results.
 Project Manager La Date Complete 4/29/92
- () 20 - The transaction is placed into the production environment.

Module Test Plan

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.

System Architect JA Date Complete 11/24/92

Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - (✓) Transaction help describes:
 - () Purpose of the transaction
 - (✓) Functions performed by the transaction
 - (✓) Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - (✓) A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - () All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing SHIFT/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction ready for a query.

- (/) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
- (/) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
- (/) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- (/) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (/) Change each field individually, committing after each change.
 - (/) Change each field, as per the field test plan, for both valid and invalid values.
- (/) Delete an existing record.
 - (/) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- (/) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - (/) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- (/) Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - (/) All data fields are correctly stored in the database.

Block: CONSTITUENT
Title: Constituents

Block Description

The constituent table identifies elements, chemical, compounds, and other items that may be found during the analysis of samples. A constituent may be made up of other constituents.

The constituent table is considered a library table. Once loaded with the baseline list of constituents for the site, it is anticipated that little further update activity will occur.

Block: CONSTITUENT
Title: Constituents

Block Description

All constituents are considered to have a validation level of 9 (fully validated). Role access control is not exercised on constituents.

Block Test Case

Block Test Results Expected

Field Name: IDENT_CONSTITUENT

Field Description

Field Test Plan

- 55a (✓) Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- 56D () Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- (✓) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- () ~~Attempt~~ Store a series of records using the SHIFT/~~F7~~ ⁵⁶ [duplicate field] key to carry forward field values from the previously stored record.
- N/A { () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Test LOV where that capability is defined.
 - () Select the first entry in the LOV.
 - () Select the last entry in the LOV.
 - () Use the FIND option and select an entry.
- N/A (✓) Enter data beyond the length of the field.
- (✓) Set the keyboard on lower case and enter data. Data should be converted to upper case.

Block: CONSTITUENT
Title: Constituents

Field Name: IDENT_CONSTITUENT

Field Test Plan

- (/) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- (/) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- (/) Attempt to enter data as lower case.

Field Test Case

- (/) Begin field with a space.
- (/) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (/) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (/) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (/) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (/) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (/) Field help describes:
 - (/) Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - (/) Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

Dep

Block: CONSTITUENT
Title: Constituents

Field Name: IDENT_CONSTITUENT

Field Test Results Expected

- () Change the field, committing after each change. *Dup*
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/E7 [duplicate field] key to carry forward field values from the previously stored record. *Dup*
- (/) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - (/) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- (/) Use SQL*Plus to verify all data fields are processed correctly.
- (/) All data fields are correctly stored in the database.
- (/) Any alphanumeric characters may be entered into the field.
- (/) Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. *FRM 40202*
- (/) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (/) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- (/) Attempt to enter data as lower case. ~~An error should result.~~

Block: CONSTITUENT
Title: Constituents

Field Name: TYPE_CONSTITUENT

Field Description

Field Test Plan

- 501 () Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- (✓) Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- (✓) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- (✓) Store a series of records using the SHIFT/^{file}P7 [duplicate field] key to carry forward field values from the previously stored record.
- N/A { () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- (✓) Test LOV where that capability is defined.
- () Select the first entry in the LOV.
- () Select the last entry in the LOV.
- (✓) Use the FIND option and select an entry.
- (✓) Enter data beyond the length of the field.
- (✓) Set the keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- (✓) Attempt to enter data as lower case.
- N/A () Modify each default value with a valid alternative.

Block: CONSTITUENT
Title: Constituents

Field Name: TYPE_CONSTITUENT

Field Test Plan

N/A () Modify each default value with an invalid alternative.

Field Test Case

N/A { () Accept the default.
() Override the default with a valid entry value.
() Override the default with an invalid data value.

N/A () Begin field with a space.

(/ Enter data beyond length of field.
() / If autoskip is in effect, cursor will skip to the next field.
(/ If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

(/ Set keyboard on lower case and enter data. Data should be converted to upper case.

(/ Clear an existing field and attempt to update the record.

Field Test Results Expected

562 () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

() Field help describes:
() Purpose of the field
() LOV reference if available
(/ Valid range of values where present
(/ Required indicator when field is NOT NULL
(/ Default value when appropriate

(/ Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

Block: CONSTITUENT
 Title: Constituents

Field Name: TYPE_CONSTITUENT

Field Test Results Expected

- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- (✓) Store a series of records using the SHIFT/^{F6} [duplicate field] key to carry forward field values from the previously stored record.
- (✓) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - (✓) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- (✓) Use SQL*Plus to verify all data fields are processed correctly.
 - (✓) All data fields are correctly stored in the database.
- N/A { () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- (✓) Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~

Block: CONSTITUENT
Title: Constituents

Field Name: NAME_CON_LONG

Field Description

Field Test Plan

- 563 (✓) Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- (✓) Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- N/A () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- N/A { () Modify each default value with a valid alternative.
() Modify each default value with an invalid alternative.
() Test LOV where that capability is defined.
() Select the first entry in the LOV.
() Select the last entry in the LOV.
() Use the FIND option and select an entry.
- (✓) Enter data beyond the length of the field.
- (✓) Set the keyboard on lower case and enter data. Data should be converted to upper case.
- N/A (✓) ~~Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.~~ Dup
- N/A () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- (✓) Attempt to enter data as lower case.
- (✓) Skip the field using the TAB key.

Block: CONSTITUENT
 Title: Constituents

Field Name: NAME_CON_LONG

Field Test Plan

- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored.

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - (✓) Valid range of values where present
 - () Required indicator when field is NOT NULL

Block: CONSTITUENT
Title: Constituents

Field Name: NAME_CON_LONG

Field Test Results Expected

- () Default value when appropriate
- (/) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (/) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- (/) ~~Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.~~ DWP
- (/) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - (/) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- (/) Use SQL*Plus to verify all data fields are processed correctly.
- (/) All data fields are correctly stored in the database.
- (/) Bypass the field by using the TAB key.
 - (/) No error should occur.
 - (/) The database contains a null value for the field.
- () ~~Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.~~ DWP
- (/) Delete an existing value. The field should be cleared and stored as a null value.
- (/) Any alphanumeric characters may be entered into the field.
- (/) Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

N/A

Block: CONSTITUENT
Title: Constituents

Field Name: NAME_CON_LONG

Field Test Results Expected

- N/A () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- N/A () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- (/ Attempt to enter data as lower case. ~~An error should result.~~

564 Unit measure
test missing

Block: CONSTITUENT
Title: Constituents

Field Name: MOLECULAR_FORMULA

Field Description

Field Test Plan

- 558
- (/) Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
 - (/) Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (/) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
 - (/) Store a series of records using the SHIFT/~~ET~~^{FL} [duplicate field] key to carry forward field values from the previously stored record.
 - () Modify each default value with a valid alternative.
 - () Modify each default value with an invalid alternative.
 - () Test LOV where that capability is defined.
 - () Select the first entry in the LOV.
 - () Select the last entry in the LOV.
 - () Use the FIND option and select an entry.
 - (/) Enter data beyond the length of the field.
 - (/) Set the keyboard on lower case and enter data. Data should be converted to upper case.
 - (/) Skip the field using the TAB key.
 - (/) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
 - (/) Delete an existing value. The field should be cleared and stored.

Block: CONSTITUENT
Title: Constituents

Field Name: MOLECULAR_FORMULA

Field Test Case

- (/) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (/) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (/) Begin field with a space.
- (/) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (/) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (/) Set keyboard on lower case and enter data. Data should be converted to upper case.

Field Test Results Expected

- (/) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - (/) Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - (/) Required indicator when field is NOT NULL
 - (/) Default value when appropriate
- (/) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (/) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or

Block: CONSTITUENT
Title: Constituents

Field Name: MOLECULAR_FORMULA

Field Test Results Expected

COM00103 (if no LOV).

- () ~~Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.~~ Dup
- (/) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - (/) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- (/) Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- (/) Bypass the field by using the TAB key.
 - (/) No error should occur.
 - () The database contains a null value for the field.
- (/) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (/) Delete an existing value. The field should be cleared and stored as a null value.
- (/) Any alphanumeric characters may be entered into the field.

Block: CONSTITUENT
 Title: Constituents

Field Name: FLAG_TEXT

Field Description

This field is a flag that identifies when supporting text is present for a data row. The valid values are either "Y" (yes) or "N" (no). When the user desires to add text in support of a data row, the flag is set to "Y". The system then brings up the general text form and permits the user to enter text of varying types for the data row. Where text is already present, the user places the cursor on the flag field and selects to retrieve the text data. Text data may be updated. Note that updating text data causes the same hierarchical cloning that takes place elsewhere when a data row is updated. This ensures that the text data is also properly audit trail covered.

Field Test Plan

- () Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- () Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Test LOV where that capability is defined.
 - () Select the first entry in the LOV.
 - () Select the last entry in the LOV.
 - () Use the FIND option and select an entry.
- () Enter data beyond the length of the field.
- () Set the keyboard on lower case and enter data. Data should be

Block: CONSTITUENT
Title: Constituents

Field Name: FLAG_TEXT

Field Test Plan

converted to upper case.

- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- () Attempt to enter data as lower case.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

Block: CONSTITUENT
Title: Constituents

Field Name: FLAG_TEXT

Field Test Results Expected

- () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

Block: CONSTITUENT
Title: Constituents

Field Name: FLAG_TEXT

Field Test Results Expected

-
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
 - () Attempt to enter data as lower case. An error should result.

Block: CONSTITUENT_NEST
Title: Associated Constituents

Block Description

The table links two constituents. This is normally a compound broken into its individual components.

The nest may identify subordinate component constituents, synonyms, and alternates.

Block Test Case

Block Test Results Expected

Field Name: RELATIONSHIP

Field Description

The relationship identifies the nature of the linkage between two constituents. The default value is SUB (subordinate). Other valid values are ALT (alternate) and SYN (synonym). a LOV supported by TYPE_RELATIONSHIP_CODE.

Field Test Plan

- 5/65 () Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- ✓ Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- n/A () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- n/A () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- n/A { () Modify each default value with a valid alternative.
- n/A { () Modify each default value with an invalid alternative.

Block: CONSTITUENT NEST
Title: Associated Constituents

Field Name: RELATIONSHIP

Field Test Plan

- (/) Test LOV where that capability is defined.
 - (/) Select the first entry in the LOV.
 - (/) Select the last entry in the LOV.
 - () Use the FIND option and select an entry.
- ~~///~~ (/) Enter data beyond the length of the field.
- (/) Set the keyboard on lower case and enter data. Data should be converted to upper case.
- (/) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- (/) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- (/) Attempt to enter data as lower case.
- (/) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- (/) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- (/) Modify each default value with a valid alternative.
- (/) Modify each default value with an invalid alternative.

Field Test Case

- (/) Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- (/) Begin field with a space.
- (/) Enter data beyond length of field.
 - (/) If autoskip is in effect, cursor will skip to the next field.
 - (/) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the

Block: CONSTITUENT_NEST
 Title: Associated Constituents

Field Name: RELATIONSHIP

Field Test Case

last position of the field.

- (X) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (X) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (X) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- n/a

(X)

Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- n/a

()

Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- (X) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- (X) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the

Block: CONSTITUENT_NEST
Title: Associated Constituents

Field Name: RELATIONSHIP

Field Test Results Expected

record from the database.

(✓) Use SQL*Plus to verify all data fields are processed correctly.

(✓) All data fields are correctly stored in the database.

(✓) Modify each default value with a valid alternative.

(✓) Modify each default value with an invalid alternative.

01/8 () Any alphanumeric characters may be entered into the field.

() Delete an existing value.

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

(✓) Attempt to enter data as lower case. ~~An error should result.~~

(✓) Delete an existing value.

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Transaction: LI5051U1 Constituent Maintenance

```
=====
Comment ID : 560 Priority: LOW Date: 22-NOV-92
=====
Status      : ASSIGNED Date: 24-NOV-92 Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : TEST USER
System ID   : 10000000677 Oracle ROWID: 00000003.0001.0018
Last Error  : -0
Last Msg    : FRM-40400 Transaction complete -- 1 records posted an
Table       : CONSTITUENT
Block       : CONST
Field       : CONST.IDENT_CONSTITUENT
```

DESCRIPTION

UNIQUE FIELD ALLOWS CHANGES TO BE MADE AND UPDATED

UPDATES

O.K.

NOT PROBLEM

11/24/92

✓

Transaction: LI5051U1 Constituent Maintenance

```
=====
Comment ID : 561   Priority: LOW           Date: 22-NOV-92 .
=====
Status      : ASSIGNED   Date: 24-NOV-92   Class:
Type        : PROBLEM
Assigned To : LARRY TOWNER
Entered By  : TEST USER
System ID   : 10000000659      Oracle ROWID: 00000003.0000.0018
Last Error  : -0
Last Msg    : FRM-40100 At first record.
Table       : CONSTITUENT
Block       : CONST
Field       : CONST.TYPE_CONSTITUENT
=====
```

DESCRIPTION

```
=====
help is duplicated. Narrative does not describe function/purpose
=====
```

2002

Programmer's Bug Report

Transaction: LI5051U1 Constituent Maintenance

=====

Comment ID : 562 Priority: LOW Date: 22-NOV-92

=====

Status : ASSIGNED Date: 24-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By : TEST USER

System ID : 10000000659 Oracle ROWID: 00000003.0000.0018

Last Error : -0

Last Msg : FRM-40204 Cursor is at beginning of field value.

