DISTRIBUTION SHEET

To:
DistributionFrom:
R. T. AdamsDate:
08/01/94

Project Title/Work Order:

Liquid Effluent Monitoring Information System (LEMIS) - Test Plans Release 1.1

EDT No.: 141246

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	B1-18	X		
OSTI (2)	L8-07	X		

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

RECEIVED

SEP 30 1994

OSTI

DISCLAIMER

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

Sta #10
SEP - 8 1994

ENGINEERING DATA TRANSMITTAL

Page 1 of 1
1. EDT No 141246

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
11. Receiver Remarks:		12. Major Assm. Dwg. No.: N/A
		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 Trans- mittal	Orig- inator Dispo- sition	Receiv- er Dispo- sition
1	WHC-SD-WM-TP-164		0	Liquid Effluent Monitoring Information System (LEMIS) - Test Plans Release 1.1	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. R. T. Adams <i>R. T. Adams</i> 8/25/94 Signature of EDT Originator	19. L. J. Lang <i>L. J. Lang</i> 8-11-94 Authorized Representative Date for Receiving Organization	20. D. A. Guettler <i>D. A. Guettler</i> 8/25/94 Cognizant/Project Engineer's Manager	21. DOE APPROVAL (if required) Ltr. No. ☐ Approved ☐ Approved w/comments ☐ Disapproved w/comments
--	--	---	--

BD-7400-172-2 (07/91) GEF097

BD-7400-172-1 (07/91)

RELEASE AUTHORIZATION

**Document
Number:**

WHC-SD-WM-TP-164 Rev. 0

**Document
Title:**

Liquid Effluent Monitoring Information System
(LEMIS) Test Plans Release 1.1

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)

SUPPORTING DOCUMENT		1. Total Pages 390
2. Title Liquid Effluent Monitoring Information System (LEMIS) Test Plans Release 1.1	3. Number WHC-SD-WM-TP-164	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 <p>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.</p> <p>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.</p>		
<p>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.</p> <p>PATENT STATUS - This document is not to be transmitted in advance of patent clearance. It 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 use has been secured, upon request, from the Patent Counsel, U.S. Department of Energy Field Office, Richland, WA.</p> <p>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.</p>		<p>10. RELEASE STAMP</p> <div data-bbox="1090 1244 1503 1468"> <p>OFFICIAL RELEASE 23</p> <p>BY WHC</p> <p>DATE SEP - 8 1994</p> <p>Sto #10</p> </div>
9. Impact Level N/A		

**APPROVED FOR
PUBLIC RELEASE**

CW
8-30-94

This page intentionally left blank.

CONTENTS

1.0	LEMIS		1
1.1	LE0023U1	MAINTAIN SAMPLE PURPOSE CODES	1
1.2	LE2120U1	SAMPLE AUTHORIZATION FORM	11
1.3	LE2122U1	UPDATE SAMPLE SCHEDULE	98
1.4	LE3310U1	FIELD DATA DEFINITION	107
2.0	GENERIC		129
2.1	GE1001U1	MAINTAIN ADDRESS CODES	129
2.2	GE1002U1	MAINTAIN COUNTRY CODES	141
2.3	GE1003U1	MAINTAIN STATE CODES	151
2.4	GE1110U1	SECURITY PERSON MAINTENANCE	162
2.5	GE1210U1	MAINTAIN ORAGANIZATION	189
2.6	GE3002U1	MAINTAIN TABLE TYPE CODES	244
2.7	GE3003U1	MAINTAIN TEST TYPES	256
2.8	GE3004U1	VALID TEST TYPES FOR TABLE	262
2.9	GE3005U1	TYPE DATA VALID CODE	274
2.10	GE3007U1	MAINTAIN COMMENT STATUS CODES	290
3.0	LIBRARY		303
3.1	LI1002U1	MAINTAIN SITE CODE TABLE	303
3.2	LI1003U1	MAINTAIN PRECIPITATION CODES	314
3.3	LI1005U1	MAINTAIN LOCATION LEVEL CODE	326
3.4	LI1010U1	MAINTAIN LIBRARY STATUS CODE	337
3.5	LI4041U1	LOCATION MAINTENANCE	352

This page intentionally left blank.

LIQUID EFFLUENT MONITORING INFORMATION SYSTEM (LEMIS) TEST PLANS RELEASE 1.1

1.0 LEMIS

1.1 LE0023U1 MAINTAIN SAMPLE PURPOSE CODES

LEMIS Module Test Checklist Report

Name: MAINTAIN SAMPLE PURPOSE CODES
Short Name: LE0023U1
Purpose: To maintain Sample Purpose Codes

Initial Test Date: 12/16/92 *jc*
Verification Date: 12/16/92 *R*

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

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

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

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

Module Description

Module Help

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

To store a new sample purpose code:

Module Help

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

If all of the spaces are filled and more sample purpose 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 purpose 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 purpose 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

Module Test Case

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

Block Description

This is a code table supporting validation of the sample purpose code.

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.

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

Block Test Results Expected

- (✓) 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.

Field Name: SAMPLE_PURPOSE

Field Description

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

Field Name: SAMPLE_PURPOSE

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.

not null 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: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

Field Name: SAMPLE_PURPOSE

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
- NA () Store a series of records using the SHIFT/^{FC}~~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.
- () Use SQL*Plus to verify all data fields are processed correctly.
- NA () 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.~~
- (✓) 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

not null field

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

Field Name: SAMPLE_PURPOSE

Field Test Results Expected

attempt to TAB to the next field. An error should result.

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

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.
 - ☒ () 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.

not null 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

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

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
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- (/) Store a series of records using the SHIFT/⁵⁶ [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.~~
- (/) 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

not null field

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

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.

1.2 LE2120U1 SAMPLE AUTHORIZATION FORM

LEMIS Module Test Checklist Report

Name: SAMPLE AUTHORIZATION FORM
 Short Name: LE2120U1
 Purpose: Sample Authorization Form (SAF)

Initial Test Date: RJA 12/22/92

Verification Date: RJA 12/23/92

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

Block: EVENT_ORGAN_J
 Title: Customer Organization

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

		Dis		Val/	Not		
Data Element	Datatype	Len	Help	LOV	Range	Null	Ins
-----	-----	---	---	---	---	---	---

Block: EVENT_PERSON
 Title: Persons Associated With SAF

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

		Dis		Val/	Not		
Data Element	Datatype	Len	Help	LOV	Range	Null	Ins
-----	-----	---	---	---	---	---	---
RELATIONSHIP	CHAR	3	()	()	()	()	()
NAME_PERSON	CHAR	20	()	()	()	()	()
ADDR_PHONE	CHAR	13	()	()	()	()	()
ADDR_MS	CHAR	7	()	()	()	()	()

Block: FIELD_DATA_LOCATION
Title: Sampling Locations

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

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	()	()	()	()	()

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
-----	-----	---	---	---	---	---	---
TITLE_LOCATION	CHAR	40	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

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	22	()	()	()	()	()

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	15	()	()	()	()	()
HOLDING_TIME	CHAR	4	()	()	()	()	()
EXTRACTION_TIME	NUMBER	4	()	()	()	()	()

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: SAMPLE

Title: Sample

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_SAMPLE	CHAR	25	()	()	()	()	()
SAMPLE_PURPOSE	CHAR	2	()	()	()	()	()
IDENT_HEIS_NUM	CHAR	12	()	()	()	()	()

Block: SAMPLE_ANALYSIS_REQ_LINE

Title: Sample Analysis Request Line Item

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_SAMPANAL	NUMBER	3	()	()	()	()	()
COUNT_BOTTLES	NUMBER	2	()	()	()	()	()
TYPE_BOTTLE_COMPOSITIO	CHAR	2	()	()	()	()	()
SIZE_BOTTLE	NUMBER	4	()	()	()	()	()
PRESERVATIVE	CHAR	8	()	()	()	()	()

Block: SAMPLE_AUTHORIZATION

Title: Sample Authorization

	Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
	()	()	()	()	()	()	()	()
Data Element	Datatype	Dis	Len	Help	LOV	Val/	Not	Ins
						Range	Null	
IDENT SAF	CHAR	20	()	()	()	()	()	()
DATE_REQUESTED	DATE	8	()	()	()	()	()	()
PRIORITY	CHAR	1	()	()	()	()	()	()
TYPE_SAMP	CHAR	1	()	()	()	()	()	()
TYPE_PROTOCOL	CHAR	2	()	()	()	()	()	()
DESCR_PROTOCOL	CHAR	27	()	()	()	()	()	()
TYPE_SAMPLE_MATRIX	CHAR	2	()	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()	()

Block: SAMPLE_SCHEDULE

Title: Sample Schedule

	Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
	()	()	()	()	()	()	()	()
Data Element	Datatype	Dis	Len	Help	LOV	Val/	Not	Ins
						Range	Null	
IDENT_SAP	CHAR	20	()	()	()	()	()	()
TITLE_SAP	CHAR	33	()	()	()	()	()	()
IDENT_SCHED	CHAR	20	()	()	()	()	()	()
DATE_SCHEDULED	DATE	8	()	()	()	()	()	()

Block: TYPE_SAMPLE_MATRIX_CODE

Title: Sample Matrix

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

Block: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

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

Block: VALID_VALUES
Title: For PRIORITY

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

Block: VALID_VALUES
Title: for TYPE_PROTOCOL

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

Block: VALID_VALUES
Title: For TYPE_SAMP

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

The transaction maintains a SAF.

The SAF is the authorization to take a sample. As such it is based on a SAP and is subordinate to a SAP.

Text may be associated with a SAF.

Module Description

Multiple SAFs may exist for a SAP but only one may be active at a time. It will be normal to schedule future sampling as far as two or three years in the future.

Since a SAP may change between the time a sampling is scheduled and the time it occurs, scheduling a sampling will not create an entire SAF but only an abbreviated subset. (This will be a separate screen.)

At the point where a SAF is generated, it will inherit the previously scheduled item in the database. (The user will have looked at it as a sampling schedule not as a SAF but under the covers we will have created a SAF.)

OSM will enter their SAF identifier over the sample schedule ident and will complete the building of the SAF, which will include:

- 1) Pulling the methods list from the SAP. All methods will be copied directly from the SAP (another screen will be defined to allow a subset of the methods list to be pulled).
- 2) The stream locations will be copied from the SAP. Only the location title will be displayed.
- 3) The customer person information will be copied from the SAP as a default; it may be overridden by the operator. Customer person information includes: Customer name, phone, MSIN, and org code. The name of the customer organization is extracted from the organization data and displayed. The date requested is also displayed.

The following SAF specific elements will be included on the screen:

- a) Sample type (default to 'P' (protocol))
 - b) Priority (default 'T' (TPA Ranking))
 - c) Protocol (default 'R' (RCRA))
 - d) Other protocols - literal value
 - e) Sample Matrix (default 'WA' (water))
- 4) Organization information may be entered: Use LOV link to organization list. Multiple organizations may be associated with specific relationship codes.
 - 5) For each location, a sample set will be created. This set will include the sample number, using a generated number (that will be changed to the HEIS sample number in the field). The sample data line will include the purpose code (and expanded title), lab selected, default to the first lab identified with the method(s) be used.
 - 6) For each sample set, a series of method line items will be generated. Each line item will include the method, preservative, bottle quantity/type/size, extraction time in days, and holding time in days. The operator will have the option to combine methods into a

Module Description

bottle series if desired. [[Desirably, be able to automatically identify these possible combinations and provide the operator with a default that may be changed.]]

7) A SAF will have two distinct stages, development and approved. A flag will be kept to distinguish the stage. Once approved, a SAF cannot be modified unless the revision level is changed. This screen, therefore, will have to check conditions. If the SAF is approved, changes will not be allowed unless the revision level is also changed.

The subject of a sampling activity defaults to liquid effluent (LEF).

Round number remains in the database for HEIS compatibility but is not displayed or updated.

Module Help

This screen is used enter the "Sample Authorization Form" (SAF) information. A SAF is the authorization to take samples. It is subordinate to a SAP and must be related to a SAP. A SAF may only be created if a sample has been previously scheduled.

To create a SAF:

1. Select the SAMPLE SCHEDULE for which to create the SAF:
 - a) Press <F12> to enter query mode. (If "Enter a query" appears at the bottom of the screen, F12 does not need to be pressed)
 - b) A Sample Schedule may be selected by any of the four fields: SAP ID, SAP Title, Schedule ID, and Scheduled Date. To select by one of these fields, enter the desired value (ie SAP ID) into the appropriate field.
 - c) If multiple records match the entered value (ie multiple scheduled samples for a SAP) they will be displayed in date order with the earliest one first. Use the arrow keys to display the scheduled sample that this SAF will be for.
2. Enter the "Sample Authorization" information:
 - a) Press <PageDown> to enter the "Sample Authorization" block.
 - b) Enter the SAF ID. If the SAF ID is not known, or should be defaulted to the value of the schedule ID, type '.' and press <Enter>. This will copy the schedule ID into the SAF ID.
 - c) Enter the Date Requested in DD/MM/YY format. This is the date that Mobil Sample Labs is being requested to actually perform the sampling activity.
 - d) Enter "Priority", "Sample Type", "Protocol", and "Matrix". See the field help for these fields for a description of each.
 - e) If needed, text may be entered for this SAF by pressing <F7>.

Module Help

3. Enter/Update the Persons Associated with the SAF:

- a) Press <PageDown> to enter the "Persons..." block. If the customer name and/or the HASM coordinator were entered when this schedule was entered, the information will be displayed here. If the information is correct, skip over this block by pressing <PageDown> and continue this procedure with step 4. If this information needs to be entered or modified continue this procedure with the next step.
- b) If a customer contact, or HASM Coordinator has to be entered, use the arrow keys to move the cursor to an empty record. If one of these records needs to be modified, move the cursor to the record to be modified.
- c) If this is a new entry, enter "CUS" (for customer) or "HAS" (for HASM Coordinator) into the relationship field. LOV is available. Only one customer and one HASM Coordinator may be defined for each SAF. "HAS" has been defined as the default for this field. Press <Enter> to move to the next field.
- f) Enter the name of the person and press <Enter>.
- g) Enter the phone number and press <Enter>.
- h) Enter the MSIN and press <Enter>.
- i) Press <F10> to save this information in the database.