Table : CONSTITUENT

Block : CONST

Field : CONST.TYPE_CONSTITUENT

DESCRIPTION

no LOV mentioned in help

Transaction: LI5051U1 Constituent Maintenance

=====

Comment ID : 563 Priority: LOW Date: 22-NOV-92

=====

Status : ASSIGNED Date: 24-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By : TEST USER

System ID : 10000000678 Oracle ROWID: 00000003.0002.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : CONSTITUENT

Block : CONST

Field : CONST.NAME_CON_LONG

DESCRIPTION

=====

help states column is optional. Hint text does not. Help is
duplicated. Help also states no blank allowed.

✓

Transaction: LI5051U1 Constituent Maintenance

=====

Comment ID : 565 Priority: LOW Date: 22-NOV-92

=====

Status : ASSIGNED Date: 24-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By : TEST USER

System ID : 10000000678 Oracle ROWID:

Last Error : -0

Last Msg : FRM-40100 At first record.

Table : CONSTITUENT_NEST

Block : CONNST

Field : CONNST_RELATIONSHIP

DESCRIPTION

narrative needs help

1st two sentences make sense
✓ then the last 3 sentences are
kinda of mixed up.

Tag

3.8 LI5052U1 LABORATORY MAINTENANCE

LEMIS Module Test Checklist Report

Name: LABORATORY MAINTENANCE
 Short Name: LI5052U1
 Purpose: Maintains laboratory information

Initial Test Date: 11/30/92 JK 12/1/92 JK

Verification Date: 12-1-92 JA O.R.

Help	Disp Fkeys	Security			
		Ins	Del	Query	Upd
()	()	()	()	()	()

Block: LABORATORY
 Title: Laboratory Definition

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry	
()	()	()	()	()	()	()	()	
Data Element		Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT LAB		CHAR	6	()	()	()	()	()
NAME LAB		CHAR	40	()	()	()	()	()

Block: LABORATORY

Title: Laboratory Definition

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_LAB	CHAR	6	()	()	()	()	()
NAME_LAB	CHAR	40	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: LABORATORY_NEST

Title: Associated Laboratories

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
RELATIONSHIP	CHAR	3	()	()	()	()	()

Block: ORGANIZATION

Title: Organization

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_ORGAN	CHAR	12	()	()	()	()	()
TITLE	CHAR	40	()	()	()	()	()

Module Description

The module provides ability to add or modify but not delete a laboratory.

The laboratory will be linked to an organization. The lab address data is kept with the organization.

The ability to identify the analytical methods performed by the laboratory is included.

The relationship between method and laboratory defaults to REL. Where a laboratory is preferred as the performer of a specific method, the relationship code is DEF (default).

Allow for ten methods to be displayed if possible.

The validation level for laboratories is 9. The ident_owner field

Module Description

defaults to 'library'.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

Test status conditions are:

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
 - Developer Date Complete 11/29/92
- () 3 - Initial unit testing complete.
 - Developer Date Complete 11/29/92

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Module Test Plan

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
Developer _____ Date Complete _____
- () 8 - Secondary testing complete, no problems found.
Tester _____ Date Complete _____

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Test status conditions are:

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.
Tester _____ Date Complete _____

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Module Test Plan

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

() Reviews tests results.

() Reviews problem reports to ensure all corrections done.

() Updates NOTES text to reflect testing results.

Project Manager [Signature] Date Complete 12/1/92

() 20 - The transaction is placed into the production environment.

() Software files in production directories.

() DMR 550 updated to add transaction.

() DMR 900 updated to add transaction.

() Transaction added to production usage menu.

System Architect [Signature] Date Complete 12/1/92

Module Test Case

(✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

(✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]

(✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

(✓) Transaction help describes:

(✓) Purpose of the transaction

(✓) Functions performed by the transaction

(N) Where multiple page, what happens on each page

(✓) Add a new record by using the insert key.

(✓) The insert key clears all blocks of the entry screen.

(✓) A new record can be successfully added and committed.

(✓) Add a new record by scrolling past the last entry using the down arrow key.

(✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.

(✓) All fields of the transaction blank when the end of the string of retrieved records is reached.

(✓) A new data record can be successfully added.

(✓) Query a single record.

(✓) A single record is retrieved.

(✓) Pressing SHIFT/F12 clears the record and permits entering a second query.

(✓) The second query successfully retrieves the desired data.

(✓) Query multiple records using wild cards.

(✓) Pressing F12 clears the screen and makes the transaction ready for a query.

(✓) A query composed of a trailing % wildcard retrieves the desired data. ~~Verify by running the same query through SQL*Plus.~~

(✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.

(✓) A query using both prefix and suffix % wildcards retrieves the correct data. ~~Verify by running the same query through SQL*Plus.~~

(✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~

ORIG ID protected against update

(✓) Change each field individually, committing after each change.

(✓) Change each field, as per the field test plan, for both valid and invalid values.

(✓) Delete an existing record. *unable to delete as 'testuser'*

*

() The selected record is deleted from the database. Verify ~~by using SQL*Plus to attempt to retrieve the record.~~

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

N/A
() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.

() All data fields are correctly stored in the database.

Block: LABORATORY

Title: Laboratory Definition

Block Description

A laboratory is a library table. The table contains those labs that provide analytical services to the site.

Address information for the laboratory is carried through the functional unit data. A functional unit/organization is attached to every laboratory.

Block Test Case laboratories is kept at validation level 9 (fully ~~ta~~ is available to all users of the system.

Block Test Results Expected

Field Name: IDENT LAB

Field Description

Corresponds to Lab Code in HEIS. Unique identifier for the laboratory.

Field Test Plan

- (✓) Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- (✓) Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
- (✓) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- NA Store a series of records using the SHIFT/ENTER [duplicate field] key to carry forward field values from the previously stored record.
- NA Modify each default value with a valid alternative.
- NA Modify each default value with an invalid alternative.
- () Test LOV where that capability is defined.
- NA () Select the first entry in the LOV.
- () Select the last entry in the LOV.
- () Use the FIND option and select an entry.
- (✓) Enter data beyond the length of the field.

Field Name: IDENT_LAB

Field Test Plan

- (✓) Set the keyboard on lower case and enter data. Data should be converted to upper case.
- [() Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.] Dup Test
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- (✓) Attempt to enter data as lower case.

- [() Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.] Dup Test
- [() Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.] "

Field Test Case

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - ~~(NA)~~ If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - ~~(NA)~~ LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL

Field Name: IDENT_LAB

Field Test Results Expected

~~NA~~ Default value when appropriate

(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using ~~SQL*Plus before and after the changes were made.~~

(✓) Change the field, committing after each change.

(✓) Change the field where a range of valid values exists:

() Value below valid range should display message COM00101 or COM00103 (if no LOV).

~~NA~~ () Value above valid range should display message COM00101 or COM00103 (if no LOV).

() Store a series of records using the SHIFT/F7^b [duplicate field] key to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

~~NA~~ () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

(✓) Any alphanumeric characters may be entered into the field.

(✓) Delete an existing value.

(✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

(✓) Attempt to enter data as lower case. ~~An error should result.~~

(✓) Delete an existing value.

converts to upper case

does not apply

Field Name: IDENT_LAB

Field Test Results Expected

- (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted]. is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Field Name: NAME LAB

Field Description

The currently used name of the laboratory.

Field Test Plan

- 6456 Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- (✓) Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
 - (✓) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
 - (✓) Store a series of records using the SHIFT/ENTER [duplicate field] key to carry forward field values from the previously stored record.
 - NA Modify each default value with a valid alternative.
 - NA Modify each default value with an invalid alternative.
 - NA Test LOV where that capability is defined.
 - () Select the first entry in the LOV.
 - () Select the last entry in the LOV.
 - () Use the FIND option and select an entry.
 - (✓) Enter data beyond the length of the field.
 - (✓) Set the keyboard on lower case and enter data. Data should be converted to upper case.
 - (✓) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
 - (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
 - (✓) Attempt to enter data as lower case.
 - (✓) Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
 - (✓) Enter data into the field, backspace the data out of the field and

Same

Same

Field Name: NAME_LAB

Field Test Plan

attempt to TAB to the next field. An error should occur.

Field Test Case

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - (✗) If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✗) LOV reference if available
 - (✗) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✗) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key
 ✗ to carry forward field values from the previously stored record.
 DHP
Est

Field Name: NAME_LAB

Field Test Results Expected

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

(✓) Any alphanumeric characters may be entered into the field.

(✓) Delete an existing value.

(✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

(✓) Attempt to enter data as lower case. ~~An error should result.~~ Converts to upper case

() Delete an existing value.

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. duplicate Test

() Attempt to bypass the field. An error message should tell you that the field data must be entered. 11

() Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Field Name: FLAG_TEXT

Field Description

This field is a flag that identifies when supporting text is present for a data row. The valid values are either "Y" (yes) or "N" (no). When the user desires to add text in support of a data row, the flag is set to "Y". The system then brings up the general text form and permits the user to enter text of varying types for the data row. Where text is already present, the user places the cursor on the flag field and selects to retrieve the text data. Text data may be updated. Note that updating text data causes the same hierarchical cloning that takes place elsewhere when a data row is updated. This ensures that the text data is also properly audit trail covered.

Field Test Plan

- () Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- () Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Test LOV where that capability is defined.
 - () Select the first entry in the LOV.
 - () Select the last entry in the LOV.
 - () Use the FIND option and select an entry.
- () Enter data beyond the length of the field.
- () Set the keyboard on lower case and enter data. Data should be converted to upper case.
- () Delete an existing data field value. Determine that the system

Field Name: FLAG_TEXT

Field Test Plan

will display an error if the data field is NOT NULL.

- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- () Attempt to enter data as lower case.
- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

Field Name: FLAG_TEXT

Field Test Results Expected

- () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

Field Name: FLAG_TEXT

Field Test Results Expected

- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: LABORATORY_NEST
 Title: Associated Laboratories

Block Description

This table provides recursive association between different laboratories. The association is most often used when one lab is a subsidiary of another. It may also be used to identify labs that are used as subcontractors to another.

Block Test Case

Block Test Results Expected

Field Name: RELATIONSHIP

Field Description

The field identifies the association between two labs. Valid codes:

SUB - subsidiary
 ALT - alternate
 CON - subcontractor

Field Test Plan

- () Display column help. Verify that the narrative adequately describes the purpose and functions of the column in end-user terms.
- () Change the data field. Verify that the change was made by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Test LOV where that capability is defined.
 - () Select the first entry in the LOV.

646 Cannot access Field
 to Test

Block: LABORATORY_NEST
Title: Associated Laboratories

Field Name: RELATIONSHIP

Field Test Plan

- () Select the last entry in the LOV.
- () Use the FIND option and select an entry.
- () Enter data beyond the length of the field.
- () Set the keyboard on lower case and enter data. Data should be converted to upper case.
- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- () Attempt to enter data as lower case.
- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

Block: LABORATORY_NEST
Title: Associated Laboratories

Field Name: RELATIONSHIP

Field Test Case

- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

Block: LABORATORY_NEST
Title: Associated Laboratories

Field Name: RELATIONSHIP

Field Test Results Expected

- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value:
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Transaction: LI5052U1 Laboratory Information Maintenance

=====

Comment ID : 513 Priority: MED Date: 18-NOV-92

=====

Status : ASSIGNED Date: 18-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By: LARRY TOWNER

System ID : 10000000682 Oracle ROWID: 0000000D.0004.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : LABORATORY

Block : LAB

Field : LAB.IDENT_LAB

DESCRIPTION

Text entry is rejected for this transaction.

FIXED
11/29/92
✓

Programmer's Bug Report

Transaction: LI5052U1 Laboratory Information Maintenance

=====

Comment ID : 646 Priority: LOW Date: 30-NOV-92

=====

Status : ASSIGNED Date: 30-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000001248 Oracle ROWID:

Last Error : -0

Last Msg : FRM-40202 Field must be entered.

Table : LABORATORY_NEST

Block : LABNST

Field : LABNST.DSP_IDENT_LAB

DESCRIPTION

User cannot move the cursor to the 'relationship' field to perform
updating/testing.*NOT PROD**11/30/92*

3.9 LI5053Q1 METHOD PICK LIST

Testing for this module was performed during the testing of LI5056U1 ASSIGN METHODS TO LABORATORY.

3.10 LI5053U1 METHOD MAINTENANCE

LEMIS Module Test Checklist Report

Laboratory

Name: METHOD MAINTENANCE

Short Name: LI5053U1

Purpose: Update the library of laboratory procedures

Initial Test Date: 11/24/92 JC 12/1/92 SK

Verification Date: 12/1/92 JC

	Disp	Security			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: METHOD

Title: Laboratory Methods

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	30	()	()	()	()	()
TITLE_METHOD	CHAR	34	()	()	()	()	()

Block: METHOD
Title: Methods

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	8	()	()	()	()	()
TITLE_METHOD	CHAR	34	()	()	()	()	()
TYPE_METHOD	CHAR	2	()	()	()	()	()
EXTRACTION_TIME	NUMBER	5	()	()	()	()	()
HOLDING_TIME	CHAR	6	()	()	()	()	()
PRESERVATIVE	CHAR	9	()	()	()	()	()
QUAN_BOTTLE	NUMBER	2	()	()	()	()	()
TYPE_BOTTLE_COMPOSITIO	CHAR	2	()	()	()	()	()
SIZE_BOTTLE	NUMBER	5	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: METHOD_NEST
Title: Associated Methods

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
RELATIONSHIP	CHAR	3	()	()	()	()	()

Block: TYPE_BOTTLE_CODE
Title: Type of Containers

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Module Description

The module maintains laboratory methods. Methods may be added, modified or deleted, but may not be modified or deleted if they are existing in live records on the database.

Method is a hierarchical structure. Through use of the method_nest table, methods may be associated with each other. There are three levels of method identified. Each serves a specific purpose. They are:

Module Description

Special Analysis Classes Analysis Classes Methods

The special analysis classes are ones that add or subtract constituents from analysis classes. For example: ICP metals plus Sn. In this example, the special analysis class would link to the analysis class ICP metals with a relationship of REL and to the constituent Sn with a relationship of PLS.

Analysis classes are groupings of methods that are used by labs to detect that group of constituents. Analysis classes do not directly link to constituents.

Methods are the specific laboratory test procedures that are used to detect one or more specific constituents. The constituent associated with a method are linked to that method.

This module supports maintenance of the method alone. It identifies the constituents related to the method, the default bottle type, holding time, extraction time, bottle composition, size and quantity and preservative to be used with the method.

Text may be associated with the method.

A method may be associated with one or more other methods. The most common associations are ALT (alternate) or RPL (replacement/substitute). Provide the capability to display two other method associations. The default is ALT. Display the method identifier, title, and relationship code.

A method is associated with a default container type and size. Where substitutes are warranted, this is done by laboratory preference (found in the lab-method table) or by field substitution.

A LOV is provided to the type_bottle_code table.

The operator enters the bottle composition and size. This is verified with the code table and an error occurs if:

1. The bottle composition is incorrect.
2. The bottle size for the composition is incorrect.

A method is associated with a default preservative. A LOV is provided to the type_preservative_code table.

The data validation level of each method defaults to 9. Ident_owner defaults to 'library'.

The method type code defaults to ME.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

Test status conditions are:

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Developer	<u> </u>	Date Complete	<u>11/25/92</u>
Developer	<u> </u>	Date Complete	<u>11/25/92</u>

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.

Module Test Plan

Tester _____ Date Complete _____
 () 6 - Problem resolution in progress.
 () 7 - Problem resolution complete.
 Developer _____ Date Complete _____
 () 8 - Secondary testing complete, no problems found.
 Tester PC Date Complete 11/26/92

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Test status conditions are:

() 9 - System testing in progress.
 () 10 - System testing complete, problems found.
 Tester _____ Date Complete _____
 () 11 - Problem resolution in progress.
 () 12 - Problem resolution complete.
 Developer _____ Date Complete _____
 () 13 - System testing complete, no problems found.
 Tester _____ Date Complete _____

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and

Module Test Plan

determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.

Project Manager LS Date Complete 12/1/92

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.

System Architect AA Date Complete 12/1/92

Module Test Case

✓✓ Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

- (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]

566/ ✓ Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

- (✓) Transaction help describes:

- ✓ Purpose of the transaction
- ✓ Functions performed by the transaction
- ✓ Where multiple page, what happens on each page

584/ ✓ Add a new record by using the insert key.

- ✓ The insert key clears all blocks of the entry screen.
- ✓ A new record can be successfully added and committed.

✓ Add a new record by scrolling past the last entry using the down arrow key.

- (✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
- (✓) All fields of the transaction blank when the end of the string of retrieved records is reached.

✓ A new data record can be successfully added.

✓ Query a single record.

- ✓ A single record is retrieved.

✓ Pressing SHIFT/F12 clears the record and permits entering a

✓(✓) Second query.

✓(✓) The second query successfully retrieves the desired data.

✓(✓) Query multiple records using wild cards.

✓(✓) Pressing F12 clears the screen and makes the transaction ready for a query.

✓(✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.

✓(✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.

✓(✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.

585 ✓(✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

*Commit
but doesn't
commit*

672 () Change each field individually, committing after each change.

() Change each field, as per the field test plan, for both valid and invalid values. .

✓(✓) Delete an existing record.

✓ (✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.

() All data fields are correctly stored in the database.

Block: METHOD
Title: Methods

Block Description

A laboratory procedure is equivalent to an analytical method. It is

Block: METHOD
Title: Methods

Block Description

the analysis process used to determine the presence of one or more constituents. The table contains those procedures that have been identified to support detection of constituents.

Laboratory procedures may be associated with one or more constituents. They may be also associated with the laboratories that are qualified to perform the procedures.

The data for all lab procedures is kept at validation level 9 (fully validated). Data is available to all users of the system.

Some samples may share the same method. Lab names are not needed to define methods because they are set by procedures specified by the customer. Methods might include:

TOC (total organic carbon)
ICP (inductively coupled plasma)
IC (ion chromatography)
PH

Each method may have up to 150 constituents, or analytes.

Block Test Case

Block Test Results Expected

Field Name: IDENT_METHOD

Field Description

This field is the user visible unique identifier for the lab procedure.

Field Test Case

- ✓(✓) Begin field with a space.
- ✓(✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - ✓(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

Block: METHOD
Title: Methods

Field Name: IDENT_METHOD

Field Test Case

- ✓(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- ✓(✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- ✓(✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

✓(✓) Field help describes:

- ✓(✓) Purpose of the field
- () LOV reference if available
- ✓(✓) Valid range of values where present
- ✓(✓) Required indicator when field is NOT NULL
- () Default value when appropriate

Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

- ✓(✓) Store a series of records using the SHIFT/^{FL}E7 [duplicate field] key to carry forward field values from the previously stored record.

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

*no dups
allowed for
key field*

Block: METHOD
Title: Methods

Field Name: IDENT_METHOD

Field Test Results Expected

- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- ✓(✓) Any alphanumeric characters may be entered into the field.
- ✓(✓) Delete an existing value.
- ✓ (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- ✓(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- ✓(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- ✓(✓) Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: METHOD
Title: Methods

Field Name: TITLE_METHOD

Field Description

Enter a 1-40 character descriptive title for the procedure.

Field Test Case

- ✓(✓) Begin field with a space.
- ✓(✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - ✓(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ✓(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- ✓(✓) Clear an existing field and attempt to update the record.

Req'd FIELD

Field Test Results Expected

- ✓(✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - ✓() Field help describes:
 - ✓(✓) Purpose of the field
 - ✓(✓) LOV reference if available
 - ✓(✓) Valid range of values where present
 - ✓(✓) Required indicator when field is NOT NULL
 - () Default value when appropriate
- ✓(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- ✓(✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

Block: METHOD
Title: Methods

Field Name: TITLE_METHOD

Field Test Results Expected

- FL*
- ✓(✓) Store a series of records using the SHIFT/FT [duplicate field] key to carry forward field values from the previously stored record.
 - () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - NA* () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
 - ✓(✓) Any alphanumeric characters may be entered into the field.
 - ✓(✓) Delete an existing value.
 - ✓(✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - ✓(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - ✓(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
 - ✓(✓) Attempt to enter data as lower case. ~~An error should result.~~
 - () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- 34P*

Block: METHOD
Title: Methods

Field Name: TYPE_METHOD

Field Description

Field Test Case

see prob. # 672

- ☒ () Accept the default.
- ☒ () Override the default with a valid entry value.
- ☒ () Override the default with an invalid data value.
- ☒ () Begin field with a space.
- ☒ () Enter data beyond length of field.
- ☒ () If autoskip is in effect, cursor will skip to the next field.
- ☒ () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ☒ () Set keyboard on lower case and enter data. Data should be converted to upper case.
- ☒ () Clear an existing field and attempt to update the record.

Field Test Results Expected

- ☒ () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

() Field help describes:

- ☒ () Purpose of the field
- ☒ () LOV reference if available
- ☒ () Valid range of values where present
- ☒ () Required indicator when field is NOT NULL
- ☒ () Default value when appropriate

- ☒ () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- ☒ () Change the field, committing after each change.

Block: METHOD
Title: Methods

Field Name: TYPE_METHOD

Field Test Results Expected

- ✓(✓) Change the field where a range of valid values exists:
() Value below valid range should display message COM00101 or COM00103 (if no LOV).
- ✓(✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- N/A () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- ✓(✓) Modify each default value with a valid alternative.
- ✓(✓) Modify each default value with an invalid alternative.
- N/A () Any alphanumeric characters may be entered into the field. NO
- ✓(✓) Delete an existing value.
- ✓(✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- ✓(✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- ✓(✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- ✓(✓) Attempt to enter data as lower case. An error should result.

Block: METHOD
Title: Methods

Field Name: EXTRACTION_TIME

Field Description

Field Test Case

- ✓(✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- ✓(✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- ✓(✓) Attempt to enter alpha data.
- ✓(✓) Attempt to enter longer value than allowed.
- ✓(✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- ✓(✓) Attempt to enter negative value. No error should occur.

Field Test Results Expected

- ✓(✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - ✓() Field help describes:
 - ✓(✓) Purpose of the field
 - () LOV reference if available
 - ✓(✓) Valid range of values where present
 - ✓(✓) Required indicator when field is NOT NULL
 - () Default value when appropriate
- ✓(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- ✓(✓) Change the field, committing after each change.

Block: METHOD
Title: Methods

Field Name: EXTRACTION_TIME

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).

✓(✓) Store a series of records using the SHIFT/^{Fl}F7 [duplicate field] key to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

✓(✓) Bypass the field by using the TAB key.

() No error should occur.

() The database contains a null value for the field.

✓(✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

✓(✓) Delete an existing value. The field should be cleared and stored as a null value.

590 (✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.

✓(✓) Begin field entry with a space. ~~An error should occur.~~

Block: METHOD
Title: Methods

Field Name: HOLDING TIME

Field Description

Field Test Case

- ✓(✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- ✓(✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- ✓(✓) Begin field with a space.
- ✓(✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - ✓(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ✓(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

Field Test Results Expected

- ✓(✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - ✓(✓) Field help describes:
 - ✓(✓) Purpose of the field
 - () LOV reference if available
 - ✓(✓) Valid range of values where present
 - ✓(✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- ✓(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

Block: METHOD
Title: Methods

Field Name: HOLDING_TIME

Field Test Results Expected

- ✓ Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- ✓ Store a series of records using the SHIFT/F7 ^{fl6} [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- WA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- ✓ Bypass the field by using the TAB key.
 - () No error should occur.
 - () The database contains a null value for the field.
- ✓ Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- ✓ Delete an existing value. The field should be cleared and stored as a null value.
- ✓ Any alphanumeric characters may be entered into the field.
- ✓ Attempt to enter data as lower case. ~~An error should result.~~

Block: METHOD
Title: Methods

Field Name: PRESERVATIVE

Field Description

Field Test Case

- ✓ (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- ✓ (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- ✓ A/A () Begin field with a space.
- ✓ (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - ✓ (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ✓ (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

Field Test Results Expected

- ✓ (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - ✓ (✓) Field help describes:
 - (✓) Purpose of the field
 - () LOV reference if available — No LOV in help text
 - (✓) Valid range of values where present
 - ✓ (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- ✓ (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

Block: METHOD
Title: Methods

Field Name: PRESERVATIVE

Field Test Results Expected

- ✓(✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- ✓(✓) Store a series of records using the SHIFT/^{to}F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- N/A () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- ✓(✓) () All data fields are correctly stored in the database.
- ✓(✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- ✓(✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- ✓(✓) Delete an existing value. The field should be cleared and stored as a null value.
- N/A () Any alphanumeric characters may be entered into the field. *must match LOV*
- ✓(✓) Attempt to enter data as lower case. ~~An error should result.~~

Block: METHOD
Title: Methods

Field Name: QUAN_BOTTLE

Field Description

Field Test Case

- ✓(✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- ✓(✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- ✓(✓) Attempt to enter alpha data.
- ✓(✓) Attempt to enter longer value than allowed.
- ✓(✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- ✓(✓) Attempt to enter negative value. No error should occur.

REQ'D
FIELD

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

() Field help describes:

- () Purpose of the field
- () LOV reference if available
- (✓) Valid range of values where present
- (✓) Required indicator when field is NOT NULL
- () Default value when appropriate

- ✓(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- ✓(✓) Change the field, committing after each change.