4. Enter Customer Organization:

- a) Press <PageDown> to enter the "Customer Organization" block. If a customer organization was entered when the schedule was created, there will already be information in this block.
- b) Enter the org code of the customer organization or use LOV to see a list of organizations. Only one customer organization may be defined for a SAF.

5. Enter the Locations for this SAF:

- a) Press <PageDown> to enter the "Sampling Locations" block.
- b) All of the locations defined in the SAP should be displayed in this block. If some of these locations are not to be used in this SAF, they may be deleted by moving the cursor to the location to be deleted and pressing <Delete>. This will not delete the location, only remove it's connection to this SAF.
- c) If a location needs to be added to the SAF, use the arrow keys to move the cursor to a blank record (or press <Insert>) Then use <F9> (LOV) to select a valid location for this SAF. Only locations defined in the SAP will be displayed in the LOV.

6. Enter the samples for a location.

- a) Use the arrow keys to move the cursor to the desired sampling location.
- b) Press <PageDown> to move into the "Sample" block.
- c) Enter the unique sample identifier. Enter "." to generate a sample ID from the SAF ID. (This will be the SAF ID followed by "-" and a sequence number) Press <Enter> to move to the next field.
- d) Enter the purpose for this sample. Example purposes are:

Module Help

PRIMARY, SPLIT, BLANK etc. LOV is available for this field.
Press <Enter> to move to the next field.

- e) Enter the ID of the lab that will perform the testing on this sample. (ie WESTON, DATACH, etc). LOV is available for this field.
- f) If more samples are to be entered, press the down arrow key or the <Insert> key to move to an empty record and repeat steps 6c) through 6e).
- g) Enter the "Sample Request Line Items" for every sample (as detailed in section 7).

7. Enter Sample Request Line Item.

- a) Press <PageDown> to move to the "Sample Analysis Request Line Item" block.
- b) Enter the "Request Number". This field will determine which analyses use the same set of bottles. If this field is left blank, the system will automatically fill it with the next number in sequence for this sample. See the next section: "Linking Request Lines" for more information about this field.
- c) Enter the "Analysis Class" ID. LOV is available. If there is only one method for an analysis class, then that method will be defaulted into the method field.
- d) Enter the method. LOV is available. Only valid methods for the selected analysis class will be allowed. Also only methods that have been selected for this SAF's SAP will be allowed.
- e) When the method has been selected, the bottle qty, type, size, and preservative will be defaulted from information defined for the selected method and the selected laboratory. These values may be overridden.
- f) Enter/modify the bottle quantity.
- g) Enter/modify the bottle type. LOV is available.
- h) Enter/modify the bottle size. LOV is available.
- i) Enter/modify the preservative. LOV is available.
- j) Press <F10> to save the new record to the database.
- k) Repeat steps 7b)->7j) for every method to be performed on this sample.

Linking Request Lines:

When more than one method is to be performed using the same bottle or set of bottles, the request lines should be merged using one of the following procedures:

Initial Entry Procedure:

If it is known that a new entry (new record) will be using the same set of bottles, the following procedure may be used:

- 1. Enter the same "Req Num" as the other request line(s) that will

Module Help

- use the same set of bottles.
2. Select "Analysis Class" and "Method" as above.
 3. Make sure the bottle qty, type, size and preservative match the other records with the same "Req Num". If they do not match the system will not save the record to the database. Instead an error message will be displayed.

Update procedure:

- If there are two existing records that should be combined into the same bottle or set of bottles follow this procedure:
1. Move the cursor to the record to be linked to another record.
 2. Make sure the bottle qty, type, size, and preservative match the record to be linked to.
 3. Enter the "Req Num" of the record to link to in the "Req Num" field. If the bottle information does not match an error message to this effect will be displayed.
 4. Press <F10> to commit the 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 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.

Module Test Plan

- () 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 12/23/92
- () 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

Module Test Plan

- () 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 12/16/92
- () 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 12/16/92

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. (N/A)
- () 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. (N/A)

- () The insert key clears all blocks of the entry screen. When the block supports multiple rows, all fields of the row pointed to by the cursor are cleared.
- () 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/A

- () 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 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

7489
QUERY
SORT

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.

N/A

- () 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.~~

N/A

() Delete an existing record.

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

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_ORGAN_J
Title: Customer Organization

Block Description

Block Test Case

Block Test Results Expected

- (✓) Display block help. Verify that the narrative adequately describes the purpose and functions of the block in end-user terms.
 - (✓) Block help describes:
 - (✓) Purpose of the block
 - (✓) Options available in the block
 - (✓) The relationship of the block to the transaction or to the principal parent block of the transaction
 - (✓) Navigation to other blocks of the transaction
- (✓) Store a series of record using the F6 [duplicate record] key to carry forward field values from the previously stored record.
 - (✓) Attempting to store the duplicated record without change should result in an error.
 - (✓) Change key fields and store the record. It should store successfully.
- () Delete a record from the block. If the record has dependent records, the delete should fail unless the help text indicates that a cascading delete is permitted.
 - (✓) Where block synchronization is used, ensure that the block data is in sync with the parent block.
- () Query a single record.
 - () A single record is retrieved.
 - () Pressing the F12 key clears the record and permits entering a second query.
 - () The second query successfully retrieves the desired data.

Block: EVENT_ORGAN_J
Title: Customer Organization

Block Test Results Expected

- N/A
- () 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.
 - () A query using a prefix % wildcard retrieves the correct data.
 - () A query using both prefix and suffix % wildcards retrieves the correct data.
 - (✓) Verify that the data queried in the block are ordered logically.
 - () Verify that multiple organization records can not be created.
 - (✓) Create record should not open a new record.
 - () Delete record should not work.
 - (✓) Clear record should not work.
 - (✓) Enter query should not work.
 - (✓) Count query should not work.
 - (✓) The down arrow key should not open a new record.

YES IT SHOULD
SEE #752

Block: EVENT_PERSON
Title: Persons Associated With SAF

Block Description

Block Test Case

Block Test Results Expected

- (✓) Display block help. Verify that the narrative adequately describes the purpose and functions of the block in end-user terms.
 - (✓) Block help describes:
 - () Purpose of the block
 - () Options available in the block
 - () The relationship of the block to the transaction or to the principal parent block of the transaction
 - () Navigation to other blocks of the transaction
- (✓) Store a series of record using the F6 [duplicate record] key to carry forward field values from the previously stored record.
 - (✓) Attempting to store the duplicated record without change should result in an error.
 - (✓) Change key fields and store the record. It should store successfully.
- (✓) Delete a record from the block. If the record has dependent records, the delete should fail unless the help text indicates that a cascading delete is permitted.
- (✓) Where block synchronization is used, ensure that the block data is in sync with the parent block.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing the F12 key 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.
 - (✓) A query using a prefix % wildcard retrieves the correct

Block: EVENT_PERSON
Title: Persons Associated With SAF

Block Test Results Expected

- (✓) data.
- (✓) A query using both prefix and suffix % wildcards retrieves the correct data.
- (✓) Verify that the data queried in the block are ordered logically.

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.
- N/ (✓) 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: EVENT_PERSON
Title: Persons Associated With SAF

Field Name: RELATIONSHIP

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).
 - () 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.

Block: EVENT PERSON

Title: Persons Associated With SAF

Field Name: RELATIONSHIP

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.~~
 CONVERTS TO UPPERCASE.
 - (✓) 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_PERSON

Title: Persons Associated With SAF

Field Name: NAME_PERSON

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: EVENT_PERSON
Title: Persons Associated With SAF

Field Name: NAME_PERSON

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/F4 [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.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~

Block: EVENT_PERSON
Title: Persons Associated With SAF

Field Name: ADDR_PHONE

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: EVENT_PERSON
Title: Persons Associated With SAF

Field Name: ADDR_PHONE

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/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.
- () 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: EVENT_PERSON
Title: Persons Associated With SAF

Field Name: ADDR_MS

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: EVENT_PERSON
Title: Persons Associated With SAF

Field Name: ADDR_MS

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.
- () 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: FIELD_DATA_LOCATION
Title: Sampling Locations

Block Description

Block Test Case

Block Test Results Expected

- () Display block help. Verify that the narrative adequately describes the purpose and functions of the block in end-user terms. 753
- () Block help describes:
 - () Purpose of the block
 - () Options available in the block
 - () The relationship of the block to the transaction or to the principal parent block of the transaction
 - () Navigation to other blocks of the transaction
- (✓) Store a series of record using the F6 [duplicate record] key to carry forward field values from the previously stored record.
 - (✓) Attempting to store the duplicated record without change should result in an error.
 - () Change key fields and store the record. It should store successfully.
- (✓) Delete a record from the block. If the record has dependent records, the delete should fail unless the help text indicates that a cascading delete is permitted.
- (✓) Where block synchronization is used, ensure that the block data is in sync with the parent block.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing the F12 key 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.
 - (✓) A query using a prefix % wildcard retrieves the correct

Block: FIELD DATA LOCATION
Title: Sampling Locations

Block Test Results Expected

data.

() A query using both prefix and suffix % wildcards retrieves the correct data.

(✓) Verify that the data queried in the block are ordered logically.

(✓) Verify that all of the location selected in the SAP were copied to the SAF when it was created.

(✓) Verify that textual data related to a location is displayable from this block.

OK.

Block: LABORATORY
Title: Laboratory Definition

Block Description

Block Test Case

Block Test Results Expected

Field Name: IDENT_LAB

Field Description

Corresponds to Lab Code in HEIS. Unique identifier for the laboratory.

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

Block: LABORATORY
Title: Laboratory Definition

Field Name: IDENT_LAB

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.
 - ☐ 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.

Block: LABORATORY
Title: Laboratory Definition

Field Name: IDENT_LAB

Field Test Results Expected

Block: LEMIS_LOCATION
Title: LEMIS Location

Block Description

Block Test Case

Block Test Results Expected

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

Block: LEMIS_LOCATION
 Title: LEMIS Location

Field Name: TITLE_LOCATION

Field Test Results Expected

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.
- ☐ 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: LEMIS_LOCATION
Title: LEMIS Location

Field Name: FLAG_TEXT

Field Description

Field Test Case

Field Test Results Expected

Block: METHOD
Title: Analysis Class

Block Description

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

Block: METHOD
 Title: Analysis Class

Field Name: IDENT_METHOD

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.
 - () 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.

Block: METHOD
Title: Analysis Class

Field Name: IDENT_METHOD

Field Test Results Expected

Block: METHOD
Title: Laboratory Methods

Block Description

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

N/A () Change the field in an existing record. Verify that the change

Block: METHOD
Title: Laboratory Methods

Field Name: IDENT_METHOD

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.
- N/A*) 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
- (✓) 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.
- () Any alphanumeric characters may be entered into the field.
- () Delete an existing value.
 - () For NOT NULL fields, the COM00101 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: Laboratory Methods

Field Name: IDENT_METHOD

Field Test Results Expected

Block: METHOD
Title: Laboratory Methods

Field Name: HOLDING_TIME

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. *Display only*
- 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* Begin field with a space.
- N/A*) 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.

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. *Display only*

Block: METHOD
Title: Laboratory 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 [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.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result.

Block: METHOD
Title: Laboratory 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: Laboratory 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/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.
- () Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
- () Begin field entry with a space. An error should occur.

Block: ORGANIZATION
Title: Organization

Block Description

Block Test Case

Block Test Results Expected

Field Name: IDENT_ORGAN

Field Description

Organization title abbreviation. Format is XXX/XXX (alphanumeric).

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.

- (751) () 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

Block: ORGANIZATION
Title: Organization

Field Name: IDENT_ORGAN

Field Test Results Expected

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.
- () Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result.
- (752) () Delete an existing value.
 - () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. (752)
- (✓) 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: ORGANIZATION
 Title: Organization

Field Name: TITLE

Field Description

Field Test Case

- N/A* *DISPLAY 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.
 - () 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: ORGANIZATION
Title: Organization

Field Name: TITLE

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.
- () 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.

Block: SAMPLE
Title: Sample

Block Description

Block Test Case

Block Test Results Expected

- (✓) Display block help. Verify that the narrative adequately describes the purpose and functions of the block in end-user terms.
 - (✓) Block help describes:
 - () Purpose of the block
 - () Options available in the block
 - () The relationship of the block to the transaction or to the principal parent block of the transaction
 - () Navigation to other blocks of the transaction
- (✓) Store a series of record using the F6 [duplicate record] key to carry forward field values from the previously stored record.
 - (✓) Attempting to store the duplicated record without change should result in an error.
 - (✓) Change key fields and store the record. It should store successfully.
- (✓) Delete a record from the block. If the record has dependent records, the delete should fail unless the help text indicates that a cascading delete is permitted.
- (✓) Where block synchronization is used, ensure that the block data is in sync with the parent block.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing the F12 key 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.
 - (✓) A query using a prefix % wildcard retrieves the correct

Block: SAMPLE
Title: Sample

Block Test Results Expected

data.

(1) A query using both prefix and suffix % wildcards retrieves the correct data.

(1) Verify that the data queried in the block are ordered logically.

(1) Verify that when "." is entered into the "Sample ID" field that the value is defaulted correctly.