Block: METHOD
Title: Methods

Field Name: QUAN_BOTTLE

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).

✓(✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

(✓) Bypass the field by using the TAB key.

() No error should occur.

() The database contains a null value for the field.

✓(✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

✓(✓) Delete an existing value. The field should be cleared and stored as a null value.

✓(✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.

✓(✓) Begin field entry with a space. An error should occur.

Space disallows

wrong test

pgm won't allow blank

Block: METHOD
Title: Methods

Field Name: TYPE_BOTTLE_COMPOSITION

Field Description

Field Test Case

- ✓(✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- ✓(✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- NA ✓(✓) Begin field with a space.
- ✓(✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - ✓(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ✓(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

REQ'D
FIELD

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field - Needs
 - () LOV reference if available - Needs
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL - wrong
 - () Default value when appropriate
- ✓(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

Block: METHOD
Title: Methods

Field Name: TYPE_BOTTLE_COMPOSITION

Field Test Results Expected

- ☒ Change the field, committing after each change.
- ☐ Change the field where a range of valid values exists:
 - ☐ Value below valid range should display message COM00101 or COM00103 (if no LOV).
- 592 ☒ Store a series of records using the SHIFT/END [duplicate field] key to carry forward field values from the previously stored record.
- ☐ Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- ☐ The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- ☐ Use SQL*Plus to verify all data fields are processed correctly.
- ☒ Bypass the field by using the TAB key.
 - ☐ No error should occur.
 - ☐ The database contains a null value for the field.
- ☒ Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- ☒ Delete an existing value. The field should be cleared and stored as a null value.
- ☒ Any alphanumeric characters may be entered into the field.
- ☒ Attempt to enter data as lower case. An error should result.

NO, IT'S REQ'D
no blanks allowed
must be from LOV

Block: METHOD
Title: Methods

Field Name: SIZE_BOTTLE

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- ✓(✓) Attempt to enter alpha data.
- ✓(✓) Attempt to enter longer value than allowed.
- (✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- (✓) Attempt to enter negative value. No error should occur.

FIELD is
REQ'D

must be
LOV option

Field Test Results Expected

- ✓(✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- ✓(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- ✓(✓) Change the field, committing after each change.

Block: METHOD
Title: Methods

Field Name: SIZE_BOTTLE

Field Test Results Expected

- () Change the field where a range of valid values exists:
☒ Value below valid range should display message COM00101 or COM00103 (if no LOV).

☒ Store a series of records using the SHIFT/^{FB} [duplicate field] key to carry forward field values from the previously stored record.

☒ Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

☒ The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

☒ Use SQL*Plus to verify all data fields are processed correctly.

☒ Bypass the field by using the TAB key.

() No error should occur.

() The database contains a null value for the field.

☒ Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

☒ Delete an existing value. The field should be cleared and stored as a null value.

☒ Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.

☒ Begin field entry with a space. ~~An error should occur.~~

REQ'D
Field -
Wrong Test
value restricted
to
LOV options

Block: METHOD
Title: Methods

Field Name: FLAG_TEXT

not testable

Field Description

This field is a flag that identifies when supporting text is present for a data row. The valid values are either "Y" (yes) or "N" (no). When the user desires to add text in support of a data row, the flag is set to "Y". The system then brings up the general text form and permits the user to enter text of varying types for the data row. Where text is already present, the user places the cursor on the flag field and selects to retrieve the text data. Text data may be updated. Note that updating text data causes the same hierarchical cloning that takes place elsewhere when a data row is updated. This ensures that the text data is also properly audit trail covered.

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- () Field help describes:

Block: METHOD
Title: Methods

Field Name: FLAG_TEXT

Field Test Results Expected

- () Purpose of the field
- () LOV reference if available
- () Valid range of values where present
- () Required indicator when field is NOT NULL
- () Default value when appropriate

- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.

- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not

Block: METHOD
Title: Methods

Field Name: FLAG_TEXT

Field Test Results Expected

-
- be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
 - () Attempt to enter data as lower case. An error should result.
 - () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: METHOD_NEST
 Title: Associated Methods

Block Description

Block Test Case

Block Test Results Expected

Field Name: RELATIONSHIP

Field Description

The relationship identifies the nature of the linkage between two occurrences of the table parents. A default value is typically assigned. Other values are found in a LOV supported by TYPE_RELATIONSHIP_CODE.

Field Test Case

- (✓) Accept the default.
- (✓) Override the default with a valid entry value.
- n/a () Override the default with an invalid data value.
- n/a () Begin field with a space.
- () Enter data beyond length of field.
- n/a () If autoskip is in effect, cursor will skip to the next field.
- n/a () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- n/a () Set keyboard on lower case and enter data. Data should be converted to upper case.
- n/a () Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

Block: METHOD_NEST
Title: Associated Methods

Field Name: RELATIONSHIP

Field Test Results Expected

- (☒) Field help describes:
 - (☒) Purpose of the field
 - (☒) LOV reference if available
 - (☐) Valid range of values where present
 - (☐) Required indicator when field is NOT NULL
 - (☒) Default value when appropriate

N/A () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

N/A () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

() The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

() All data fields are correctly stored in the database.

N/A () Modify each default value with a valid alternative.

N/A () Modify each default value with an invalid alternative.

N/A () Any alphanumeric characters may be entered into the field.

N/A () Delete an existing value.

Block: METHOD_NEST
Title: Associated Methods

Field Name: RELATIONSHIP

Field Test Results Expected

- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Programmer's Bug Report

Application: LIBR Library functions

Transaction: LI5053U1 Laboratory Method Maintenance

=====

Comment ID : 584 Priority: LOW Date: 24-NOV-92

=====

Status : ASSIGNED Date: 26-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40350 Query caused no records to be retrieved.

Table : METHOD_NEST

Block : METNST

Field : METNST.DSP_IDENT_METHOD

DESCRIPTION

tab from the last field in block 1 moved me to first field in block 2.
I should have had to page down.

CORRECTIVE ACTION

11/26/92 - Ken - The FLDNAV preference s/b N

11/30/92

FIXED.

✓

Transaction: LI5053U1 Laboratory Method Maintenance

=====
Comment ID : 585 Priority: LOW Date: 24-NOV-92
=====

Status : ASSIGNED Date: 26-NOV-92 Class:
Type : PROBLEM
Assigned To: TODD ADAMS
Entered By : TEST USER
System ID : 10000000658 Oracle ROWID: 00000030.0000.0018
Last Error : -0
Last Msg : FRM-40400 Transaction complete -- 1 records posted an
Table : METHOD
Block : METHOD
Field : METHOD.IDENT_METHOD

DESCRIPTION

ID field should be unique but I was able to change data and commit*NOT
PROB**11/30/92.
✓*

Transaction: LI5053U1 Laboratory Method Maintenance

=====

Comment ID : 590 Priority: LOW Date: 24-NOV-92

=====

Status : DOCCHNG Date: 26-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By : TEST USER

System ID : 10000000955 Oracle ROWID: 00000033.0010.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : METHOD

Block : METHOD

Field : METHOD.EXTRACTION_TIME

DESCRIPTION

HELP STATES COMMA CAN BE USED. CANNOT GET COMMA TO BE ACCEPTED

11/29

Transaction: LI5053U1 Laboratory Method Maintenance

=====

Comment ID : 591 Priority: LOW Date: 24-NOV-92

=====

Status : DOCCHNG Date: 26-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By : TEST USER

System ID : 10000000955 Oracle ROWID: 00000033.0010.0018

Last Error : -0

Last Msg : FRM-40204 Cursor is at beginning of field value.

Table : METHOD

Block : METHOD

Field : METHOD.QUAN BOTTLE

DESCRIPTION

bottle, comp and size need descriptive help added

11/27

Programmer's Bug Report

Transaction: LI5053U1 Laboratory Method Maintenance

```
=====
Comment ID : 592   Priority: LOW           Date: 24-NOV-92
=====
Status      : ASSIGNED   Date: 26-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : TEST USER
System ID   : 10000000953      Oracle ROWID: 00000033.000E.0018
Last Error  : -0
Last Msg    : FRM-40350  Query caused no records to be retrieved.
Table       : METHOD
Block       : METHOD
Field       : METHOD.TYPE_BOTTLE_COMPOS
              ITION
=====
```

DESCRIPTION

trying to commit dup field. comp and size are connected in some way.
Shift F6 allows comp to copy but not size thus it commits then
disappears

~~not fixed~~ *FIXED* 11/30
MSF *FL* *OA*
SIZE *ASD*
✓

Transaction: LI5053U1 Laboratory Method Maintenance

=====

Comment ID : 593 Priority: LOW Date: 24-NOV-92

=====

Status : ASSIGNED Date: 26-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By: TEST USER

System ID : 10000000953 Oracle ROWID: 00000033.000E.0018

Last Error : -0

Last Msg : FRM-40204 Cursor is at beginning of field value.

Table : METHOD

Block : METHOD

Field : METHOD.SIZE_BOTTLE

DESCRIPTION

field is tied to comp therefore will not permit numeric data and , + -
or . to be added and committed

~~NOT PASS~~ FIXED 11/30
~~MUST HAVE COMP~~
~~W/STOR~~
✓

Programmer's Bug Report

Application: LIBR Library functions

Transaction: LI5053U1 Laboratory Method Maintenance

=====
Comment ID : 672 Priority: LOW Date: 01-DEC-92
=====

Status : ASSIGNED Date: 01-DEC-92 Class:
Type : PROBLEM
Assigned To: TODD ADAMS
Entered By : LARRY TOWNER
System ID : 10000002670 Oracle ROWID: 000000BE.0004.0018
Last Error : -0
Last Msg : FRM-40350 Query caused no records to be retrieved.
Table : METHOD
Block : METHOD
Field : METHOD.IDENT_METHOD

DESCRIPTION

-
1. While modifying fields, committing & retrieving records the process seemed to loose records. — *FIXED SORT*
 2. After modifying and committing 'type' field, always defaults to 'ME'.

— *REMOVED TYPE FROM DISPUTA
AND DEFAULT TO "ME"*

12/1/92

3.11 LI5053U2 ANALYSIS CLASS MAINTENANCE

LEMIS Module Test Checklist Report

Name: ANALYSIS CLASS MAINTENANCE
 Short Name: LI5053U2
 Purpose: Update the library of analysis classes

Initial Test Date: 11/25/92 LC

Verification Date: 11/30/92 RYA CNO 60651

	Disp	Security			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: METHOD

Title: Laboratory Methods

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	30	()	()	()	()	()
TITLE_METHOD	CHAR	35	()	()	()	()	()

Block: METHOD

Title: Analysis Class

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry		
()	()	()	()	()	()	()	()		
Data Element			Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD			CHAR	12	()	()	()	()	()
TITLE_METHOD			CHAR	40	()	()	()	()	()
FLAG_TEXT			CHAR	1	()	()	()	()	()

Block: METHOD_NEST

Title: Associated Methods

Help ()	Scroll ()	Text ()	Navigate ()	Ins ()	Del ()	Upd ()	Qry ()		
Data Element			Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
RELATIONSHIP			CHAR	3	()	()	()	()	()

Module Description

The module maintains analysis classes. Analysis classes may be added, modified or deleted, but may not be modified or deleted if they are existing in live records on the database.

Method is a hierarchical structure. Through use of the method_nest table, methods may be associated with each other. There are three levels of method identified. Each serves a specific purpose. They are:

Special Analysis Classes
 Analysis Classes
 Methods

The special analysis classes are ones that add or subtract constituents from analysis classes. For example: ICP metals plus Sn. In this example, the special analysis class would link to the analysis class ICP metals with a relationship of REL and to the constituent Sn with a relationship of PLS.

Analysis classes are groupings of methods that are used by labs to detect that group of constituents. Analysis classes do not directly link to constituents.

Methods are the specific laboratory test procedures that are used to detect one or more specific constituents. The constituent associated with a method are linked to that method.

Module Description

This module supports analysis classes. An analysis class may NOT be associated directly with constituents.

Text may be associated with the analysis class.

An analysis class may be associated with one or more methods. Provide the capability to display 3-6 on the screen.

The default relationship code is REL. Where a default method is identified for an analysis class, the relationship code is DEF.

The method type code defaults to AC.

Holding time, extraction time, bottle type, and preservative are not used with this module.

The validation level for the data defaults to 9.
Ident-owner defaults to 'library'.

This level of method is not associated with a laboratory.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

Test status conditions are:

Module Test Plan

- () 1 - Transaction under development, untested.
 () Transaction added to system and security files.
 () Specific test plan/test results/test case added.
 () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 () Help added
 () LOV tested
 Developer JA Date Complete 11/25/92
- () 3 - Initial unit testing complete.
 Developer JA Date Complete 11/25/92

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
 () 5 - Secondary testing complete, problems found.
 Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
 () 7 - Problem resolution complete.
 Developer _____ Date Complete _____
- () 8 - Secondary testing complete, no problems found.
 Tester JE Date Complete 11/25/92

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Module Test Plan

Test status conditions are:

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.
Tester _____ Date Complete _____

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.

Project Manager SAJ Date Complete 11/30/92

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu.

System Architect SA Date Complete 11/30/92

Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
- (✓) Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.

- (/) Transaction help describes:
 - (/) Purpose of the transaction
 - (/) Functions performed by the transaction
 - () Where multiple page, what happens on each page
- (/) Add a new record by using the insert key.
 - (/) The insert key clears all blocks of the entry screen.
 - (/) A new record can be successfully added and committed.
- (/) Add a new record by scrolling past the last entry using the down arrow key.
 - () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - (/) All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (/) A new data record can be successfully added.
- (/) Query a single record.
 - (/) A single record is retrieved.
 - (/) Pressing ~~SHIFT~~/F12 clears the record and permits entering a second query.
 - (/) The second query successfully retrieves the desired data.
- (/) Query multiple records using wild cards.
 - (/) Pressing F12 clears the screen and makes the transaction ready for a query.
 - (/) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - (/) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (/) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- 604 (/) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (/) Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
- (/) Delete an existing record.
 - (/) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
- () All data fields are correctly stored in the database.

Block: METHOD

Title: Analysis Class

Block Description

A laboratory procedure is equivalent to an analytical method. It is the analysis process used to determine the presence of one or more constituents. The table contains those procedures that have been identified to support detection of constituents.

Laboratory procedures may be associated with one or more constituents. They may be also associated with the laboratories that are qualified to perform the procedures.

The data for all lab procedures is kept at validation level 9 (fully validated). Data is available to all users of the system.

Some samples may share the same method. Lab names are not needed to define methods because they are set by procedures specified by the customer. Methods might include:

TOC (total organic carbon)
 ICP (inductively coupled plasma)
 IC (ion chromatography)
 PH

Each method may have up to 150 constituents, or analytes.

Block Test Case

Block Test Results Expected

Field Name: IDENT_METHOD

Block: METHOD
Title: Analysis Class

Field Description

This field is the user visible unique identifier for the lab
Field Test Case

- n/A () Begin field with a space.
- ✓ () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - ✓ () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ✓ () Set keyboard on lower case and enter data. Data should be converted to upper case.

Field Test Results Expected and attempt to update the record.

- ✓ () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - ✓ () Purpose of the field
 - () LOV reference if available
 - ✓ () Valid range of values where present
 - ✓ () Required indicator when field is NOT NULL
 - ✓ () Default value when appropriate
- n/A ✓ () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- n/A ✓ () Store a series of records using the SHIFT/^{FLC} [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that

Block: METHOD
 Title: Analysis Class

Field Test Results Expected

the records changed and deleted above are entered into the journal table correctly.

- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Delete an existing value.
 - (✓) For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.
- (✓) Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- (✓) Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

DUP

Block: METHOD
Title: Analysis Class

Field Name: TITLE_METHOD

Field Description

Enter a 1-40 character descriptive title for the procedure.

Field Test Case

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

Block: METHOD
 Title: Analysis Class

Field Name: TITLE_METHOD

Field Test Results Expected

- fc*
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
 - () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
 - () Any alphanumeric characters may be entered into the field.
 - () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
 - () Attempt to enter data as lower case. An error should result.
 - () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- DWP*

Block: METHOD
 Title: Analysis Class

NO Testing

Field Name: FLAG_TEXT

Field Description

This field is a flag that identifies when supporting text is present for a data row. The valid values are either "Y" (yes) or "N" (no). When the user desires to add text in support of a data row, the flag is set to "Y". The system then brings up the general text form and permits the user to enter text of varying types for the data row. Where text is already present, the user places the cursor on the flag field and selects to retrieve the text data. Text data may be updated. Note that updating text data causes the same hierarchical cloning that takes place elsewhere when a data row is updated. This ensures that the text data is also properly audit trail covered.

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:

Block: METHOD
 Title: Analysis Class

Field Name: FLAG_TEXT

Field Test Results Expected

- () Purpose of the field
- () LOV reference if available
- () Valid range of values where present
- () Required indicator when field is NOT NULL
- () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the _JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not

Block: METHOD
Title: Analysis Class

Field Name: FLAG_TEXT

Field Test Results Expected

- be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
 - () Attempt to enter data as lower case. An error should result.
 - () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
 - () Attempt to bypass the field. An error message should tell you that the field data must be entered.
 - () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: METHOD_NEST
 Title: Associated Methods

Block Description

Block Test Case

Block Test Results Expected

Field Name: RELATIONSHIP

Field Description

The relationship identifies the nature of the linkage between two occurrences of the table parents. A default value is typically assigned. Other values are found in a LOV supported by TYPE_RELATIONSHIP_CODE.

Field Test Case

- 605 () Accept the default.
- (/) Override the default with a valid entry value.
- N/A () Override the default with an invalid data value.
- N/A () Begin field with a space.
- (/) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (/) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (/) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (/) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (/) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

Block: METHOD_NEST
Title: Associated Methods

Field Name: RELATIONSHIP

Field Test Results Expected

- ☒ Field help describes:
 - ☒ Purpose of the field
 - ☒ LOV reference if available
 - ☒ Valid range of values where present
 - ☒ Required indicator when field is NOT NULL
 - ☒ Default value when appropriate
- ☒ () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- ☒ Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- ☒ Modify each default value with a valid alternative.
- ☒ () Modify each default value with an invalid alternative.
- ☒ Any alphanumeric characters may be entered into the field.
- ☒ () Delete an existing value.

Block: METHOD_NEST
Title: Associated Methods

Field Name: RELATIONSHIP

Field Test Results Expected

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

n/a () Attempt to bypass the field. An error message should tell you that the field data must be entered.

n/a () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

/ () Attempt to enter data as lower case. ~~An error should result.~~

n/a () Delete an existing value.

() For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

() Attempt to bypass the field. An error message should tell you that the field data must be entered. *Das*

() Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Programmer's Bug Report

Transaction: LI5053U2 Analysis Class Support

=====

Comment ID : 601 Priority: LOW Date: 25-NOV-92

=====

Status : ASSIGNED Date: 26-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By: TEST USER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40350 Query caused no records to be retrieved.

Table : METHOD

Block : METHOD

Field : METHOD.IDENT_METHOD

DESCRIPTION

module name is different on printout vs menu

✓
NOT PROB
11/30/92

Transaction: LI5053U2 Analysis Class Support

=====

Comment ID : 604 Priority: LOW Date: 25-NOV-92

=====

Status : ASSIGNED Date: 26-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000001029 Oracle ROWID: 00000033.000F.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : METHOD

Block : METHOD

Field : METHOD.IDENT_METHOD

DESCRIPTION

ID field allowed update and commit

NOT PROB

11/30/92

✓

Transaction: LI5053U2 Analysis Class Support

```
=====
Comment ID : 605    Priority: LOW          Date: 25-NOV-92
=====
Status      : ASSIGNED    Date: 26-NOV-92    Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : TEST USER
System ID   : 10000000669      Oracle ROWID: 0000003A.0008.0018
Last Error  : -0
Last Msg    : FRM-40400 Transaction complete -- 2 records posted an
Table       : METHOD_NEST
Block       : METNST
Field       : METNST.DSP_IDENT_METHOD
```

DESCRIPTION

trailing spaces in method ID LOV. will not commit

NOT PROBLEM

11/30/92

✓

3.12 LI5053U3 ANALYSIS CLASS PLUS MAINTENANCE

LEMIS Module Test Checklist Report

Name: ANALYSIS CLASS PLUS MAINTENANC
 Short Name: LI5053U3
 Purpose: Update the library of special analysis classe

Initial Test Date: 11/30/92 *A*

Verification Date: 12/1/92 *JK*

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: CONSTITUENT
 Title: Constituents

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_CONSTITUENT	CHAR	30	()	()	()	()	()
NAME_CON_LONG	CHAR	60	()	()	()	()	()

Block: METHOD
 Title: Analysis Classes

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	30	()	()	()	()	()
TITLE_METHOD	CHAR	35	()	()	()	()	()

Block: METHOD

Title: Analysis Class Plus/Minus

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	30	()	()	()	()	()
TITLE_METHOD	CHAR	40	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: METHOD_CONSTITUENT

Title: Plus/Minus Constituents

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
RELATIONSHIP	CHAR	3	()	()	()	()	()

Block: METHOD_NEST

Title: Associated Analysis Classes

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins

Module Description

The module maintains special analysis classes. Classes may be added, modified or deleted, but may not be modified or deleted if they are existing in live records on the database.

The ability to associate special constituents is provided.

Method is a hierarchical structure. Through use of the method_nest table, methods may be associated with each other. There are three levels of method identified. Each serves a specific purpose. They are:

Special Analysis Classes
 Analysis Classes
 Methods

The special analysis classes are ones that add or subtract constituents from analysis classes. For example: ICP metals plus Sn.

Module Description

In this example, the special analysis class would link to the analysis class ICP metals with a relationship of REL and to the constituent Sn with a relationship of PLS.

Analysis classes are groupings of methods that are used by labs to detect that group of constituents. Analysis classes do not directly link to constituents.

Methods are the specific laboratory test procedures that are used to detect one or more specific constituents. The constituent associated with a method are linked to that method.

This module supports maintenance of the special analysis classes. It may link only to analysis classes and to constituents as described above.

Text may be associated with the special analysis class.

The method type code defaults to AP.

Extraction time, holding time, bottle type, and preservative are not used with the screen.

The validation level for the data defaults to 9.
Ident-owner defaults to 'library'.

This level of method is not associated with a laboratory.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

Module Test Plan

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

Test status conditions are:

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
 - Developer RA Date Complete 11/30/92
- () 3 - Initial unit testing complete.
 - Developer RA Date Complete 11/30/92

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
 - Tester _____ Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
 - Developer _____ Date Complete _____
- () 8 - Secondary testing complete, no problems found.
 - Tester JE Date Complete 11/30/92

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off

Module Test Plan

the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

Test status conditions are:

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.
Tester _____ Date Complete _____

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

- () 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.
 - () Reviews tests results.
 - () Reviews problem reports to ensure all corrections done.
 - () Updates NOTES text to reflect testing results.
 - Project Manager EJ Date Complete 12/1/92
- () 20 - The transaction is placed into the production environment.
 - () Software files in production directories.
 - () DMR 550 updated to add transaction.
 - () DMR 900 updated to add transaction.
 - () Transaction added to production usage menu.
 - System Architect JY Date Complete 12/1/92

Module Test Case

- (✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.

- (✓) Duplicate row insertion attempts result in error message
COM00102 [Duplicate value, enter new value or F1 for help]
- (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - (✓) Transaction help describes:
 - (✓) Purpose of the transaction
 - (✓) Functions performed by the transaction
 - (✓) Where multiple page, what happens on each page
- (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - () A new record can be successfully added and committed.
- (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - (✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - (✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing ~~SHIFT~~F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
- () Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
- (✓) Change each field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change each field individually, committing after each change.
 - (✓) Change each field, as per the field test plan, for both valid

and invalid values.

(✓) Delete an existing record.

(✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.

(B) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

(✓) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

(✓) Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date-updated, owner role, and any other autogenerated fields.

(✓) All data fields are correctly stored in the database.

Block: METHOD

Title: Analysis Class Plus/Minus

Block Description

A laboratory procedure is equivalent to an analytical method. It is the analysis process used to determine the presence of one or more constituents. The table contains those procedures that have been identified to support detection of constituents.

Laboratory procedures may be associated with one or more constituents. They may be also associated with the laboratories that are qualified to perform the procedures.

The data for all lab procedures is kept at validation level 9 (fully validated). Data is available to all users of the system.

Some samples may share the same method. Lab names are not needed to define methods because they are set by procedures specified by the customer. Methods might include:

TOC	(total organic carbon)
ICP	(inductively coupled plasma)
IC	(ion chromatography)
PH	

Each method may have up to 150 constituents, or analytes.

Block: METHOD
 Title: Analysis Class Plus/Minus

Block Test Case

Block Test Results Expected

Field Name: IDENT_METHOD

Field Description

This field is the user visible unique identifier for the lab procedure.

Field Test Case

- (✓) Begin field with a space.
- (✓) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

QUP

Block: METHOD

Title: Analysis Class Plus/Minus

Field Name: IDENT_METHOD

Field Test Results Expected

- ☒ Change the field, committing after each change.
- ☒ Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- ☒ Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- ☒ Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly. *SKIP*
- () All data fields are correctly stored in the database.
- ☒ Any alphanumeric characters may be entered into the field.
- ☒ Delete an existing value. *40102*
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- ☒ Attempt to bypass the field. An error message should tell you that the field data must be entered.
- ☒ Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- ☒ Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. *OK*

Block: METHOD
Title: Analysis Class Plus/Minus

Field Name: IDENT_METHOD

Field Test Results Expected

- () Attempt to bypass the field. An error message should tell you that the field data must be entered. *WP*
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result. *WP*

Block: METHOD
 Title: Analysis Class Plus/Minus

Field Name: TITLE_METHOD

Field Description

Enter a 1-40 character descriptive title for the procedure.