Field Name: IDENT_SAMPLE

Field Description

Field Test Case

Field Test Results Expected

Block: SAMPLE
Title: Sample

Field Name: SAMPLE_PURPOSE

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: SAMPLE
Title: Sample

Field Name: SAMPLE_PURPOSE

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.
- () 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: SAMPLE
Title: Sample

Field Name: IDENT_HEIS_NUM

Field Description

Field Test Case

Field Test Results Expected

Block: SAMPLE_ANALYSIS_REQ_LINE
 Title: Sample Analysis Request Line Item

Block Description

Block Test Case

Block Test Results Expected

- (✓) Display block help. Verify that the narrative adequately describes the purpose and functions of the block in end-user terms.
 - (✓) Block help describes:
 - (✓) Purpose of the block
 - (✓) Options available in the block
 - (✓) The relationship of the block to the transaction or to the principal parent block of the transaction
 - (✓) Navigation to other blocks of the transaction
- (✓) Store a series of record using the F6 [duplicate record] key to carry forward field values from the previously stored record.
 - () Attempting to store the duplicated record without change should result in an error.
 - (✓) Change key fields and store the record. It should store successfully.
- (✓) Delete a record from the block. If the record has dependent records, the delete should fail unless the help text indicates that a cascading delete is permitted.
- () Where block synchronization is used, ensure that the block data is in sync with the parent block.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing the F12 key 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.
 - (✓) A query using a prefix % wildcard retrieves the correct

Block: SAMPLE_ANALYSIS_REQ_LINE
Title: Sample Analysis Request Line Item

Block Test Results Expected

- data.
- (✓) A query using both prefix and suffix % wildcards retrieves the correct data.
- (✓) Verify that the data queried in the block are ordered logically.
- (✓) Verify that all methods selected in the SAP are copied to the SAF.
- (✓) Verify that a new "Req Num" is generated when nothing is entered into this field. 759
- (✓) Verify that the correct qty, type, size, and preservative is defaulted into the current record after the method is entered. (This should be based on the method and the lab)
- (✓) Verify that the request lines are merged correctly.
 - (✓) Does not allow "Req Num" to be changed to a number with mis-matching bottle information.
 - (✓) Does not allow the bottle information for an existing record to be changed so that it does not match other records with the same "Req Num"

Field Name: IDENT_SAMPANAL

Field Description

Field Test Case

Field Test Results Expected

Field Name: COUNT_BOTTLES

Field Description

Field Test Case

Field Test Results Expected

Block: SAMPLE_ANALYSIS_REQ_LINE
Title: Sample Analysis Request Line Item

Field Name: TYPE_BOTTLE_COMPOSITION

Field Description

Field Test Case

Field Test Results Expected

Field Name: SIZE_BOTTLE

Field Description

Field Test Case

Field Test Results Expected

Field Name: PRESERVATIVE

Field Description

Field Test Case

Field Test Results Expected

Block: SAMPLE_AUTHORIZATION
Title: Sample Authorization

Block Description

Block Test Case

Block Test Results Expected

- (☒) Display block help. Verify that the narrative adequately describes the purpose and functions of the block in end-user terms.
- (☒) Block help describes:
 - (☒) Purpose of the block
 - (☒) Options available in the block
 - (☒) The relationship of the block to the transaction or to the principal parent block of the transaction
 - (☒) Navigation to other blocks of the transaction
- () Store a series of record using the F6 [duplicate record] key to carry forward field values from the previously stored record.
- (☒) Attempting to store the duplicated record without change should result in an error.
- () Change key fields and store the record. It should store successfully.
- (☒) Delete a record from the block. If the record has dependent records, the delete should fail unless the help text indicates that a cascading delete is permitted.
- (☒) Where block synchronization is used, ensure that the block data is in sync with the parent block.
- () Query a single record.
 - (☒) A single record is retrieved.
 - () Pressing the F12 key 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.
 - () A query using a prefix % wildcard retrieves the correct

Block: SAMPLE_AUTHORIZATION
 Title: Sample Authorization

Block Test Results Expected

data.

- () A query using both prefix and suffix % wildcards retrieves the correct data.

N/K Verify that the data queried in the block are ordered logically.

(✓) Verify that multiple SAFs may not be created for a schedule.

(✓) Create record should not create a new record.

(✓) Delete record should not delete the record.

(✓) Clear record should not work.

(✓) Enter query should not work.

(✓) Count query should not work.

(✓) The Down arrow key should not open a new record.

(✓) Verify the entering "." into the SAF ID field defaults the SAF ID correctly.

(✓) Verify Correct navigation/mandatory field when protocol is "OTHER"

(✓) If protocol is not other, cursor should skip "Specify" field in both directions.

(✓) If protocol is "OTHER" then cursor should move through the "Specify" field in both directions.

(✓) If protocol is "OTHER" the cursor should not be able to move through the "Specify" field without entering a value.

(✓) If protocol is changed from "OTHER" to another value, the value in the "Specify" field should be cleared.

(✓) If protocol is "OTHER" the record should not be able to be committed without a value in the "Specify" field.

O.K. (✓) Verify that the "Person" and "Organization" information is copied correctly from the SAP.

Field Name: IDENT_SAF

Field Description

Field Test Case

Field Test Results Expected

Block: SAMPLE_AUTHORIZATION
Title: Sample Authorization

Field Name: DATE_REQUESTED

Field Description

Field Test Case

Field Test Results Expected

Field Name: PRIORITY

Field Description

Field Test Case

Field Test Results Expected

Field Name: TYPE_SAMP

Field Description

Field Test Case

Field Test Results Expected

Field Name: TYPE_PROTOCOL

Field Description

Field Test Case

Field Test Results Expected

Block: SAMPLE_AUTHORIZATION
Title: Sample Authorization

Field Name: DESCR_PROTOCOL

Field Description

Field Test Case

Field Test Results Expected

Field Name: TYPE_SAMPLE_MATRIX

Field Description

Field Test Case

Field Test Results Expected

Block: SAMPLE_AUTHORIZATION
Title: Sample Authorization

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: SAMPLE AUTHORIZATION
Title: Sample Authorization

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
- () 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.

Block: SAMPLE_AUTHORIZATION
Title: Sample Authorization

Field Name: FLAG_TEXT

Field Test Results Expected

- () 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: SAMPLE_SCHEDULE
Title: Sample Schedule

Block Description

Block Test Case

Block Test Results Expected

- (✓) Display block help. Verify that the narrative adequately describes the purpose and functions of the block in end-user terms.
- (✓) Block help describes:
 - () Purpose of the block
 - () Options available in the block
 - () The relationship of the block to the transaction or to the principal parent block of the transaction
 - () Navigation to other blocks of the transaction
- (N/K) () Store a series of record using the F6 [duplicate record] key to carry forward field values from the previously stored record.
- () Attempting to store the duplicated record without change should result in an error.
- () Change key fields and store the record. It should store successfully.
- (N/K) () Delete a record from the block. If the record has dependent records, the delete should fail unless the help text indicates that a cascading delete is permitted.
- (N/K) () Where block synchronization is used, ensure that the block data is in sync with the parent block.
- (✓) Query a single record.
 - (✓) A single record is retrieved.
 - (✓) Pressing the F12 key 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.
 - (✓) A query using a prefix % wildcard retrieves the correct

Block: SAMPLE_SCHEDULE
Title: Sample Schedule

Block Test Results Expected

- data.
(✓) A query using both prefix and suffix % wildcards retrieves the correct data.
(✓) Verify that the data queried in the block are ordered logically.

Field Name: IDENT_SAP

Field Description

Field Test Case

Field Test Results Expected

Field Name: TITLE_SAP

Field Description

Field Test Case

Field Test Results Expected

Field Name: IDENT_SCHED

Field Description

Field Test Case

Field Test Results Expected

Block: SAMPLE_SCHEDULE
Title: Sample Schedule

Field Name: DATE_SCHEDULED

Field Description

Field Test Case

Field Test Results Expected

Block: TYPE SAMPLE MATRIX CODE
 Title: Sample Matrix

SAMP AUTHORIZATION

Block Description

Block Test Case

Block Test Results Expected

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.
- N/A Begin field with a space. NOT NULL
- () 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.
- (✓) 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: TYPE_SAMPLE_MATRIX_CODE
Title: Sample Matrix

Field Name: DESCRIPTION

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
- (✓) 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.
- () 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.

Block: TYPE_SAMPLE_MATRIX_CODE
Title: Sample Matrix

Field Name: DESCRIPTION

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.
- () 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_PURPOSE
Title: Sample Purpose Codes

Block Description

Block Test Case

Block Test Results Expected

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.
 - () 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: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

Field Name: DESCRIPTION

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.
- () 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: TYPE_SAMPLE_PURPOSE
Title: Sample Purpose Codes

Field Name: DESCRIPTION

Field Test Results Expected

- () 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: VALID_VALUES
Title: For PRIORITY

Block Description

Block Test Case

Block Test Results Expected

Field Name: DESCRIPTION

Field Description

Field Test Case

Field Test Results Expected

Block: VALID_VALUES
Title: for TYPE_PROTOCOL

Block Description

Block Test Case

Block Test Results Expected

Field Name: DESCRIPTION

Field Description

Field Test Case

Field Test Results Expected

Block: VALID_VALUES
Title: For TYPE_SAMP

Block Description

Block Test Case

Block Test Results Expected

Field Name: DESCRIPTION

Field Description

Field Test Case

Field Test Results Expected

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

Comment ID : 749 Priority: LOW Date: 22-DEC-92

Status : NOTPROB Date: 22-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TODD ADAMS

System ID : 92000009785 Oracle ROWID:

Last Error : -0

Last Msg : FRM-40350 Query caused no records to be retrieved.

Table : SAMPLE_SCHEDULE

Block : SS

Field : SS.IDENT_SAP

DESCRIPTION

Sort for query is wrong. Its not sorting by the Ident field.

CORRECTIVE ACTION

12/22/92 - Not a problem. The information is sorted by the scheduled date in ascending order, then by the ident. This way the HASM people can do an open query to find the next scheduled event.

✓ VERIFIED
RJA
12/23
✓

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

```
=====
Comment ID : 750    Priority: LOW                      Date: 22-DEC-92
=====
Status      : FIXED                      Date: 22-DEC-92      Class:
Type        : PROBLEM
Assigned To : KEN ATKINS
Entered By  : TODD ADAMS
System ID   : 92000009453                  Oracle ROWID: 0000052B.0009.0007
Last Error  : -0
Last Msg    : FRM-40350    Query caused no records to be retrieved.
Table       : FIELD_DATA_LOCATION
Block       : FLDDAT
Field       : FLDDAT.DSP_TITLE_LOCATION
=====
```

DESCRIPTION

Tried to edit text and got "No text available". Looks like
flag text sequencing problem.

CORRECTIVE ACTION

12/22/92 - This is a display only text for the location. This message
means that there is no text for the location. I will reword the
message to make it clearer.

VERIFIED
RJA
12/8
✓

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

```
=====
Comment ID : 751   Priority: LOW   Date: 22-DEC-92
=====
Status      : NOTPROB   Date: 22-DEC-92   Class:
Type        : PROBLEM
Assigned To : KEN ATKINS
Entered By  : TODD ADAMS
System ID   :                               Oracle ROWID: 00000503.000A.0007
Last Error  : -0
Last Msg    : FRM-40350 Query caused no records to be retrieved.
Table       : EVENT_ORGAN_J
Block       : EVEORG
Field       : EVEORG.DSP_IDENT_ORGAN
=====
```

DESCRIPTION

No field level help for Org Ident.

CORRECTIVE ACTION

12/22/92 - Since there is only one field, I did not put field level help in. Field level help requires a post-generation change for this one because the lookup table resides in a different application.

VERIFIED

12/23 RJA

✓

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

```
=====
Comment ID : 752      Priority: LOW      Date: 22-DEC-92
=====
Status      : FIXED      Date: 22-DEC-92      Class:
Type        : PROBLEM
Assigned To : KEN ATKINS
Entered By  : TODD ADAMS
System ID   :      Oracle ROWID: 00000503.000A.0007
Last Error  : -0
Last Msg    : FRM-40400 Transaction complete -- 1 records posted an
Table       : EVENT_ORGAN_J
Block       : EVEORG
Field       : EVEORG.DSP_IDENT_ORGAN
=====
```

DESCRIPTION

When trying to delete organization, I get error stating that there can only be one organization tied to the SAF. If Org is optional, I should be able to delete it.

CORRECTIVE ACTION

12/22/92 - Fixed.

VERIFIED

RJA

12/23

✓

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

```
=====
Comment ID : 753      Priority: LOW      Date: 22-DEC-92
=====
Status      : FIXED      Date: 23-DEC-92      Class:
Type        : PROBLEM
Assigned To:              KEN ATKINS
Entered By :              TODD ADAMS
System ID   : 92000009453      Oracle ROWID: 0000052B.0009.0007
Last Error  : -0
Last Msg    : FRM-40401 No changes to commit.
Table       : FIELD_DATA_LOCATION
Block       : FLDDAT
Field       : FLDDAT.DSP_TITLE_LOCATION
=====
```

DESCRIPTION

Block help does not adequately describe the purpose and function of the location block. It looks like its pulling the table help and using module block help might allow you to specifically describe its usage in this form.

CORRECTIVE ACTION

12/23/92 - Added module block help

VERIFIED
RJA
✓
12/23

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

=====

Comment ID : 754 Priority: LOW Date: 22-DEC-92

=====

Status : FIXED Date: 23-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TODD ADAMS

System ID : 92000009832 Oracle ROWID: 00000323.0007.0007

Last Error : -0

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

Table : SAMPLE

Block : SAMPLE

Field : SAMPLE.IDENT_SAMPLE

DESCRIPTION

I think there should be some mention of the HIES number field in the block level help letting the user know that they cannot access it in this form and why it is there.??