Field Test Case

- (X) Begin field with a space.
- (X) Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (X) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (X) Clear an existing field and attempt to update the record.

Field Test Results Expected

- (X) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (X) Field help describes:
 - (X) Purpose of the field
 - (X) LOV reference if available
 - (X) Valid range of values where present
 - (X) Required indicator when field is NOT NULL
 - (X) Default value when appropriate
- (X) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (X) Change the field, committing after each change.
 - (X) Change the field where a range of valid values exists:
 - (X) Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

Block: METHOD
Title: Analysis Class Plus/Minus

Field Name: TITLE_METHOD

Field Test Results Expected

- ☒ Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database. *SKIP*
- () Use SQL*Plus to verify all data fields are processed correctly. *SKIP*
- () All data fields are correctly stored in the database.
- ☒ Any alphanumeric characters may be entered into the field.
- ☒ Delete an existing value.
 - ☒ For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- ☒ Attempt to bypass the field. An error message should tell you that the field data must be entered. *DUP*
- ☒ Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- ☒ Attempt to enter data as lower case. An error should result.
- ☒ Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. *DUP*
- () Attempt to bypass the field. An error message should tell you that the field data must be entered. *DUP*
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result. *DUP*

Block: METHOD
 Title: Analysis Class Plus/Minus

Field Name: FLAG_TEXT

Field Description

This field is a flag that identifies when supporting text is present for a data row. The valid values are either "Y" (yes) or "N" (no). When the user desires to add text in support of a data row, the flag is set to "Y". The system then brings up the general text form and permits the user to enter text of varying types for the data row. Where text is already present, the user places the cursor on the flag field and selects to retrieve the text data. Text data may be updated. Note that updating text data causes the same hierarchical cloning that takes place elsewhere when a data row is updated. This ensures that the text data is also properly audit trail covered.

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- () Field help describes:

Block: METHOD
 Title: Analysis Class Plus/Minus

Field Name: FLAG_TEXT

Field Test Results Expected

- () Purpose of the field
- () LOV reference if available
- () Valid range of values where present
- () Required indicator when field is NOT NULL
- () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not

Block: METHOD

Title: Analysis Class Plus/Minus

Field Name: FLAG_TEXT

Field Test Results Expected

be deleted] is displayed at commit.

- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: METHOD CONSTITUENT
Title: Plus/Minus Constituents

Block Description

This table links a lab method with a constituent that may be identified through use of the method.

Block Test Case

Block Test Results Expected

Field Name: RELATIONSHIP

Field Description

Field Test Case

- ☒ Accept the default.
- ☒ Override the default with a valid entry value.
- ☒ Override the default with an invalid data value.
- ☒ Begin field with a space.
- ☒ Enter data beyond length of field.
 - ☒ If autoskip is in effect, cursor will skip to the next field.
 - ☒ If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- ☒ Set keyboard on lower case and enter data. Data should be converted to upper case.
- ☒ Clear an existing field and attempt to update the record.

Field Test Results Expected

- ☒ Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - ☒ Field help describes:
 - ☒ Purpose of the field

Block: METHOD CONSTITUENT
 Title: Plus/Minus Constituents

Field Name: RELATIONSHIP

Field Test Results Expected

- ☒ LOV reference if available
- ☒ Valid range of values where present
- ☒ Required indicator when field is NOT NULL
- ☒ Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made. *Don't*
- () Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly. *SKIP*
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database. *SKIP*
- () Modify each default value with a valid alternative. *DND*
- () Modify each default value with an invalid alternative. *DEP*
- ☒ Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.

MISSING Ident field

Block: METHOD_CONSTITUENT
Title: Plus/Minus Constituents

Field Name: RELATIONSHIP

Field Test Results Expected

- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- () Attempt to enter data as lower case. An error should result.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.

Block: METHOD_NEST
Title: Associated Analysis Classes

Block Description

Block Test Case

Block Test Results Expected

missing fields

Transaction: LI5053U3 Special Analysis Class Support

```
=====
Comment ID : 659    Priority: LOW          Date: 30-NOV-92
=====
Status      : ASSIGNED    Date: 30-NOV-92    Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : KEN ATKINS
System ID   : 10000002716    Oracle ROWID: 0000003A.000E.0018
Last Error  : -0
Last Msg    : FRM-40400 Transaction complete -- 1 records posted an
Table       : METHOD_NEST
Block       : METNST
Field       : METNST.DSP_IDENT_METHOD
=====
```

DESCRIPTION

The constituent plus/min block does not sync with the analysis class block. Shouldn't it? if not the boxes should be fixed.

FIXED.

LIMITED NEST
BLOCK TO 1 RECORD
12/1/92
✓

Programmer's Bug Report

Transaction: LI5053U3 Special Analysis Class Support

=====

Comment ID : 661 Priority: LOW Date: 30-NOV-92

=====

Status : ASSIGNED Date: 30-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By: KEN ATKINS

System ID : 10000002716 Oracle ROWID: 0000003A.000E.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : METHOD_NEST

Block : METNST

Field : METNST.DSP_IDENT_METHOD

DESCRIPTION

Allowed the same constituent with both PLS and MIN relationships. This
does not make sense.

TABLE

12/1/92

✓

3.13 LI5055U1 ASSIGN CONSTITUENTS TO METHODS

LEMIS Module Test Checklist Report

Name: ASSIGN CONSTITUENTS TO METHODS

Short Name: LI5055U1

Purpose: Identify all constituents related to a metho

Initial Test Date: 11/30/92 SK

Verification Date: 11/30/92 RJA (No Bug)

Help	Disp	Security			
		Ins	Del	Query	Upd
()	()	()	()	()	()

Block: CONSTITUENT
Title: Constituents

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_CONSTITUENT	CHAR	30	()	()	()	()	()
NAME_CON_LONG	CHAR	60	()	()	()	()	()
TYPE_CONSTITUENT	CHAR	2	()	()	()	()	()

Block: METHOD

Title: Laboratory Method

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	12	()	()	()	()	()
TITLE_METHOD	CHAR	36	()	()	()	()	()

Block: METHOD_CONSTITUENT

Title: Constituents Tested For By Methods

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
CONTRACT_DETECT_LIM	NUMBER	18	()	()	()	()	()
UNIT_MEASURE	CHAR	5	()	()	()	()	()
DATE_LIMIT_BEGIN	DATE	8	()	()	()	()	()
DATE_LIMIT_END	DATE	8	()	()	()	()	()
LAB_PERIOD	NUMBER	4	()	()	()	()	()
TYPE_LAB_MEDIA	CHAR	1	()	()	()	()	()
PCT_ACCURACY	NUMBER	5	()	()	()	()	()
REL_STD_DEV	NUMBER	8	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: TYPE_CONSTITUENT_CODE

Title: Types of Constituent Data Records

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

This module will be used to assign 1 or more constituents to a given analytical method. The screen is primarily used to add more constituents to an existing module.

The process involves storing the method_constituent table occurrences, one for each association. The relationship code defaults to REL (related).

Module Description

For each association, the data found in method_constituent should be stored.

The validation level for the data defaults to 9.
Ident-owner defaults to 'library'.

Module Help

Use this module to assign constituents to a given lab procedure.

The method identifier is entered in the Method ID field and the RUN QY key combination pressed. The method title will be displayed.

To link constituents to the methods testing for them:

1. Press the PAGE DOWN key to move the cursor to the Const ID line.
2. Enter the constituent identifier or use LOV to select the desired constituent. If using LOV:
 - a. Press the LOV key to display the available constituents.
 - b. Use the up and down arrow keys to select the desired constituent.
 - c. Press the COMMIT key (F10) to move the constituent to the screen.
3. When selected, the constituent name and type will be displayed.
4. The default relationship code is REL. Use LOV to identify other alternate relationships between constituent and method.
5. TAB to the Volume Threshold. Enter the volume and associated unit of measure.
6. Enter the contractual date the limitation begins and ends.
7. Enter the lab period, media, and acceptable accuracy percentage.
8. Enter the relative standard deviation.
9. Press the COMMIT key (F10) to store the association information in the database.

The Text? flag indicates whether free-form text is associated with the constituent/method association. If the flag is set to 'N', no text is present. To add text, press the TEXT key (F7). A popup screen will appear that will permit you to enter as much text as desired. Several different types of text may be available. Once the text has been entered, the text flag is set to 'Y'.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

Module Test Plan

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

Test status conditions are:

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.

Developer _____	Date Complete _____
-----------------	---------------------
- () 8 - Secondary testing complete, no problems found.

Tester _____	Date Complete _____
--------------	---------------------

When problems are found:

1. The tester passes the checklist to the project manager for review and action.

Module Test Plan

2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.

3. The developer corrects the described problems; records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.

4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.
- () 10 - System testing complete, problems found.
Tester _____ Date Complete _____
- () 11 - Problem resolution in progress.
- () 12 - Problem resolution complete.
Developer _____ Date Complete _____
- () 13 - System testing complete, no problems found.

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

() 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.

- () Reviews tests results.
- () Reviews problem reports to ensure all corrections done.
- () Updates NOTES text to reflect testing results.
- Project Manager [Signature] Date Complete 1/30/92

() 20 - The transaction is placed into the production environment.

- () Software files in production directories.
- () DMR 550 updated to add transaction.
- () DMR 900 updated to add transaction.
- () Transaction added to production usage menu
- System Architect [Signature] Date Complete 1/30/92

Module Test Case

- NA
- () Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 - () Duplicate row insertion attempts result in error message COM00102 [Duplicate value, enter new value or F1 for help]
 - (✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 - (✓) Transaction help describes:
 - (✓) Purpose of the transaction
 - (✓) Functions performed by the transaction
 - (✓) Where multiple page, what happens on each page
 - (✓) Add a new record by using the insert key.
 - (✓) The insert key clears all blocks of the entry screen.
 - (✓) A new record can be successfully added and committed.
 - (✓) Add a new record by scrolling past the last entry using the down arrow key.
 - (✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - (✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
 - (✓) A new data record can be successfully added.
 - (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing SHIFT/F12 clears the record and permits entering a second query.
 - (✓) The second query successfully retrieves the desired data.
 - (✓) Query multiple records using wild cards.
 - (✓) Pressing F12 clears the screen and makes the transaction ready for a query.
 - (✓) A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through
 - (✓) A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - (✓) A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through SQL*Plus.
 - (✓) Change each field in an existing record. Verify that the change

occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

- (✓) Change each field individually, committing after each change.
- (✓) Change each field, as per the field test plan, for both valid and invalid values.
- (✓) Delete an existing record.
 - () The selected record is deleted from the database. Verify ~~by using SQL*Plus to attempt to retrieve the record.~~
 - () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - NK () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are stored correctly, including ID_ keys, date-updated, owner role, and any other autogenerated fields.
 - () All data fields are correctly stored in the database.

Block: METHOD_CONSTITUENT

Title: Constituents Tested For By Methods

Block Description

This table links a lab method with a constituent that may be identified through use of the method.

Block Test Case

Block Test Results Expected

Field Name: CONTRACT_DETECT_LIM

Field Description

Field Test Case

Block: METHOD_CONSTITUENT

Title: Constituents Tested For By Methods

Field Name: CONTRACT_DETECT_LIM

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Attempt to enter alpha data.
- (✓) Attempt to enter longer value than allowed.
- (✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- (✓) Attempt to enter negative value. No error should occur.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (NA) LOV reference if available
 - (✓) Valid range of values where present
 - (NA) Required indicator when field is NOT NULL
 - (NA) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change the field, committing after each change.
 - () Change the field where a range of valid values exists:
 - (✓) Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - (NA) Value above valid range should display message COM00101 or

Block: METHOD_CONSTITUENT
 Title: Constituents Tested For By Methods

Field Name: CONTRACT_DETECT_LIM

Field Test Results Expected

- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- NR () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
- (✓) Begin field entry with a space. An error should occur.

no error, but

Left Justified

Block: METHOD_CONSTITUENT

Title: Constituents Tested For By Methods

Field Name: UNIT_MEASURE

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Accept the default.
- (✓) Override the default with a valid entry value.
- (✓) Override the default with an invalid data value.
- (✓) Begin field with a space. *won't work*
- (✓) Enter data beyond length of field.
 - ~~(X)~~ If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.
- (✓) Clear an existing field and attempt to update the record. *block OK*

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- 65* (✓) Field help describes:
 - (✓) Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present

Block: METHOD_CONSTITUENT
 Title: Constituents Tested For By Methods

Field Name: UNIT_MEASURE

Field Test Results Expected

- (✓) Required indicator when field is NOT NULL
- (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; ~~display the record using SQL*Plus before and after the changes were made.~~
- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/⁶~~F7~~ [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
 - () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Modify each default value with a valid alternative.
- (✓) Modify each default value with an invalid alternative.

Block: METHOD_CONSTITUENT

Title: Constituents Tested For By Methods

Field Name: UNIT_MEASURE

Field Test Results Expected

- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. An error should result.
- (✓) Delete an existing value.

() For NOT NULL fields, the COM00107 message [field value may not
NA be deleted] is displayed at commit.

() Attempt to bypass the field. An error message should tell you
NA that the field data must be entered.

() Enter data into the field, backspace the data out of the field and
attempt to TAB to the next field. An error should result.

*as long as it
exists on table*

Block: METHOD_CONSTITUENT
 Title: Constituents Tested For By Methods

Field Name: DATE_LIMIT_BEGIN

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Attempt to store an invalid date:
 - (✓) Enter the day as 00.
 - (✓) Enter the day as 32.
 - (✓) Enter the day as 30 in February.
 - (✓) Enter the day as 31 September.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✗) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✗) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change the field, committing after each change.
 - (✓) Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or

Block: METHOD_CONSTITUENT
Title: Constituents Tested For By Methods

Field Name: DATE_LIMIT_BEGIN

Field Test Results Expected

(✓) Store a series of records using the SHIFT/~~E~~⁶ [duplicate field] key to carry forward field values from the previously stored record.

() Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

HK () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

() Use SQL*Plus to verify all data fields are processed correctly.

(✓) Bypass the field by using the TAB key.

(✓) No error should occur.

(✓) The database contains a null value for the field.

(✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

(✓) Delete an existing value. The field should be cleared and stored as a null value.

(✓) Enter an invalid date. The invalid date should be rejected.

Block: METHOD_CONSTITUENT
 Title: Constituents Tested For By Methods

Field Name: DATE_LIMIT_END

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field; backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Attempt to store an invalid date:
 - (✓) Enter the day as 00.
 - (✓) Enter the day as 32.
 - (✓) Enter the day as 30 in February.
 - (✓) Enter the day as 31 September.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (H) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (H) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change the field, committing after each change.
 - (✓) Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or

Block: METHOD_CONSTITUENT

Title: Constituents Tested For By Methods

Field Name: DATE_LIMIT_END

Field Test Results Expected

- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- NA () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Enter an invalid date. The invalid date should be rejected.

Block: METHOD_CONSTITUENT
 Title: Constituents Tested For By Methods

Field Name: LAB_PERIOD

Field Description

Time, in weeks, that the analysis laboratory has to process a sample and return the analysis results.

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Attempt to enter alpha data.
- (✓) Attempt to enter longer value than allowed.
- (✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- (✓) Attempt to enter negative value. No error should occur.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

Block: METHOD_CONSTITUENT
Title: Constituents Tested For By Methods

Field Name: LAB_PERIOD

Field Test Results Expected

- (✓) Change the field, committing after each change.
- (✓) Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
- (✓) Begin field entry with a space. ~~An error should occur.~~ *Left Just. free*

Block: METHOD_CONSTITUENT
 Title: Constituents Tested For By Methods

Field Name: TYPE_LAB_MEDIA

Field Description

Identifies the medium that the limit applies to. Choices are "S" for solid and "L" for liquid.

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (N/A) Begin field with a space. *1 Char Field*
- (✓) Enter data beyond length of field.
 - (N/A) If autoskip is in effect, cursor will skip to the next field.
 - (✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- (✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - (✓) Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and

Block: METHOD_CONSTITUENT
Title: Constituents Tested For By Methods

Field Name: TYPE_LAB_MEDIA

Field Test Results Expected

after the changes were made.

- (✓) Change the field, committing after each change.
- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~

only 'L' or 'S' are valid
shift to upper case

Block: METHOD_CONSTITUENT
 Title: Constituents Tested For By Methods

Field Name: PCT_ACCURACY

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Attempt to enter alpha data. *won't allow*
- (✓) Attempt to enter longer value than allowed.
- (✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.

- (✓) Attempt to enter negative value. No error should occur. *wrong! Range 0-100 enforced.*

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

- (✓) Field help describes:
 - (✓) Purpose of the field
 - (✗) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator, when field is NOT NULL
 - (✗) Default value when appropriate

- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SOL*Plus before and after the changes were made.

- (✓) Change the field, committing after each change.

Block: METHOD_CONSTITUENT

Title: Constituents Tested For By Methods

Field Name: PCT_ACCURACY

Field Test Results Expected

- (✓) Change the field where a range of valid values exists:
 - (✓) Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - (✓) Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
- NO* () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Numeric data (0-9), comma (,), plus (+), minus *NO* (-), and period (.) are permitted.
- (✓) Begin field entry with a space. ~~An error should occur.~~

Left Justified

Block: METHOD_CONSTITUENT
 Title: Constituents Tested For By Methods

Field Name: REL_STD_DEV

Field Description

Field Test Case

- (✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- (✓) Attempt to enter alpha data.
- (✓) Attempt to enter longer value than allowed.
- (✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- (✓) Attempt to enter negative value. No error should occur.

Field Test Results Expected

- (✓) Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - (✓) Field help describes:
 - () Purpose of the field
 - (✓) LOV reference if available
 - (✓) Valid range of values where present
 - (✓) Required indicator when field is NOT NULL
 - (✓) Default value when appropriate
- (✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - (✓) Change the field, committing after each change.

Block: METHOD_CONSTITUENT
Title: Constituents Tested For By Methods

Field Name: REL_STD_DEV

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - NK* () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - NK* () The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
- () All data fields are correctly stored in the database.
- (✓) Bypass the field by using the TAB key.
 - (✓) No error should occur.
 - (✓) The database contains a null value for the field.
- (✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.
- (✓) Delete an existing value. The field should be cleared and stored as a null value.
- (✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
- (✓) Begin field entry with a space. ~~An error should occur.~~

LEFT Justifies

Block: METHOD_CONSTITUENT
Title: Constituents Tested For By Methods

Field Name: FLAG_TEXT

Field Description

Field Test Case

- () Accept the default.
- () Override the default with a valid entry value.
- () Override the default with an invalid data value.
- () Begin field with a space.
- () Enter data beyond length of field.
 - () If autoskip is in effect, cursor will skip to the next field.
 - () If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.
- () Set keyboard on lower case and enter data. Data should be converted to upper case.
- () Clear an existing field and attempt to update the record.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
 - () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- () Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - () Change the field, committing after each change.

Block: METHOD_CONSTITUENT

Title: Constituents Tested For By Methods

Field Name: FLAG_TEXT

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- () Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.
- () Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - () The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.
- () Use SQL*Plus to verify all data fields are processed correctly.
 - () All data fields are correctly stored in the database.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit.
- () Attempt to bypass the field. An error message should tell you that the field data must be entered.
- () Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should result.
- () Attempt to enter data as lower case. An error should result.

Application: LIBR Library functions

ansaction: LI5055U1 Assign Constituents to Methods

=====
Comment ID : 651 Priority: LOW Date: 30-NOV-92
=====

Status : ASSIGNED Date: 30-NOV-92 Class:
Type : PROBLEM
Assigned To: TODD ADAMS
Entered By : TEST USER
System ID : 10000001394 Oracle ROWID: 00000035.0007.0018
Last Error : -0
Last Msg : FRM-40400 Transaction complete -- 1 records posted an
Table : METHOD_CONSTITUENT
Block : METCON
Field : METCON.UNIT_MEASURE

DESCRIPTION

Unit of measure field can be blank/null - help text says it cannot be blank.

FIXED HELP
11/30/92

3.14 LI5056U1 ASSIGN METHODS TO LABORATORY

LEMIS Module Test Checklist Report

Name: ASSIGN METHODS TO LABORATORY
 Short Name: LI5056U1
 Purpose: Assign methods to specific laboratories
 Initial Test Date: 11/20/92 JC 12/1/92 JK
 Verification Date: 12/1/92 JA O.K.

	Disp	----- Security -----			
Help	Fkeys	Ins	Del	Query	Upd
()	()	()	()	()	()

Block: LABORATORY
 Title: Laboratory Definition

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_LAB	CHAR	6	()	()	()	()	()
NAME_LAB	CHAR	40	()	()	()	()	()

Block: LAB_METHOD

Title: Methods Processed By A Laboratory

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_BOTTLE_COMPOSITIO	CHAR	2	()	()	()	()	()
SIZE_BOTTLE	NUMBER	5	()	()	()	()	()
QUAN_BOTTLE	NUMBER	2	()	()	()	()	()

Block: METHOD

Title: Laboratory Methods

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_METHOD	CHAR	12	()	()	()	()	()
TITLE_METHOD	CHAR	34	()	()	()	()	()

Module Description

This form is used to assign analysis methods to specific labs that are authorized to perform those procedures.

The default relationship is REL. Where a specific lab is preferred for the analysis of a specific method, the relationship is DEF (default).

The laboratory may specify a different default bottle type and size from that identified as standard for a method. A LOV to type_bottle_code is used. Where a different bottle type and size are specified, the number of bottles desired is also entered.

The default validation level is 9. The default ident-owner is 'library'.

Module Help

This screen assigns analytical methods to the laboratories that are qualified to perform the method within the requirements defined for the Hanford Site.

The screen identifies a specific laboratory to which methods will be assigned.

1. Enter the lab identifier and press shift-F12.

Module Help

2. The laboratory title will be displayed.

To assign the methods to the laboratory:

1. Press the PAGE DOWN key to move the cursor to the first METHOD ID line.
2. Press the down arrow key, if required, to move the cursor to the first blank line.
3. Press LOV to display the available methods.
4. Use the up and down arrow keys to select the desired method.
5. Press the COMMIT key (F10) to move the method to the screen.
6. To add more methods, press the down arrow key to the next blank space and repeat the selection process.

If all of the spaces are filled and more methods are to be added, press the down arrow key and the already entered methods will scroll upward, leaving space for an additional method to be added.

7. Press the COMMIT key (F10) to store the selected methods in the database associated with the displayed laboratory.

Module Test Plan

TEST PLAN

Testing of the transaction is performed at five levels:

1. Developer testing.
2. Secondary testing.
3. System testing.
4. End user testing.
5. Production certification.

As each applicable section of this test plan is completed, the designated individual responsible for the section will sign the plan in the appropriate places. Each check off item will be completed. The status of the transaction will be updated in the data dictionary as the transaction progresses through the development and test steps.

Developer Testing

As each individual transaction is generated, the developer performs tests against the transaction as defined in the detail test plan. Upon successful completion of this testing, the transaction is made available for secondary testing.

Test status conditions are:

Module Test Plan

- () 1 - Transaction under development, untested.
 - () Transaction added to system and security files.
 - () Specific test plan/test results/test case added.
 - () Test plan narrative script loaded.
- () 2 - Development complete, initial unit testing in progress.
 - () Help added
 - () LOV tested
- () 3 - Initial unit testing complete.

Secondary Testing

Secondary testing is performed by a person external to the development team. Tests are performed using a standardized test checklist. Initial testing is performed to ensure that the transaction meets the functional requirements of the transaction and satisfies the test plan narrative that follows.

- () 4 - Secondary testing in progress.
- () 5 - Secondary testing complete, problems found.
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
 - Developer _____ Date Complete _____
- () 8 - Secondary testing complete; no problems found.
 - Tester Date Complete 11/20/92

When problems are found:

1. The tester passes the checklist to the project manager for review and action.
2. The project manager assigns the checklist to the developer of the transaction for correction of the problems.
3. The developer corrects the described problems, records the corrective actions taken and the number of hours of corrective effort used, tests the transaction for correct operation, and returns the checklist to the project manager.
4. The project manager returns the checklist to the tester for retesting.

The above sequence of events repeats until the tester signs off the transaction as passing the test requirements.

System Testing

System testing is performed by a second independent tester. This tester repeats the test plan scenario, the test cases, and any other tests desired under end-user operating conditions.

- () 9 - System testing in progress.

Module Test Plan

- () 10 - System testing complete, problems found.
 () 11 - Problem resolution in progress.
 () 12 - Problem resolution complete.
 Developer _____ Date Complete _____
 () 13 - System testing complete, no problems found.
 Tester _____ Date Complete _____

When problems are found, the same corrective action and retest event sequence as defined in secondary testing above are performed.

End User Testing

The transaction is tested by end users. When appropriate, problem resolution takes place as in previous test segments.