CORRECTIVE ACTION

12/23/92 - Added help to module block to describe purpose of HEIS number field

VERIFIED
12/23
RJA
✓

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

=====

Comment ID : 755 Priority: LOW Date: 22-DEC-92

=====

Status : FIXED Date: 23-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TODD ADAMS

System ID : 92000009820 Oracle ROWID: 00000463.0005.0007

Last Error : -0

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

Table : SAMPLE_AUTHORIZATION

Block : SAMAUT

Field : SAMAUT.DATE_REQUESTED

DESCRIPTION

Date format in field help is wrong (dd-mmm-yy).

CORRECTIVE ACTION

12/23/92 - Fixed

✓ EMER

12/23

✓ RJA

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

=====

Comment ID : 756 Priority: LOW Date: 22-DEC-92

=====

Status : FIXED Date: 22-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : KEN ATKINS

System ID : 92000009725

Oracle ROWID: 0000052B.0006.0007

Last Error : -0

Last Msg : FRM-40350 Query caused no records to be retrieved. .

Table : FIELD_DATA_LOCATION

Block : FLDDAT

Field : FLDDAT.DSP_TITLE_LOCATION

DESCRIPTION

Did not display locations. Gave duplicate error message when location was added.

CORRECTIVE ACTION

12/22/92 - Added code to force commit after entering the SAF information.

VERIFIED

RJA

12/23 ✓

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

Comment ID : 757 Priority: LOW Date: 22-DEC-92

Status : FIXED Date: 22-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : KEN ATKINS

System ID : 92000009793 Oracle ROWID: 00000323.0004.0007

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 2 records posted an

Table : SAMPLE

Block : SAMPLE

Field : SAMPLE.IDENT SAMPLE

DESCRIPTION

Did not add sequence number when creating first sample using ".".

CORRECTIVE ACTION

12/22/92 - Fixed.

VERIFIED

ROA

12/23

✓

✓

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

Comment ID : 765 Priority: LOW Date: 23-DEC-92

Status : FIXED Date: 23-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : KEN ATKINS

System ID : 92000009942

Oracle ROWID: 00000324.0006.0007

Last Error : -0

Last Msg : FRM-40407 Transaction complete -- posted records comm

Table : SAMPLE

Block : SAMPLE

Field : SAMPLE.IDENT_SAMPLE

DESCRIPTION

Sample ID Sequence not generated correctly. Does not find maximum sample ID and add one.

CORRECTIVE ACTION

12/23/92 - Fixed sample id generating procedure to post changes to db and then query db to get maximum sequence.

VERIFIED

RDA

✓
12/23

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

=====

Comment ID : 758 Priority: LOW Date: 23-DEC-92

=====

Status : FIXED Date: 23-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TODD ADAMS

System ID : 92000009871 Oracle ROWID: 0000032D.0015.0007

Last Error : -0

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

Table : SAMPLE_ANALYSIS_REQ_LINE

Block : REQLIN

Field : REQLIN.TYPE_BOTTLE_COMPOS
ITION

DESCRIPTION

Help says this field is mandatory, hint says OPTIONAL.

CORRECTIVE ACTION

12/23/92 - Removed "OPTIONAL" from this and other bottle fields.

VERIFIED

12/23/

PDA

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

Comment ID : 760 Priority: LOW Date: 23-DEC-92

Status : FIXED Date: 23-DEC-92 Class:
Type : PROBLEM
Assigned To: KEN ATKINS
Entered By : TODD ADAMS
System ID : 92000009871 Oracle ROWID: 0000032D.0010.0007
Last Error : -0
Last Msg : FRM-40350 Query caused no records to be retrieved.
Table : SAMPLE_ANALYSIS_REQ_LINE
Block : REQLIN
Field : REQLIN.IDENT_SAMPANAL

DESCRIPTION

Request number sequence generating problem. I added 3 records then committed. I then added 6 more and committed. I then added a record by tabbing through the request # field, expecting to get #10 and instead got #6.

CORRECTIVE ACTION

12/23/92 - Fixed sequence number generate procedure to post changes and then query max from db.

VERIFIED

RJA

12/23

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2120U1 Sample Authorization Form

```
=====
Comment ID : 765      Priority: LOW      Date: 23-DEC-92
=====
Status      : ASSIGNED      Date: 23-DEC-92      Class:
Type        : PROBLEM
Assigned To:              KEN ATKINS
Entered By :              KEN ATKINS
System ID   : 92000009942      Oracle ROWID: 00000324.0006.0007
Last Error  : -0
Last Msg    : FRM-40407      Transaction complete -- posted records comm
Table       : SAMPLE
Block       : SAMPLE
Field       : SAMPLE.IDENT_SAMPLE
=====
```

DESCRIPTION

```
-----
Sample ID Sequence not generated correctly. Does not find maximum
sample ID and add one.
```

VERIFIED
N/A
✓
12/23

Function Key Test Checklist for form: LE212041

			Blocks <u>SAF</u>			
			SAMPLE SCHEDULE 1	SAMPLE AUTH 2	PERSON 3	ORG 4
Form Func	Function Key					
	Show Vers	Shift-F10	☒	☒	☒	☒
	Comment/Bug	Shift-F3	☒	☒	☒	☒
	Fast Access	F5	☒	☒	☒	☒
	Back	F4 or Esc	☒	☒	☒	☒
	Main Menu	Shift-F4	☒	☒	☒	☒
	Key Tmpl	Ctrl-F1	☒	☒	☒	☒
	Contxt Prnt	F11	☒	☒	☒	☒
	Query Print	Ctrl-F11	☒	☒	☒	☒
	Ora Print	Shift-F11	☒	☒	☒	☒
Block Func	Help	F1	☒	☒ 755	☒	☒ 751
	Key Help	Shift-F1	☒	☒	☒	☒
	Online Doc	F2	☒	☒	☒	☒
	Dis Journal	Shift-F2	☒	☒	☒	☒
	Rel Data	F3	☒	☒	☒	☒
	Spec Func 1	Shift-F5	☒	☒	☒	☒
	Spec Func 2	Ctrl-F5	☒	☒	☒	☒
	Edit Text	F7	☒	☒	☒	☒
	Exit Sys	Ctrl-F4	☒	☒	☒	☒
	Enter Qry	F12	☒	☒	☒	☒
Query	Correct Query fields		☒	☒	☒	☒
	Execute Qry	Shift-F12	☒	☒	☒	☒
	Count Qry	Ctrl-F12	☒	☒	☒	☒
	Uppercase? & LOVs?		☒	☒	☒	☒
	Query Order		☒ 749	☒	☒	☒
Navigate	Home	Home	☒	☒	☒	☒
	End	End	☒	☒	☒	☒
	Nxt/Prv Blk	PgDn/PgUp	☒	☒	☒	☒
	Scrl Up/Dn	Shift-Up/Down	☒	☒	☒	☒
	Prev/Next Fld		☒	☒	☒	☒
Insert/Validate	Up/Down		☒	☒	☒	☒
	Insert Rec	Insert	☒	☒	☒	☒
	Add Record at end		☒	☒	☒	☒
	Mandatory Check		☒	☒	☒	☒
	Dupl Fld	Shift-F6	☒	☒	☒	☒
	Dupl Rec	F6	☒	☒	☒	☒
	Duplicate Check		☒	☒	☒	☒
	Validation		☒	☒	☒	☒
	List Val	F9	☒	☒	☒	☒
	Optional Fields Marked		☒	☒	☒	☒
Del/Clr	Uppercase		☒	☒	☒	☒
	Commit	F10	☒	☒	☒	☒
	Delete Rec	Delete	☒	☒	☒	☒ 752
	Clear Fld	F8	☒	☒	☒	☒
	Clear Frm	Shift-F8	☒	☒	☒	☒
	Clear Rec	Ctrl-F8	☒	☒	☒	☒

Bugs:

- 749 1) QUERY ORDER - TOP BLOCK
- 751 2) NO FIELD LEVEL HELP FOR ORG IDENT
- 752 3) WONT ALLOW DELETE
- 75 4) DATE FORMAT IS WRONG IN FIELD HELP FOR DATE REQUESTED.
- 5)
- 6)
- 7)
- 8)
- 9)
- 10)

Fixed Tested

☒	☒
☒	☒
☒	☒
☒	☒
☒	☒
☐	☐
☐	☐
☐	☐
☐	☐
☐	☐

Bugs:		Fixed	Tested
750	1) NO TEXT AVAILABLE WHEN I TRIED TO EDIT TEXT.	☒	☒
753	2) INADEQUATE BLOCK HELP	☒	☒
754	3) MENTION HEIS # IN BLOCK HELP.	☒	☒
58	4) HELP SAYS <div style="border: 1px solid black; padding: 5px; display: inline-block;">BOTTLE QTY SIZE PRESGLU</div> TYPE IS MANDATORY, HINT SAYS OPTIONAL	☐	☒
	5)	☐	☐
	6)	☐	☐
	7)	☐	☐
	8)	☐	☐
	9)	☐	☐
759	10) REQUEST IF SEQUENCE PROBLEM	☒	☒
760			

1.3 LE2122U1 UPDATE SAMPLE SCHEDULE

LEMIS Module Test Checklist Report

Name: UPDATE SAMPLE SCHEDULE
 Short Name: LE2122U1
 Purpose: To view the sample schedule for a SAP.

Initial Test Date: 12/18/92 *fe*

Verification Date: 12/18/92 *fe*

	Help	Disp Fkeys	Security			
			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	35	()	()	()	()	()
REV_EVENT	NUMBER	3	()	()	()	()	()

Block: SAMPLE_SCHEDULE
Title: Sample Schedule

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_SCHED	CHAR	20	()	()	()	()	()
DATE_SCHEDULED	DATE	8	()	()	()	()	()
IDENT_SAF	CHAR	20	()	()	()	()	()
DATE_REQUESTED	DATE	8	()	()	()	()	()

Module Description

Module Help

Module Test Plan

Module Test Case

Block: EVENT
Title: SAP

Block Description

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.

Block: EVENT
Title: SAP

Field Name: IDENT_EVENT

Field Test Case

- () 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 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.

Block: EVENT
Title: SAP

Field Name: IDENT_EVENT

Field Test Results Expected

- () 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/F7 [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: SAMPLE_SCHEDULE
Title: Sample Schedule

Block Description

Block Test Case

Block Test Results Expected

Field Name: IDENT_SCHED

Field Description

Field Test Case

Field Test Results Expected

Field Name: DATE_SCHEDULED

Field Description

Field Test Case

Field Test Results Expected

Field Name: IDENT_SAF

Field Description

Field Test Case

Field Test Results Expected

745 no help

Block: SAMPLE_SCHEDULE
Title: Sample Schedule

Field Name: DATE_REQUESTED

Field Description

Field Test Case

Field Test Results Expected

Programmer's Bug Report

Application: LEMIS Liquid Effluent Monitoring Information System

Transaction: LE2122U1 Update Sample Schedule

=====

Comment ID : 745 Priority: LOW Date: 18-DEC-92

=====

Status : FIXED Date: 18-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40100 At first record.

Table : SAMPLE_SCHEDULE

Block : SS

Field : SS.DATE_SCHEDULED

DESCRIPTION

no help available

CORRECTIVE ACTION

12/18/92 - Help added.

1.4 LE3310U1 FIELD DATA DEFINITION

LEMIS Module Test Checklist Report

Name: FIELD DATA DEFINITION
 Short Name: LE3310U1
 Purpose: Define field data for a sample
 Initial Test Date: 12/17/92 HC
 Verification Date: 12/18/92 K

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

Block: FIELD_DATA_LOCATION
 Title: Field Data for a Location

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

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
DATE_COLLECT_START	DATE	8	()	()	()	()	()
IDENT_CHAIN_CUSTODY	CHAR	7	()	()	()	()	()
TITLE_LOGBOOK	CHAR	25	()	()	()	()	()
NUM_LOGBOOK	CHAR	8	()	()	()	()	()
NUM_LOG_PAGE	NUMBER	4	()	()	()	()	()
SAMPLE_CLARITY	CHAR	6	()	()	()	()	()
SAMPLE_COLOR	CHAR	9	()	()	()	()	()
SAMPLE_TEMPERATURE	NUMBER	6	()	()	()	()	()
CONDUCTIVITY	CHAR	6	()	()	()	()	()
PH	NUMBER	4	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

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	()	()	()	()	()

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
TITLE_LOCATION	CHAR	40	()	()	()	()	()

Block: SAMPLE
Title: Sample

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_SAMPLE	CHAR	25	()	()	()	()	()
IDENT_HEIS_NUM	CHAR	12	()	()	()	()	()
SAMPLE_PURPOSE	CHAR	2	()	()	()	()	()

Block: SAMPLE_SCHEDULE
Title: Sample Authorization

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry
()	()	()	()	()	()	()	()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_SAP	CHAR	20	()	()	()	()	()
TITLE_SAP	CHAR	35	()	()	()	()	()
IDENT_SAF	CHAR	20	()	()	()	()	()
DATE_REQUESTED	DATE	8	()	()	()	()	()

Block: TYPE SAMPLE PURPOSE
Title: Sample Purpose Codes

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

Module Description

This transaction creates the field data stub for a sample.

1. Link to a sample.
2. Create the stub information.
3. Optionally, link associated person(s).
4. Optionally, link associated resource(s).
5. Optionally, link to container(s).

Field data may contain text. ATM (atmospheric text) is a valid text code.

The sampling location(s) associated with field data are defined. Use only the location title.

For each location:

1. Identify person taking sample. Use literal values. Multiple persons can be associated with the taking of a sample.
2. Identify the scheduled sampling date. This should default to the scheduled sampling date of the SAF.
3. The date and time of the collection begin will be identified as the "collection date." The date and time collection ended will be retained in the database but not displayed.
4. The atmospheric, geographic, and flow rate data will not be displayed.

For each location, individual sample information may be entered:

1. Sample ID. The generated sample identifier is replaced by the HEIS sample number.
2. The sample purpose code and expanded name are displayed.
3. The selected lab for the sample is displayed.
4. QC type and expanded name and turnaround type code and expanded

Module Description

text are displayed.

5. Subject defaults to "LEF"; round number defaults to 1. Neither are displayed.

Basic field data information is entered:

- 1) Logbook number
- 2) Logbook page number.
- 3) Conductivity.
- 4) pH
- 5) Sample clarity
- 6) Sample color
- 7) Temperature of sample
- 8) Chain of custody number.

Module Help

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.

Module Test Plan

- () 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 12/17/92
- () 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 Jo Date Complete 12/18/92
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution 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.

Module Test Plan

- () 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 Le Date Complete 12/18/92
- () 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 RA Date Complete 12/18/92

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.
- () 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: FIELD_DATA_LOCATION

Title: Field Data for a Location

Block Description

This table is the primary link between the sample and the field data information that is recorded when the sample is taken.

Field data information is recorded for a sample at the same location for all portions of the sample. The location is normally some point on a waste stream. Grouping the location with the field data reduces redundant data entry of field data for each sample item.

In addition to the data items entered during scheduling, the following data is entered during field sampling, and is mandatory to maintain compatibility with HEIS:

Sample id (HEIS number)
Actual sampling date and time

It is strongly suggested that the following items be entered into LEMIS, to ensure data integrity:

Chain of Custody ID
Log book title
Field note page numbers.

It is suggested that the following items be entered by the field techs after a sample has been taken, to ensure data integrity:

Temperature
Wind Velocity
Ph
Conductivity
Weather (precipitation)
Sample Clarity
Sample Color

Preparing to take a sample

Sampling and Mobile Labs would benefit from the same reports that were mentioned previously as desirable for scheduling. These are the schedule summary report and individual sample schedule report.

The schedule summary report will be beneficial to Sampling and

Block: FIELD_DATA_LOCATION
Title: Field Data for a Location

Block Description

Mobile Labs when they are either allocating their resources, or preparing for sampling the next day. The report should be organized by date, and at least contain requested schedule date, location, and the SAF #. This report could be a screen that allows the field tech to search through a list of scheduled samples, by requested date. This report may not be available until release two.

There is also a need for a report screen that shows the requested sample date, location, SAF #, field contact, data owner, sample matrix (always liquid effluent), text? and all the bottle information necessary to prepare for taking the sample. It may prove undesirable to flash all of this information on the screen, as the user scans through the sample dates. However, location, date, and SAF # should be updated on the screen as the user scans the list, so that he can select the sample of interest quickly. Note that the HEIS number has not been entered yet, so the sample must be identified by the SAF #, or location and date.

After the sample is taken, the field tech accesses the sample, either by location and date, or SAF # and round number, and enters the following data:

- HEIS # (if not already entered)
- Sample date (actual sampling date)
- Chain of custody #s
- Logbook title and page numbers
- Temperature
- Conductivity
- Ph
- Sample clarity
- Sample color

Block Test Case

Block Test Results Expected

Field Name: DATE_COLLECT_START

Field Description

Block: FIELD_DATA_LOCATION
Title: Field Data for a Location

Field Name: DATE_COLLECT_START

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.

Not null 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
- () Store a series of records using the SHIFT ^{FL6} [duplicate field] key to carry forward field values from the previously stored record.

N/A

Block: FIELD_DATA_LOCATION
Title: Field Data for a Location

Field Name: DATE_COLLECT_START

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.

() Bypass the field by using the TAB key.

() No error should occur.

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

not null field

(X) 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 *not null* as a null value.

(X) Enter an invalid date. The invalid date should be rejected.

Field Name: IDENT_CHAIN_CUSTODY

Field Description

The identifier of the chain of custody form for the sample.

Field Test Case

Field Test Results Expected

Block: FIELD_DATA_LOCATION
Title: Field Data for a Location

Field Name: TITLE_LOGBOOK ✓

Field Description

Field Test Case

Field Test Results Expected

Field Name: NUM_LOGBOOK ✓

Field Description

Field Test Case

Field Test Results Expected

Field Name: NUM_LOG_PAGE ✓

Field Description

The page number within the log book.

Field Test Case

Field Test Results Expected

Field Name: SAMPLE_CLARITY ✓

Field Description

The clarity of the sample. LOV available.

Field Test Case

Field Test Results Expected

help txt states can leave empty - field is not null

731

733

Block: FIELD_DATA_LOCATION
Title: Field Data for a Location

Field Name: SAMPLE_COLOR ✓

Field Description

The color of the sample. A LOV is provided. Entering the first three characters of the color will cause the rest to be loaded automatically.

Field Test Case

Field Test Results Expected

Field Name: SAMPLE_TEMPERATURE ✓

Field Description

Field Test Case

Field Test Results Expected

Field Name: CONDUCTIVITY ✓

Field Description

Field Test Case

Field Test Results Expected

Field Name: PH ✓

Field Description

Field Test Case

Field Test Results Expected

Block: FIELD_DATA_LOCATION
 Title: Field Data for a Location

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: FIELD_DATA_LOCATION
Title: Field Data for a Location

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
- () 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: FIELD_DATA_LOCATION
Title: Field Data for a Location

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.

Block: SAMPLE
Title: Sample

Block Description

A sample is a defined sampling activity to be taken at a specific location.

Each sample is broken into four subsamples which are classified by how the subsample was digested. Each subsample is assigned a process number; the four forms are:

Raw sample
KOH fusion
Water digestion
Acid digestion.

Each subsample may be appropriate for many different methods. Some subsamples will share the same method. Lab names are not needed to define methods, because they are set by procedures specified by the customer.

Block Test Case

Block Test Results Expected

Field Name: IDENT_SAMPLE

not testable

Field Description

This field is the end user unique identifier for the sample. Sample identifiers for LEMIS will be HEIS sample numbers. HEIS numbers are assigned in the following manner. OSM gets a block of HEIS numbers from HEIS. Field sampling groups then get blocks of numbers from OSM. The HEIS numbers are assigned in the field on a chain of custody form.

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: SAMPLE
Title: Sample

Field Name: IDENT_SAMPLE

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.
 - () 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: SAMPLE
Title: Sample

Field Name: IDENT_SAMPLE

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.

Field Name: IDENT_HEIS_NUM ✓

Field Description

Field Test Case

Field Test Results Expected

Block: SAMPLE
Title: Sample

Field Name: SAMPLE_PURPOSE ✓

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.
- 11/2) 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: SAMPLE
Title: Sample

Field Name: SAMPLE_PURPOSE

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.
- N/A* (/) Any alphanumeric characters may be entered into the field.
- (/) Attempt to enter data as lower case. ~~An error should result.~~

Transaction: LE3310U1 Enter Field Data

```
=====
Comment ID : 733   Priority: LOW           Date: 17-DEC-92 ,
=====
Status      : ASSIGNED   Date: 18-DEC-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : LORRAINE CAMPBELL
System ID   : 92000009757   Oracle ROWID: 0000052B.0000.0007
Last Error  : -0
Last Msg    : FRM-40202   Field must be entered.
Table       : FIELD_DATA_LOCATION
Block       : FLDDAT
Field       : FLDDAT.NUM_LOG_PAGE
=====
```

DESCRIPTION

NOT NULL FIELD, HELP STATES FIELD MAY BE LEFT EMPTY

FIXED

12/18

✓
of fe

2.0 GENERIC

2.1 GE1001U1 MAINTAIN ADDRESS CODES

LEMIS Module Test Checklist Report

Name: MAINTAIN ADDRESS CODES
 Short Name: GE1001U1
 Purpose: To maintain address codes.

Initial Test Date: 11/18/92 JC

Verification Date: 11/18/92 JC

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

Block: TYPE_ADDR_CODE
 Title: Address Type

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

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_ADDR	CHAR	3	()	()	()	()	()
DESCR_TYPE_ADDR	CHAR	30	()	()	()	()	()

Module Description

This module is used to add, update or delete valid address codes and their associated descriptions (i.e. PRI - Primary Address).

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.

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 JA Date Complete 11/18/93
- () 3 - Initial unit testing complete.
 - Developer JA Date Complete 11/18/93

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 LC 11/18/92 Date Complete _____
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.
 - Developer _____ Date Complete _____

Module Test Plan

() 8 - Secondary testing complete, no problems found.
 Tester de Date Complete 11/18/92

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 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.

Module Test Plan

Project Manager Date Complete 1/18/92
 () 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 Date Complete 11/18/92

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/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.
 - N/A () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - N/A () 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: TYPE_ADDR_CODE
Title: Address Type

Block Description

Block Test Case

Block: TYPE_ADDR_CODE
Title: Address Type

Block Test Results Expected

Field Name: TYPE_ADDR

Field Description

This is a 3-character field which represents the type of address (i.e. PRI(mary), or SEC(condary), etc...).

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.
- () 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.

Block: TYPE_ADDR_CODE
 Title: Address Type

Field Name: TYPE_ADDR

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.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.

Field Test Case

- 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.
- (✓) 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
 - 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

Block: TYPE_ADDR_CODE
Title: Address Type

Field Name: TYPE_ADDR

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 COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).
- () ~~Store~~^{flp} a series of records using the SHIFT/~~FL~~ [duplicate field] key to carry forward field values from the previously stored record. *Will not Store Dup Field's*
- (✓) 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.~~ *ALL CAPS*
- (✓) Delete an existing value. ~~flp~~
- (✓) For NOT NULL fields, ~~the COM00107 message [field value may not be deleted] is displayed at commit.~~ *Fields must be entered*
- (✓) Attempt to bypass the field. An error message should tell you that the field data must be entered.

Block: TYPE_ADDR_CODE
Title: Address Type

Field Name: TYPE_ADDR

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: TYPE_ADDR_CODE
Title: Address Type

Field Name: DESCR_TYPE_ADDR

Field Description

This is a 30 character description of the address type.

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}~~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.
- N/A () 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. - no Autoskip on
- (✓) 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.

Field Test Case

Block: TYPE_ADDR_CODE
Title: Address Type

Field Name: DESCR_TYPE_ADDR

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.
 - (X) 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.
- () Display the -JN journal record rows for the table to verify that

*Field +
Dup column
same*

Dup

Dup

Block: TYPE_ADDR_CODE
Title: Address Type

Field Name: DESCR_TYPE_ADDR

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. *Dup*
- (✓) 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*

2.2 GE1002U1 MAINTAIN COUNTRY CODES

LEMIS Module Test Checklist Report

Name: MAINTAIN COUNTRY CODES
 Short Name: GE1002U1
 Purpose: To maintain country codes.

Initial Test Date: 11/19/92 *LC*Verification Date: 11/24/92 *ZK*

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

Block: TYPE_COUNTRY_CODE
 Title: Country Code

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

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

Module DescriptionModule Help

This transaction is used to add, delete, and update country codes and descriptions.

To store a new country code:

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

Module Help

If all of the spaces are filled and more country 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 country 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 country 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.

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.

Module Test Plan

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.

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 *JD* Date Complete 11/29/92

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.

Module Test Plan

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 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/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.
 - N/A () 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 ~~SPRT~~/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_COUNTRY_CODE
Title: Country Code

Block Description

Block Test Case

Block Test Results Expected

Field Name: COUNTRY

Field Description

Field Test Case

Field Test Results Expected

*no testing
will do Description test on
both Fields.
indicated by 2 (✓s)*

Block: TYPE_COUNTRY_CODE
Title: Country Code

Field Name: DESCR_COUNTRY

Field Description

This field contains a 1-30 character descriptive title of the country.

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

- 520
- (✓) 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
 - N/A (✓) 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
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

Block: TYPE_COUNTRY_CODE
Title: Country Code

Field Name: DESCR_COUNTRY

Field Test Results Expected

- (✓) 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.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~ Auto change to ALL CAPS
- (✓) Delete an existing value.
- 521 () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. Field must be entered FRM 40202 DAP
- () 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.

Country col
unique ID cannot be
dup Field.

Application: GENERIC Application Maintenance

Transaction: GE1002U1 Maintain Country Codes

=====

Comment ID : 520 Priority: LOW Date: 19-NOV-92

=====

Status : FIXED Date: 28-NOV-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TEST USER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40102 Record must be entered or deleted first.

Table : TYPE_COUNTRY_CODE

Block : CNTRY

Field : CNTRY.COUNTRY

DESCRIPTION

Help states there is a default, None found when using down arrow down column.

CORRECTIVE ACTION

11/29/92 - Fixed help to match form.

28-NOV-92

Page 2 of 3

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE1002U1 Maintain Country Codes

```
=====
Comment ID : 521      Priority: LOW      Date: 19-NOV-92
=====
Status      : TESTCHNG      Date: 28-NOV-92      Class:
Type        : PROBLEM
Assigned To :              KEN ATKINS
Entered By  :              TEST USER
System ID   :              83              Oracle ROWID: 0000016C.0005.000B
Last Error  :              -0
Last Msg    : FRM-40202      Field must be entered.
Table       : TYPE_COUNTRY_CODE
Block       : CNTRY
Field       : CNTRY.COUNTRY
=====
```

DESCRIPTION

Deleting a value. not null field error should be frm40202 field must be entered

CORRECTIVE ACTION

11-29-92 - The form seemed to work as it should. Maybe this is a confusion about the wording in the test booklet.

2.3 GE1003U1 MAINTAIN STATE CODES

LEMIS Module Test Checklist Report

Name: MAINTAIN STATE CODES
 Short Name: GE1003U1
 Purpose: To maintain state code:

Initial Test Date: 11/19/92 LC

Verification Date: 11/29/92 LC

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

Block: TYPE_STATE_CODE
 Title: State Code

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

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

Module Description

Module Help

This transaction is used to add, delete, and update state codes and descriptions.

To store a new state code:

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

Module Help

If all of the spaces are filled and more state 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 state 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 state 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 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.