Production Status

- () 19 - The Project Manager reviews all test results and determines that the transaction has successfully completed testing.
 () Reviews tests results.
 () Reviews problem reports to ensure all corrections done.
 () Updates NOTES text to reflect testing results.
 Project Manager 221 Date Complete 12/1/92
 () 20 - The transaction is placed into the production environment.
 () Software files in production directories.
 () DMR 550 updated to add transaction.
 () DMR 900 updated to add transaction.
 () Transaction added to production usage menu
 System Architect JA Date Complete 12/1/92

Module Test Case

- ✓(✓) Attempt to store duplicate records in the database. Duplicate records are those where the unique identifier or any unique field contains duplicate values over 2 or more rows.
 () Duplicate row insertion attempts result in error message
 ~~EQM00102 [Duplicate value, enter new value or F1 for help]~~
 ✓(✓) Display transaction help. Verify that the narrative adequately describes the purpose and functions of the transaction in end-user terms.
 ✓(✓) Transaction help describes:
 ✓(✓) Purpose of the transaction
 ✓(✓) Functions performed by the transaction
 ✓(✓) Where multiple page, what happens on each page

- ✓(✓) Add a new record by using the insert key.
 - ✓(✓) The insert key clears all blocks of the entry screen.
 - ✓(✓) A new record can be successfully added and committed.
- ✓(✓) Add a new record by scrolling past the last entry using the down arrow key.
 - ✓(✓) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - ✓(✓) All fields of the transaction blank when the end of the string of retrieved records is reached.
 - ✓(✓) A new data record can be successfully added.
- ✓(✓) Query a single record.
 - ✓(✓) A single record is retrieved.
 - ✓(✓) Pressing SHIFT/F12 clears the record and permits entering a second query.
 - ✓(✓) The second query successfully retrieves the desired data.
- ✓(✓) Query multiple records using wild cards.
 - () Pressing F12 clears the screen and makes the transaction ready for a query.
 - () A query composed of a trailing % wildcard retrieves the desired data. Verify by running the same query through SQL*Plus.
 - () A query using a prefix % wildcard retrieves the correct data. Verify by running the same query under SQL*Plus.
 - () A query using both prefix and suffix % wildcards retrieves the correct data. Verify by running the same query through
- ✓(✓) Change each ^{updatable} field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
 - ✓(✓) Change each field individually, committing after each change.
 - () Change each field, as per the field test plan, for both valid and invalid values.
- ✓(✓) Delete an existing record.
 - (✓) The selected record is deleted from the database. Verify by using SQL*Plus to attempt to retrieve the record.
- 678 (✓) Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.
 - (✓) The _JN journal table for the affected table should contain rows for the initial addition of the record to the database,

ERROR-FRM-40735
Pre-delete trigger
raised unhandled
exception ORA-01400.

each change made to the database, and the deletion of the record from the database.

(✓) Use SQL*Plus to verify all data fields are stored correctly, including ID keys, date-updated, owner role, and any other autogenerated fields.

(✓) All data fields are correctly stored in the database.

Block: LAB METHOD

Title: Methods Processed By A Laboratory

Block Description

This table associates a laboratory with a specific lab procedure.

Block Test Case

Block Test Results Expected

Field Name: TYPE_BOTTLE_COMPOSITION

Field Description

Field Test Case

(✓) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.

(✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.

(A) (✓) Begin field with a space.

(✓) Enter data beyond length of field.

() If autoskip is in effect, cursor will skip to the next field.

(✓) If no autoskip, cursor will remain at end of the field and overlay characters, leaving the last character keyed in the last position of the field.

(✓) Set keyboard on lower case and enter data. Data should be converted to upper case.

READ FIELD

Block: LAB METHOD

Title: Methods Processed By A Laboratory

Field Name: TYPE_BOTTLE_COMPOSITION

Field Test Results Expected

55 ✓ Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

679 ✓ Field help describes:
 (✓) Purpose of the field
 (✓) LOV reference if available
 (✓) Valid range of values where present
 (✓) Required indicator when field is NOT NULL
 (✓) Default value when appropriate

✓ Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

✓ Change the field, committing after each change.
 () Change the field where a range of valid values exists:
 () Value below valid range should display message COM00101 or COM00103 (if no LOV).

✓ Store a series of records using the SHIFT ^{Fl}~~F~~ [duplicate field] key to carry forward field values from the previously stored record.

✓ Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

NA ✓ The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

(✓) Use SQL*Plus to verify all data fields are processed correctly.

(✓) Bypass the field by using the TAB key.

(✓) No error should occur.

(✓) The database contains a null value for the field.

(✓) Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

Reg'd
Field

Block: LAB_METHOD

Title: Methods Processed By A Laboratory

Field Name: TYPE_BOTTLE_COMPOSITION

Field Test Results Expected

(✓) Delete an existing value. The field should be cleared and stored as a null value.

52 (✓) Any alphanumeric characters may be entered into the field. -

*must match
L or*

(✓) Attempt to enter data as lower case. ~~An error should result.~~

All caps

Block: LAB_METHOD
Title: Methods Processed By A Laboratory

Field Name: SIZE_BOTTLE

Field Description

Field Test Case

- (u) TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.
- (u) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.
- ✓(u) Attempt to enter alpha data.
- ✓(u) Attempt to enter longer value than allowed.
- ✓(u) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.
- ✓(u) Attempt to enter negative value. No error should occur.

REC'D FIELD

Must match LOV

Field Test Results Expected

- 553 () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- 680 () Field help describes:
 - () Purpose of the field
 - () LOV reference if available
 - () Valid range of values where present
 - () Required indicator when field is NOT NULL
 - () Default value when appropriate
- ✓() Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.
- ✓() Change the field, committing after each change.

Block: LAB_METHOD

Title: Methods Processed By A Laboratory

Field Name: SIZE_BOTTLE

Field Test Results Expected

- () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).

✓/ Store a series of records using the SHIFT/F7 ^{fu} [duplicate field] key to carry forward field values from the previously stored record.

✓/ Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

✓/ The _JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

✓/ Use SQL*Plus to verify all data fields are processed correctly.

✓/ Bypass the field by using the TAB key.

✓/ No error should occur.

✓/ The database contains a null value for the field.

✓/ Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

✓/ Delete an existing value. The field should be cleared and stored as a null value.

554 ✓/ Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.

55 ✓/ Begin field entry with a space. ~~An error should occur.~~

cert justifies

Wrong Test

but must make
LOV

Block: LAB METHOD

Title: Methods Processed By A Laboratory

Field Name: QUAN_BOTTLEField Description

Identifies those procedures processed by a laboratory.

Field Test Case

556 () TAB through the field. Backtab to the field, enter data. After entering data in the next field, backtab to the field, and backspace the data from the field. The database should contain a null value for the field.

(✓) Enter data in the field. After entering data in the next field, backtab to the field, and space through the data. The database should contain a null value for the field.

✓(✓) Attempt to enter alpha data.

✓(✓) Attempt to enter longer value than allowed.

(✓) Attempt to enter decimal value. An error will occur if no provision has been made for decimal data in the element definition of the table.

✓(✓) Attempt to enter negative value. No error should occur.

array test

Com 30409

No - valid Range is 1-99

Field Test Results Expected

557 () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.

() Field help describes:

- () Purpose of the field
- () LOV reference if available
- () Valid range of values where present
- () Required indicator when field is NOT NULL
- () Default value when appropriate

Help: Range 1-99 not optional

681

✓(✓) Change the field in an existing record. Verify that the change occurred by displaying the record on screen before and after the changes were made; display the record using SQL*Plus before and after the changes were made.

Block: LAB_METHOD

Title: Methods Processed By A Laboratory

Field Name: QUAN_BOTTLE

Field Test Results Expected

- ☒ Change the field, committing after each change.
- ☐ Change the field where a range of valid values exists:
 - ☐ Value below valid range should display message COM00101 or COM00103 (if no LOV).

☒ Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

- ☒ Display the -JN journal record rows for the table to verify that the records changed and deleted above are entered into the journal table correctly.

- ☒ The -JN journal table for the affected table should contain rows for the initial addition of the record to the database, each change made to the database, and the deletion of the record from the database.

- ☒ Use SQL*Plus to verify all data fields are processed correctly.

- ☒ Bypass the field by using the TAB key.

- ☒ No error should occur.

- ☒ The database contains a null value for the field.

- ☒ Enter data into the field, backspace the data out of the field and TAB to the next field. The database should contain a null value.

- ☒ Delete an existing value. The field should be cleared and stored as a null value.

Required
Field
wrong
test

☒ Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.

Range
1-99

☒ Begin field entry with a space. ~~An error should occur.~~

NO, Left Justifies.

Programmer's Bug Report

Transaction: LI5056U1 Assign Methods to Laboratory

=====

Comment ID : 551 Priority: LOW Date: 22-NOV-92

=====

Status : ASSIGNED Date: 24-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By : TEST USER

System ID : 10000000656 Oracle ROWID: 00000017.0005.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : LAB_METHOD

Block : LABMTH

Field : LABMTH.TYPE_BOTTLE_COMPOSITION

DESCRIPTION

help not adequate.

✓ no description.
only INFO under
this data field.

La

Transaction: LI5056U1 Assign Methods to Laboratory

=====

Comment ID : 554 Priority: LOW Date: 22-NOV-92

=====

Status : ASSIGNED Date: 24-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000000656 Oracle ROWID: 00000017.0005.0018

Last Error : -0

Last Msg : FRM-50016 Legal characters are 0-9 - + E .

Table : LAB_METHOD

Block : LABMTH

Field : LABMTH.SIZE_BOTTLE

DESCRIPTION

NOT ABLE TO ADD A VALID VALUE (COMMA)

NOT PROB
11/30/92

Programmer's Bug Report

ransaction: LI5056U1 Assign Methods to Laboratory

=====

Comment ID : 556 Priority: LOW Date: 22-NOV-92

=====

Status : ASSIGNED Date: 24-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000000656 Oracle ROWID: 00000017.0004.0018

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 1 records posted an

Table : LAB_METHOD

Block : LABMTH

Field : LABMTH.QUAN_BOTTLE

DESCRIPTION

not able to tab back to first field

FIXED
11/30/92
✓

Transaction: LI5056U1 Assign Methods to Laboratory

```
=====
Comment ID : 557   Priority: LOW           Date: 22-NOV-92
=====
Status      : ASSIGNED   Date: 24-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : TEST USER
System ID   : 10000000656   Oracle ROWID: 00000017.0004.0018
Last Error  : -0
Last Msg    : FRM-50016   Legal characters are 0-9 - + E .
Table       : LAB_METHOD
Block       : LABMTH
Field       : LABMTH.QUAN_BOTTLE
=====
```

DESCRIPTION

message no special function allowed here com30409. does not allow
functions (decimal, comma, negative) to work. duplicate help listed.

CORRECTIVE ACTION

11/24/92 - KDA - The preference FLDNAV must be set to 'N'

F A E D

11/30/92

Transaction: LI5056U1 Assign Methods to Laboratory

```
=====
Comment ID : 552   Priority: LOW           Date: 22-NOV-92
=====
Status      : ASSIGNED   Date: 24-NOV-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : TEST USER
System ID   : 10000000656   Oracle ROWID: 00000017.0005.0018
Last Error  : -0
Last Msg    : FRM-40400 Transaction complete -- 1 records posted an
Table       : LAB_METHOD
Block       : LABMTH
Field       : LABMTH.QUAN_BOTTLE
=====
```

DESCRIPTION

trying to admit an invalid value should give an error message and not allow you to commit. and hint text needs fixing. Optional need only be noted once.

FIXED

FIXED

11/30/92

✓

~~(checked out 11/29)~~~~AUTO QUERY - TOP ROCK~~~~METHOD PICK-LIST DOESN'T WORK~~

Transaction: LI5056U1 Assign Methods to Laboratory

=====

Comment ID : 553 Priority: LOW Date: 22-NOV-92

=====

Status : ASSIGNED Date: 24-NOV-92 Class:

Type : PROBLEM

Assigned To: LARRY TOWNER

Entered By: TEST USER

System ID : 10000000656 Oracle ROWID: 00000017.0005.0018

Last Error : -0

Last Msg : FRM-50016 Legal characters are 0-9 - + E .

Table : LAB_METHOD

Block : LABMTH

Field : LABMTH.SIZE_BOTTLE

DESCRIPTION

HELP NEEDS MORE EXPLANATION

✓ no description
only info under
this data field!

✓ Also Hint text
has optional twice

Field lets you
add a character
and Commit it
then disappears

FIXED

11/30/92

Transaction: LI5056U1 Assign Methods to Laboratory

=====

Comment ID : 555 Priority: LOW Date: 22-NOV-92

=====

Status : ASSIGNED Date: 24-NOV-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : TEST USER

System ID : 10000000656 Oracle ROWID: 00000017.0005.0018

Last Error : -0

Last Msg : FRM-40407 Transaction complete -- posted records comm

Table : LAB_METHOD

Block : LABMTH

Field : LABMTH.SIZE_BOTTLE

DESCRIPTION

when beginning field with a space the space disappears when tabbing to
next field.

FIXED

11/30/92

01-DEC-92

WHC-SD-WM-TP-163 REV 0

Page 1 of 1

Programmer's Bug Report

Application: LIBR Library functions

Transaction: LI5056U1 Assign Methods to Laboratory

=====
Comment ID : 680 Priority: LOW Date: 01-DEC-92
=====

Status : ASSIGNED Date: 01-DEC-92 Class:
Type : PROBLEM
Assigned To: TODD ADAMS
Entered By : LARRY TOWNER
System ID : 10000001248 Oracle ROWID: 00000017.0007.0018
Last Error : -0
Last Msg : FRM-40202 Field must be entered.
Table : LAB_METHOD
Block : LABMTH
Field : LABMTH.SIZE_BOTTLE

DESCRIPTION

Size field: help text says data entry is optional. Data is required.

RECEIVED
NOV 02 1994
OSTI

FIXED
12/1/92
✓