Module Test Plan

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.

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 HC Date Complete 4/29/80

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

Module Test Plan

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 SG 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 SA Date Complete 11/29/92

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/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 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_STATE_CODE
Title: State Code

Block Description

Block Test Case

Block Test Results Expected

Field Name: STATE

Field Description

This field contains a 2 character abbreviation for the state.

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.

Block: TYPE_STATE_CODE
 Title: State Code

Field Name: STATE

Field Test Case

() 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. DUP

N/A () 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~~ (✓) 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

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.

Block: TYPE_STATE_CODE
Title: State Code

Field Name: STATE

Field Test Results Expected

- N/A*
- () 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 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.~~
 - ~~() 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.

*Cannot
Store unique
Fields*

Fields must be entered

Block: TYPE_STATE_CODE
Title: State Code

Field Name: STATE

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_STATE_CODE
Title: State Code

Field Name: DESCR_STATE

Field Description

This field contains a 1-30 character descriptive title of the state.

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.
- 527 () 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
 - 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.
 - (✓) 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
 - () Value above valid range should display message COM00101 or COM00103 (if no LOV).

Block: TYPE_STATE_CODE
Title: State Code

Field Name: DESCR_STATE

Field Test Results Expected

- (✓) 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.
- (✓) 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. *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.

2.4 GE1110U1 SECURITY PERSON MAINTENANCE

LEMIS Module Test Checklist Report

Name: GE1110U1 - SECURITY PERSON MAI
 Short Name: GE1110U1
 Purpose: Maintain Person Information for security purp

Initial Test Date: 12/16/92 SC

Verification Date: 12/21/92 A

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

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	30	()	()	()	()	()

Block: DATA_OWNER_PERSON_J
 Title: Data Ownership(s) for this person

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	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()
IDENT_PERSON	CHAR	12	()	()	()	()	()
DATE_EXPIRES	DATETIME	9	()	()	()	()	()

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	36	()	()	()	()	()

Block: ROLE_FUNC_UNIT_J
Title: Access role(s) for this person

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 maintain person information for this system and to assign people access roles and data ownership.

Any person who needs access to the database MUST have an entry in this table. In order to set a user up so that they have access to the database the following must be done:

Module Help

- 1) Create a person record with the correct oracle ID in the 'Oracle ID' field. The correct oracle ID should be 'OPSS\$' followed by the user's unix account id. Example: If the unix account is 'e83681' then the Oracle ID should be OPSS\$E83681.
- 2) Add the appropriate access roles and data ownerships for the desired application(s).
- 3) If the Oracle account has not yet been set up, use the special function key (Shift-F5) to enter an Oracle system manager account and password. When you press commit (F10) the OPSS\$ Oracle account will be created.

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.

Module Test PlanSecondary 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

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.

Module Test Plan

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 Date Complete 12/21/92

() 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 Date Complete 12/21/92

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.

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

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

(X) Transaction help describes:

(X) Purpose of the transaction

(X) Functions performed by the transaction

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

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

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

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

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

(X) 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: Data Ownership(s) for this person

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 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

Could not find Relationship!

Block: DATA_OWNER_PERSON_J
 Title: Data Ownership(s) for this person

Field Name: RELATIONSHIP

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).
- () 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.

Block: DATA_OWNER_PERSON_J
Title: Data Ownership(s) for this person

Field Name: RELATIONSHIP

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.

Block: PERSON

Title: Person

Block Description

Block Test Case

Block Test Results Expected

Field Name: NAME_LAST

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

Block: PERSON
Title: Person

Field Name: NAME_LAST

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).
- N/A (.) 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.
 - () 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.

Block: PERSON
Title: Person

Field Name: NAME_FIRST

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 COM00103 (if no LOV).
- (/) Store a series of records using the SHIFT/^{FB}E7 [duplicate field] key to carry forward field values from the previously stored record.

Block: PERSON
Title: Person

Field Name: NAME_FIRST

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.
- (/) 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: PERSON
Title: Person

Comment
726 OK A

Field Name: INITIALS

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 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.

Comment

86

Block: PERSON
 Title: Person

Field Name: INITIALS

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.
- (/ 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: PERSON
 Title: Person

Field Name: TITLE

Field Description

This is the job title of the person represented by this record. single format across all significant tables in the database. The number value is generated by a sequence generator IDENT_SEQUENCE. This generator creates an 11-digit number, the first two of which are the fiscal year. The generator is reset every fiscal year and the year value incremented. As each new significant table row is created, the sequence generator supplies the unique identifier and increments by one. Under this approach no two significant table rows in the database have the same identifier value.

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

Block: PERSON
Title: Person

Field Name: TITLE

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.
 - ☒ 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~~^{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.
- ☒ 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: PERSON
 Title: Person

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: PERSON
Title: Person

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
- () 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.

Block: PERSON
Title: Person

Field Name: FLAG_TEXT

Field Test Results Expected

- () 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: PERSON
Title: Person

Field Name: IDENT_PERSON

Field Description

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.
 - (X) 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.
 - () 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
 - () 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.
 - () Change the field where a range of valid values exists:
 - () Value below valid range should display message COM00101 or COM00103 (if no LOV).
- n/a (X) Store a series of records using the SHIFT/F7 [duplicate field] key to carry forward field values from the previously stored record.

Block: PERSON
Title: Person

Field Name: IDENT_PERSON

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.
- (✓) 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: PERSON
Title: Person

Field Name: DATE_EXPIRES

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

- 727
deck
A
- (✓) 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).

Block: PERSON
Title: Person

Field Name: DATE_EXPIRES

Field Test Results Expected

- () Store a series of records using the SHIFT/7 [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.
- () Enter an invalid date. The invalid date should be rejected.

Block: ROLE_FUNC_UNIT_J
Title: Access role(s) for this person

Block Description

Block Test Case

Block Test Results Expected

no testing

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE1110U1 Maintain Person

=====

Comment ID : 726 Priority: LOW Date: 16-DEC-92

=====

Status : ASSIGNED Date: 16-DEC-92 Class:

Type : COMMENT

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : 7556 Oracle ROWID: 00000024.0004.000B

Last Error : -0

Last Msg : FRM-40350 Query caused no records to be retrieved.

Table : PERSON

Block : PERSON

Field : PERSON.INITIALS

DESCRIPTION

=====

initials has four characters in field. help states to enter "Middle" initial(s). Do we want text to read just initials? or last, middle, first initials? just middle?

*NOT
problem**USE CAN USE
EITHER.**12/21/92
✓*

Transaction: GE1110U1 Maintain Person

```
=====
Comment ID : 725   Priority: LOW   Date: 16-DEC-92
=====
Status      : ASSIGNED   Date: 16-DEC-92   Class:
Type        : PROBLEM
Assigned To : KEN ATKINS
Entered By  : LORRAINE CAMPBELL
System ID   : 7030       Oracle ROWID: 00000021.0005.000B
Last Error  : -0
Last Msg    : -0
Table       : PERSON
Block       : PERSON
Field       : PERSON.NAME_LAST
=====
```

DESCRIPTION

when query error occurs ID_ROLE??

POST-CHANGE ON ID_ROLE
Triggered
AND ID-OWNER

Verified
A

Programmer's Bug Report

Transaction: GE1110U1 Maintain Person

=====

Comment ID : 727 Priority: LOW Date: 16-DEC-92

=====

Status : ASSIGNED Date: 16-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : 7556 Oracle ROWID: 00000024.0004.000B

Last Error : -0

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

Table : PERSON

Block : PERSON

Field : PERSON.DATE_EXPIRES

DESCRIPTION

=====

Help should give an example of the format used; DD-MON-YY.

Fixed
12/21
✓
verified 12/21/92
[Signature]

2.5 GE1210U1 MAINTAIN ORAGANIZATION

LEMIS Module Test Checklist Report

Name: GE1210U1 - MAINTAIN ORGANIZATI
 Short Name: GE1210U1
 Purpose: Maintain Organization Information.
 Initial Test Date: 12/18/92 AC
 Verification Date: 1/26/93 RJA

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

Block: ADDRESS
 Title: Address

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

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_ADDR	CHAR	3	()	()	()	()	()
ADDR_LINE_1	CHAR	30	()	()	()	()	()
ADDR_LINE_2	CHAR	30	()	()	()	()	()
ADDR_LINE_3	CHAR	30	()	()	()	()	()
ADDR_CITY	CHAR	20	()	()	()	()	()
STATE	CHAR	2	()	()	()	()	()
ZIPCODE	CHAR	10	()	()	()	()	()
COUNTRY	CHAR	2	()	()	()	()	()
ADDR_PHONE	CHAR	18	()	()	()	()	()
ADDR_MS	CHAR	10	()	()	()	()	()
ADDR_BLDG	CHAR	10	()	()	()	()	()
ADDR_ROOM	CHAR	10	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

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	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: TYPE_ADDR_CODE
Title: Address Type

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

Block: TYPE_COUNTRY_CODE
Title: Country Code

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

Block: TYPE_STATE_CODE
Title: State Code

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

Module Description

Module Help

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.

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.

Module Test Plan

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

- ☐ 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 RS Date Complete 1/26/83
- ☐ 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 A Date Complete 1/26/93

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. *no Trans help*

- 737 () 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.

- 738 () Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records. *Alpha order not working*
- ✓ 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: ADDRESS
Title: Address

Block Description

Block Test Case

Block Test Results Expected

Block: ADDRESS
Title: Address

Field Name: TYPE_ADDR

Field Description

This field identifies the type of address (i.e. PRI(mary), SEC(ondary), etc.)

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.

Block: ADDRESS
Title: Address

Field Name: TYPE_ADDR

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
 - ☐ Value above valid range should display message COM00101 or COM00103 (if no LOV).
- ☒ (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.
- ☒ Modify each default value with a valid alternative.
- ☒ (B) Modify each default value with an invalid alternative.
- ☒ (C) Any alphanumeric characters may be entered into the field.
- ☒ Delete an existing value.
- ☒ Attempt to enter data as lower case. An error should result.
- ☒ 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: ADDRESS
Title: Address

Field Name: TYPE_ADDR

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: ADDRESS
Title: Address

Field Name: ADDR_LINE_1

Field Description

Line 1 of the address.

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

Block: ADDRESS
Title: Address

Field Name: ADDR_LINE_1

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/Enter [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.
- () Any alphanumeric characters may be entered into the field.

Block: ADDRESS
Title: Address

Field Name: ADDR_LINE_2

Field Description

Line 2 of the Address.

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: ADDRESS
Title: Address

Field Name: ADDR_LINE_2

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/^{Die} [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.
- (/) Any alphanumeric characters may be entered into the field.

Block: ADDRESS
Title: Address

Field Name: ADDR_LINE_3

Field Description

Line 3 of the address.

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

Block: ADDRESS
Title: Address

Field Name: ADDR_LINE_3

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/~~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.
 - () 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: ADDRESS
Title: Address

Field Name: ADDR_CITY

Field Description

This is the city associated with this address.

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: ADDRESS
Title: Address

Field Name: ADDR_CITY

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/~~BT~~^{ESC} [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.
- ☒ Any alphanumeric characters may be entered into the field.
- ☒ Attempt to enter data as lower case. ~~An error should result.~~

Block: ADDRESS
Title: Address

Field Name: STATE

Field Description

This is the two character state code associated with this address.

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. *no Defaults* ✓
- () 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.
 - () Field help describes:
 - (✓) Purpose of the field
 - () LOV reference if available

*Blank ?
mandatory
optional*

Block: ADDRESS
Title: Address

Field Name: STATE

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.
 - ☒ 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/RT [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: ADDRESS
Title: Address

Field Name: STATE

Field Test Results Expected

- 11/18 () 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: ADDRESS
Title: Address

Field Name: ZIPCODE

Field Description

This is the ZIPcode for the address.

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: ADDRESS
Title: Address

Field Name: ZIPCODE

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/^{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.
- () 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: ADDRESS
Title: Address

Field Name: COUNTRY

Field Description

This is the two character country code associated with this address.

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.
- NA3 () Accept the (default).
- NA () Override the default with a valid entry value.
- () 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.

Field Test Results Expected

- () Display field help. Verify that the narrative adequately describes the purpose and functions of the field in end-user terms.
- NA3 () Field help describes:
 - () Purpose of the field
 - () LOV reference if available

Block: ADDRESS
Title: Address

Field Name: COUNTRY

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.
 - ☒ 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/¹⁰F7 [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: ADDRESS
Title: Address

Field Name: COUNTRY

Field Test Results Expected

- n/a*
- () 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.
- Null field*

Block: ADDRESS
Title: Address

Field Name: ADDR_PHONE

Field Description

This is the phone number associated with this address.

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

Block: ADDRESS
Title: Address

Field Name: ADDR_PHONE

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/^{FLC}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.
 - ☐ 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: ADDRESS
Title: Address

Field Name: ADDR_MS

Field Description

The Mail Stop associated with this address.

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: ADDRESS
Title: Address

Field Name: ADDR_MS

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/^{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.
- () 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: ADDRESS
 Title: Address

Field Name: ADDR_BLDG

Field Description

Building identifier associated with the address.

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: ADDRESS
Title: Address

Field Name: ADDR_BLDG

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/EP [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.
- (/) Any alphanumeric characters may be entered into the field.
- (/) Attempt to enter data as lower case. An error should result.

Block: ADDRESS
Title: Address

Field Name: ADDR_ROOM

Field Description

This is the room identifier (if any) for this address.

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: ADDRESS
Title: Address

Field Name: ADDR_ROOM

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/^{file}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.
- (/) Any alphanumeric characters may be entered into the field.
- (/) Attempt to enter data as lower case. An error should result.

Block: ADDRESS
Title: Address

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: ADDRESS
Title: Address

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
- () 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: ADDRESS
Title: Address

Field Name: FLAG_TEXT

Field Test Results Expected

- () Attempt to enter data as lower case. An error should result.
 - () 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: ORGANIZATION
 Title: Organization

Block Description

Block Test Case

Block Test Results Expected

Field Name: IDENT_ORGAN

Field Description

Organization title abbreviation. Format is XXX/XXX (alphanumeric).

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

Block: ORGANIZATION
Title: Organization

Field Name: IDENT_ORGAN

Field Test Results Expected

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).
- 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.
- (/) 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: ORGANIZATION
Title: Organization

Field Name: TITLE

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.
- 739 () 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:
 - 740 () 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/EP [duplicate field] key to carry forward field values from the previously stored record.

Both optional Blank & mandatory fill in stated in help

Block: ORGANIZATION
Title: Organization

Field Name: TITLE

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.
- (/) 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.

Block: ORGANIZATION
Title: Organization

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: ORGANIZATION
Title: Organization

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

- () 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.

Block: ORGANIZATION
Title: Organization

Field Name: FLAG_TEXT

Field Test Results Expected

- () 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.

*no fields
for these!*

Block: TYPE_ADDR_CODE
Title: Address Type

Block Description

Block Test Case

Block Test Results Expected

Field Name: DESCR_TYPE_ADDR

Field Description

This is a 30 character description of the address type.

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

Block: TYPE_ADDR_CODE
Title: Address Type

Field Name: DESCR_TYPE_ADDR

Field Test Results Expected

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.
- () 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: TYPE_COUNTRY_CODE
Title: Country Code

Block Description

Block Test Case

Block Test Results Expected

Field Name: DESCR_COUNTRY

Field Description

This field contains a 1-30 character descriptive title of the country.

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

Block: TYPE_COUNTRY_CODE
Title: Country Code

Field Name: DESCR_COUNTRY

Field Test Results Expected

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.
- () 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: TYPE_STATE_CODE
Title: State Code

Block Description

Block Test Case

Block Test Results Expected

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE1210U1 Maintain Organization

=====

Comment ID : 743 Priority: LOW Date: 18-DEC-92

=====

Status : ASSIGNED Date: 18-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : 92000009775 Oracle ROWID: 00000C73.0017.000B

Last Error : -0

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

Table : ADDRESS

Block : ADDR

Field : ADDR.COUNTRY

DESCRIPTION

there are no defaults to accept. help states help both allowable and
nonallowableFIXED
12/21✓
1/26

Application: GENERIC Application Maintenance

Transaction: GE1210U1 Maintain Organization

=====

Comment ID : 742 Priority: LOW Date: 18-DEC-92

=====

Status : ASSIGNED Date: 18-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : 92000009775 Oracle ROWID: 00000C73.0018.000B

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 2 records posted an

Table : ADDRESS

Block : ADDR

Field : ADDR.STATE

DESCRIPTION

no description with LOV

FIXED
12/21
✓✓
~~1/26~~
1/26

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE1210U1 Maintain Organization

=====

Comment ID : 741 Priority: LOW Date: 18-DEC-92

=====

Status : ASSIGNED Date: 18-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : 92000009775 Oracle ROWID: 00000C73.0017.000B

Last Error : -0

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

Table : ADDRESS

Block : ADDR

Field : ADDR.STATE

DESCRIPTION

help states blank both allowable and not allowable.

help states defaults available but cannot accept them

Fixed

12/21

✓
1/26

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE1210U1 Maintain Organization

=====

Comment ID : 740 Priority: LOW Date: 18-DEC-92

=====

Status : ASSIGNED Date: 18-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : 92000009775 Oracle ROWID: 00000024.0005.000B

Last Error : -0

Last Msg : FRM-40202 Field must be entered.

Table : ORGANIZATION

Block : ORGAN

Field : ORGAN.TITLE

DESCRIPTION

field cannot be left empty but help states both blank allowable and nonallowable.

Fixed
12/21
✓
✓
1/26

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE1210U1 Maintain Organization

=====

Comment ID : 739 Priority: LOW Date: 18-DEC-92

=====

Status : ASSIGNED Date: 18-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : 92000009775 Oracle ROWID: 00000024.0005.000B

Last Error : -0

Last Msg : FRM-40202 Field must be entered.

Table : ORGANIZATION

Block : ORGAN

Field : ORGAN.TITLE

DESCRIPTION

text is entered in lower case instead of upper caseFIXED
12/21
✓✓
1/26

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE1210U1 Maintain Organization

=====

Comment ID : 738 Priority: LOW Date: 18-DEC-92

=====

Status : ASSIGNED Date: 18-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : 7034 Oracle ROWID: 00000021.0009.000B

Last Error : -0

Last Msg : FRM-40400 Transaction complete -- 2 records posted an

Table : ORGANIZATION

Block : ORGAN

Field : ORGAN.IDENT_ORGAN

DESCRIPTION

order function (synchronization) not working*FIXED*
12/21 ✓✓
1/26

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE1210U1 Maintain Organization

Comment ID : 737 Priority: LOW Date: 18-DEC-92

Status : ASSIGNED Date: 18-DEC-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : LORRAINE CAMPBELL

System ID : 7034 Oracle ROWID: 00000021.0009.000B

Last Error : -0

Last Msg : FRM-40350 Query caused no records to be retrieved.

Table : ORGANIZATION

Block : ORGAN

Field : ORGAN.IDENT_ORGAN

DESCRIPTION

no transaction help available

*due 12/28
A**✓ 1/26*

2.6 GE3002U1 MAINTAIN TABLE TYPE CODES

LEMIS Module Test Checklist Report

524

Name: MAINTAIN TABLE TYPE CODES
 Short Name: GE3002U1
 Purpose: To maintain Table Type Codes

Initial Test Date: 11/19/92 JC

Verification Date: 11/29/92 JH

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

Block: DOCUMENT_HDR
 Title: Document Block

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

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
IDENT_DOCUMENT	CHAR	15	()	()	()	()	()

Block: TYPE_TABLE_CODE
 Title: System Table Type Codes

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

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_TABLE	CHAR	6	()	()	()	()	()
DESCRIPTION	CHAR	25	()	()	()	()	()
SUBJECT_DATA	CHAR	25	()	()	()	()	()

Module Description

Module Help

This transaction is used to add, update, or delete table type codes. This module supports the maintenance and retrieval of table type codes. This table maintains a list of valid codes that describe the types of tables that may exist in the database.

To store a new table 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 table 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 table 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 table 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 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

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 CA Date Complete 1/14/93

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.

Module Test Plan

() 8 - Secondary testing complete, no problems found.
 Tester Je Date Complete 4/29/90

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 Je Date Complete 12/29/92
 () 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 2/29/93

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
 - 325 (✓) Change each ^{variable} 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_TABLE_CODE
Title: System Table Type Codes

Block Description

Block Test Case

Block Test Results Expected

no testing
for Code
Field

Block: TYPE_TABLE_CODE
Title: System Table Type Codes

Field Name: TYPE_TABLE

Field Description

Field Test Case

Field Test Results Expected

*Where is
Type-Table
Field?*

Block: TYPE_TABLE_CODE
Title: System Table Type Codes

Field Name: DESCRIPTION

Field Description

This field contains a 1-30 character descriptive title of the table.

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

- 526
- () 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
 - V/A () Default value when appropriate
 - (✓) Change the field in an existing record. Verify that the change

Block: TYPE TABLE CODE
 Title: System Table Type Codes

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 ^{FC} [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.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~ ^{ALL CAPS}
- (✓) Delete an existing value.
 - () For NOT NULL fields, the ~~COM00107~~ message [field value may not be deleted] is displayed at commit.

*not null
Field*

FRM 40202

Block: TYPE_TABLE_CODE
Title: System Table Type Codes

Field Name: DESCRIPTION

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.

Field Name: SUBJECT_DATA

Field Description

Field Test Case

Field Test Results Expected

*no testing
for Subject Data
Where is online
Doc Strt ?*

Programmer's Bug Report

Application: GENERIC Application Maintenance

ansaction: GE3002U1 Table Type Code Maintenance

=====

Comment ID : 524 Priority: LOW Date: 19-NOV-92

=====

Status : FIXED Date: 28-NOV-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TEST USER

System ID : Oracle ROWID: 0000011D.0011.000F

Last Error : -0

Last Msg : FRM-40811 Shell command had error.

Table : TRANSACTION_NEST

Block : TRNNST

Field : TRNNST.PTR

DESCRIPTION

=====

Title on printout test is "Maintain Table Type Codes" menu is "Table
Type Codes Maintenance"

CORRECTIVE ACTION

=====

11/28/92 - Fixed

Programmer's Bug Report

Application: GENERIC Application Maintenance

Transaction: GE3002U1 Table Type Code Maintenance

Comment ID : 525 Priority: LOW Date: 19-NOV-92

Status : NOTPROB Date: 28-NOV-92 Class:
Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TEST USER

System ID : 6784 Oracle ROWID: 0000001B.0000.000B

Last Error : -0

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

Table : TYPE_TABLE_CODE

Block : TYPTAB

Field : TYPTAB.DSP_IDENT_DOCUMENT

DESCRIPTION

what is this column for??? It has LOV activated. Help needs help!

CORRECTIVE ACTION

11-28-92 - Removed this column from the screen. It is no longer needed.

2.7 GE3003U1 MAINTAIN TEST TYPES

LEMIS Module Test Checklist Report

Name: MAINTAIN TEXT TYPES

Short Name: GE3003U1

Purpose: To maintain text types for related textual da

Initial Test Date: 11/19/92 *RC*Verification Date: 11/25/92 *Lz*

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

Block: TYPE_TEXT_CODE
 Title: Text Type Codes

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

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_TEXT	CHAR	3	()	()	()	()	()
DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

This module is used to add, update or delete text types in the text type table.

Module Help

Use this module to add, update or delete text types and their descriptions in the text type table.

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.

Developer A Date Complete 11/17/92

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.
- () 6 - Problem resolution in progress.
- () 7 - Problem resolution complete.

Module Test Plan

() 8 - Secondary testing complete, no problems found.

Tester SCDate Complete 11/29/92

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.

() 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 SCDate Complete 12/29/92

() 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 A Date Complete 2/24/82

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]
- 523 () 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/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 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 ^{update} 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. *unique fields cannot be updated*
- (✓) 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_TEXT_CODE
Title: Text Type Codes

Block Description

Block Test Case

Block Test Results Expected

*no field
no block
testing*

Block: TYPE_TEXT_CODE
Title: Text Type Codes

Field Name: TYPE_TEXT

Field Description

Field Test Case

Field Test Results Expected

Field Name: DESCRIPTION

Field Description

Field Test Case

Field Test Results Expected

2.8 GE3004U1 VALID TEST TYPES FOR TABLE

LEMIS Module Test Checklist Report

Name: VALID TEXT TYPES FOR TABLE

Short Name: GE3004U1

Purpose: To maintain valid text types for each table.

Initial Test Date: 11-25-92 RJAVerification Date: 11-25-92 RJA

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

Block: TYPE_TABLE_CODE

Title: System Table Type Codes

Help	Scroll	Text	Navigate	Ins	Del	Upd	Qry	
()	()	()	()	()	()	()	()	
Data Element		Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
TYPE_TABLE		CHAR	6	()	()	()	()	()
DESCRIPTION		CHAR	30	()	()	()	()	()
SUBJECT_DATA		CHAR	30	()	()	()	()	()

Block: TYPE_TEXT_CODE

Title: Text Type Codes

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

Block: VALID_TEXT_TYPE
 Title: Valid Text Types

Help ()	Scroll ()	Text ()	Navigate ()	Ins ()	Del ()	Upd ()	Qry ()
Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
FLAG_PRIMARY	CHAR	1	()	()	()	()	()
TYPE_TEXT	CHAR	3	()	()	()	()	()

Module Description

This module is used to add, update or delete text types for a given table type. The table type and description cannot be updated in this module.

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.

Module Test Plan

- () 2 - Development complete, initial unit testing in progress.
 () Help added
 () LOV tested
 Developer Date Complete 11/25/92
 () 3 - Initial unit testing complete.
 Developer 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 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.

Module Test Plan

- () 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 SA Date Complete 11/24/92
- () 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 11/25/92

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
 - () 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: VALID_TEXT_TYPE
Title: Valid Text Types

Block Description

Block Test Case

Block Test Results Expected

Field Name: FLAG_PRIMARY

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/F4 [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.

Block: VALID_TEXT_TYPE
 Title: Valid Text Types

Field Name: FLAG_PRIMARY

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.
- (✓) 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.

Block: VALID_TEXT_TYPE
Title: Valid Text Types

Field Name: FLAG_PRIMARY

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

Block: VALID_TEXT_TYPE
Title: Valid Text Types

Field Name: FLAG_PRIMARY

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 COM0010~~4~~ 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: VALID_TEXT_TYPE
Title: Valid Text Types

Field Name: TYPE_TEXT

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/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.
- (✓) 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.

Block: VALID_TEXT_TYPE
Title: Valid Text Types

Field Name: TYPE_TEXT

Field Test Plan

- () 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 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: VALID_TEXT_TYPE
 Title: Valid Text Types

Field Name: TYPE_TEXT

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.
 - () 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.

2.9 GE3005U1 TYPE DATA VALID CODE

LEMIS Module Test Checklist Report

Name: TYPE DATA VALID CODE
 Short Name: GE3005U1
 Purpose: To maintain data validation codes.

Initial Test Date: 11/18/92 JC

Verification Date: ~~11/18/92 JC~~ 11/29/92 J3

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

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
LEVEL DATA VALID	NUMBER	1	()	()	()	()	()
CODE DESCRIPTION	CHAR	30	()	()	()	()	()

Module Description

This module is used to add, update or delete data validation codes and their descriptions.

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.

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 A Date Complete 11/14/92
- () 3 - Initial unit testing complete.
 - Developer A Date Complete 11/14/92

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

Module Test Plan

() 8 - Secondary testing complete, no problems found.

Tester Date Complete 11/18/92

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 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.

Module Test Plan

Project Manager 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 Date Complete 11/29/92

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/A) Where multiple page, what happens on each page
- () 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.
 - N/A) Synchronization of all blocks occurs while scrolling to the end of the string of retrieved records.
 - N/A) 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.
 - 512 () 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 ^{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 ^{updatable} 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.

Single
Digit
Column only.
wild card
hard to test.

Block: TYPE_DATA_VALID_CODE
Title: Data Validation Levels

Block Description

Block Test Case

Block: TYPE_DATA_VALID_CODE
Title: Data Validation Levels

Block Test Results Expected

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 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.
- () Attempt to store alphabetic characters in the field.
- () Begin the field entry with a space. An error should result.
- () Delete an existing data field value. Determine that the system will display an error if the data field is NOT NULL.

Block: TYPE_DATA_VALID_CODE
Title: Data Validation Levels

Field Name: LEVEL_DATA_VALID

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.
- () Modify each default value with a valid alternative.
- () Modify each default value with an invalid alternative.

Field Test Case

- () Accept the default. *not able to test.*
- (/ Override the default with a valid entry value.
- (/ Override the default with an invalid data value.
- N/A) Clear an existing field and attempt to update the record.
- (/ Attempt to enter alpha data.
- N/A) 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.
- N/A) 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
 - N/A () Required indicator when field is NOT NULL
 - (/ Default value when appropriate

Block: TYPE_DATA_VALID_CODE
 Title: Data Validation Levels

Field Name: LEVEL_DATA_VALID

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 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/FG [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.
- 517 () 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.
- 518 () 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

N/A for not null

Block: TYPE_DATA_VALID_CODE
Title: Data Validation Levels

Field Name: LEVEL_DATA_VALID

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.

519

Block: TYPE_DATA_VALID_CODE
Title: Data Validation Levels

Field Name: CODE_DESCRIPTION

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/^{fg}~~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.
- N/A () 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.
- (✓) 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.~~ *Dep*
- () ~~Enter data into the field, backspace the data out of the field and attempt to TAB to the next field. An error should occur.~~ *Dep*

Block: TYPE_DATA_VALID_CODE
 Title: Data Validation Levels

Field Name: CODE_DESCRIPTION

Field Test Case

- (✓) Begin field with a space. ~~13-2~~
- (✓) 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. Dup
- (✓) 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. N/A
- (✓) 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}~~F7~~ [duplicate field] key to carry forward field values from the previously stored record. Dup

Block: TYPE_DATA_VALID_CODE
Title: Data Validation Levels

Field Name: CODE_DESCRIPTION

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.
- (✓) Any alphanumeric characters may be entered into the field.
- () Attempt to enter data as lower case. An error should result. *no error*
- () Delete an existing value. *ALL CAPS*
- () For NOT NULL fields, the COM00107 message [field value may not be deleted] is displayed at commit. *Def*
- (✓) 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: GENERIC Application Maintenance

Transaction: GE3005U1 Maintain Data Validation Codes

Comment ID : 512 Priority: LOW Date: 18-NOV-92

Status : FIXED Date: 18-NOV-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TEST USER

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40353 Query cancelled.

Table : TYPE_DATA_VALID_CODE

Block : LVDAVA

Field : LVDAVA.LEVEL_DATA_VALID

DESCRIPTION

looks like field may be a multiple character field, yet only one character appears. Won't accept command to do a single query.

CORRECTIVE ACTION

11/18/92 - Changed the query width to 1.

Programmer's Bug Report

Transaction: GE3005U1 Maintain Data Validation Codes

Comment ID : 517 Priority: LOW Date: 18-NOV-92

Status : TESTCHNG Date: 18-NOV-92 Class:

Type : PROBLEM

Assigned To: KEN ATKINS

Entered By : TEST USER

System ID : 6723

Oracle ROWID: 00000171.0005.000B

Last Error : -0

Last Msg : FRM-50016 Legal characters are 0-9 - + E .

Table : TYPE_DATA_VALID_CODE

Block : LVDAVA

Field : LVDAVA.LEVEL_DATA_VALID

DESCRIPTION

numeric data comma, plus, minus, and period cannot be added without getting error message. Numbers 0 through 9 work.

CORRECTIVE ACTION

11/18/92 - , + - . are not valid for 1 digit numbers. Modified help to reflect this.

Programmer's Bug Report

Transaction: GE3005U1 Maintain Data Validation Codes

```
=====
Comment ID : 518   Priority: LOW           Date: 18-NOV-92
=====
Status      : TESTCHNG   Date: 18-NOV-92   Class:
Type        : PROBLEM
Assigned To : LARRY TOWNER
Entered By  : TEST USER
System ID   : 44         Oracle ROWID: 00000171.0003.000B
Last Error  : -0
Last Msg    : FRM-40200  Field is protected against update.
Table       : TYPE_DATA_VALID_CODE
Block       : LVDAVA
Field       : LVDAVA.LEVEL_DATA_VALID
=====
```

DESCRIPTION

By trying to delete a value using F8 the message is "FRM40200 Field is protected against update" instead of com00107 field value may not be deleted.

CORRECTIVE ACTION

11/18/92 - Larry, I do not know where COM00107 came from in the black book? If a field is set to be non-updateable, than ANYTHING that attempts to change it (clear field, lov, etc) will give the FRM-40200 error.

Transaction: GE3005U1 Maintain Data Validation Codes

```
=====
Comment ID : 519   Priority: LOW           Date: 18-NOV-92
=====
Status      : TESTCHNG   Date: 18-NOV-92   Class:
Type        : PROBLEM
Assigned To : LARRY TOWNER
Entered By  : TEST USER
System ID   :          44   Oracle ROWID: 00000171.0003.000B
Last Error  : -0
Last Msg    : FRM-40202   Field must be entered.
Table       : TYPE_DATA_VALID_CODE
Block       : LVDAVA
Field       : LVDAVA.LEVEL_DATA_VALID
=====
```

DESCRIPTION

not able to backspace data out of the field. No error message when tabbing to next field, tab is allowed.

CORRECTIVE ACTION

11/18/92 - This test cannot be performed when the data fills the whole field. This means that in one character fields it cannot be done at all.

2.10 GE3007U1 MAINTAIN COMMENT STATUS CODES

LEMIS Module Test Checklist Report

Name: MAINTAIN COMMENT STATUS CODES

Short Name: GE3007U1

Purpose: To maintain list of comment status codes

Initial Test Date: 11-25-92 RJAVerification Date: 11-25-92 RJA

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

Block: TYPE_COMMENT_STATUS_CODE

Title: Comment Status Code

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

Data Element	Datatype	Dis Len	Help	LOV	Val/ Range	Not Null	Ins
COMMENT_STATUS	CHAR	10	()	()	()	()	()
DESCR_STATUS	CHAR	30	()	()	()	()	()

Module Description

To maintain the comment status codes.

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.

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 11/25/92
- () 3 - Initial unit testing complete.
 - Developer 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.

Module Test Plan

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.

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 Date Complete 11/25/92

Module Test Plan

() 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 A Date Complete 11/29/92

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: TYPE_COMMENT_STATUS_CODE
 Title: Comment Status Code

Block Description

Block Test Case

Block: TYPE_COMMENT_STATUS_CODE
 Title: Comment Status Code

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 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.

Block: TYPE_COMMENT_STATUS_CODE
Title: Comment Status Code

Block Test Results Expected

- (✓) 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: COMMENT_STATUS

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.

Block: TYPE_COMMENT_STATUS_CODE
Title: Comment Status Code

Field Name: COMMENT_STATUS

Field Test Plan

- (✓) 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.
- (✓) 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.

Block: TYPE_COMMENT_STATUS_CODE
Title: Comment Status Code

Field Name: COMMENT_STATUS

Field Test Case

- (✓) 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.
 - () 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

Block: TYPE_COMMENT_STATUS_CODE
Title: Comment Status Code

Field Name: COMMENT_STATUS

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.
- (✓) 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: TYPE_COMMENT_STATUS_CODE
Title: Comment Status Code

Field Name: DESCR_STATUS

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/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.
- (✓) 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.

Block: TYPE_COMMENT_STATUS_CODE
Title: Comment Status Code

Field Name: DESCR_STATUS

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.

Block: TYPE_COMMENT_STATUS_CODE
Title: Comment Status Code

Field Name: DESCR_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.
- (✓) 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.

3.0 LIBRARY

3.1 LI1002U1 MAINTAIN SITE CODE TABLE

LEMIS Module Test Checklist Report

Name: MAINTAIN SITE CODE TABLE
 Short Name: LI1002U1
 Purpose: Maintain the site code table values

Initial Test Date: 12/16/92 LC

Verification Date: 12/16/92 LC

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

Block: TYPE_SITE_CODE
 Title: Site Type Code

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

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

Module Description

This table supports the update and retrieval functions for the type_site_code table. The table identifies the valid codes that may be associated with locations on the Hanford Site. The codes define the nature of the site and the structures/features found there.

Display 12 lines of site code entry. Only the site code and description will be displayed.

Module Help

This screen supports the update and retrieval functions for the site code table. This table maintains a list of valid codes that describe the various waste sites on Hanford.

To store a new site code:

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

If all of the spaces are filled and more site codes 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 codes will scroll upward, leaving space for an additional code to be entered.

2. Enter the appropriate site 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 site 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. Use the down arrow key to move the cursor to the code entry 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 12/16/90

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 San Date Complete 12/12/92
- () 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 Plan

System Architect ADate Complete 12/16/92

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

Should be on at end

- (✓) 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: TYPE_SITE_CODE
Title: Site Type Code

Block Description

Block Test Case

Block Test Results Expected

Field Name: SITE_TYPE

Field Description

This code is derived from WIDS SiteType and describes the structural

Block: TYPE_SITE_CODE
Title: Site Type Code

Field Name: SITE_TYPE

Field Description

type of the waste site.

Field Test Case

- (✓) Clear an existing field and attempt to update the record.
- (✓) 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.
- () 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/F7 [duplicate field] key

→ PLUG
LOV not avail for field

Block: TYPE_SITE_CODE
Title: Site Type Code

Field Name: SITE_TYPE

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.
- () 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.
- () Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
- () Begin field entry with a space. An error should occur.

Block: TYPE_SITE_CODE
Title: Site Type Code

Field Name: DESCRIPTION

Field Description

Field Test Case

- 1722
- () 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.

1707
null
field

Field Test Results Expected

- 1722
- () 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_SITE_CODE
Title: Site Type Code

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).
 - () 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.
- () 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 that the field data must be entered.

not null field

Block: TYPE_SITE_CODE
Title: Site Type Code

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.~~

3.2 LI1003U1 MAINTAIN PRECIPITATION CODES

LEMIS Module Test Checklist Report

Name: MAINTAIN PRECIPITATION CODES
 Short Name: LI1003U1
 Purpose: Maintain the precipitation type code table
 Initial Test Date: 12/16/92 *LC*
 Verification Date: 12/16/92 *LC*

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

Block: TYPE_PRECIP_CODE
 Title: Types of Precipitation

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

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

Module Description

This module supports the maintenance and retrieval of the type-precip-code table. This table maintains a list of valid types of precipitation.

Display 12 lines of precipitation codes. Only the precipitation code and description are displayed.

Module Help

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

To store a new precipitation code:

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

If all of the spaces are filled and more precipitation 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 precipitation 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

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 precipitation 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 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

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 PC Date Complete 12/16/92

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 Shy Date Complete 12/14/92
- () 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 Plan

System Architect ADate Complete 12/16/92

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
 - (✓) 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: TYPE_PRECIP_CODE
 Title: Types of Precipitation

Block Description

Block Test Case

Block Test Results Expected

Field Name: TYPE_PRECIPITATION

Field Description

Block: TYPE_PRECIP_CODE
Title: Types of Precipitation

Field Name: TYPE_PRECIPITATION

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.

non null 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: TYPE_PRECIP_CODE
Title: Types of Precipitation

Field Name: TYPE_PRECIPITATION

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
- 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 () Bypass the field by using the TAB key.
 - () 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.
- n/a () 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 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.

non null field

Block: TYPE_PRECIP_CODE
Title: Types of Precipitation

Field Name: TYPE_PRECIPITATION

Field Test Results Expected

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

Block: TYPE_PRECIP_CODE
 Title: Types of Precipitation

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. *non 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.
- () 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

Block: TYPE_PRECIP_CODE
 Title: Types of Precipitation

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).
 - () Value above valid range should display message COM00101 or
- (✓) 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.
- () All data fields are correctly stored in the database.
- N/A () Bypass the field by using the TAB key.
- () 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.
- N/A () 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 that the field data must be entered.

Block: TYPE_PRECIP_CODE
Title: Types of Precipitation

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.

3.3 LI1005U1 MAINTAIN LOCATION LEVEL CODE

LEMIS Module Test Checklist Report

Name: MAINTAIN LOCATION LEVEL CODE
 Short Name: LI1005U1
 Purpose: Maintain location level code table data

Initial Test Date: 12/16/92 *RC*

Verification Date: 12/17/92 *RC*

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

Block: TYPE_LOCATION_LEVEL_CODE
 Title: Location Levels

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

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

Module Description

This module provides update and retrieval display support for the type-location-level-code table. The code table identifies the valid location levels associated with the database. These levels define a hierarchy of locations from the planet Earth down to a interior location.

Display 12 lines of data. The level code and description are the only fields displayed.

Module Help

This transaction supports the maintenance and retrieval of location level codes. Location level codes are used to identify the hierarchical levels at which a location exists within an overall scheme. At the top (level 0) is the world. The level number values ascend as we move toward a more finite and specific spot on the face of the planet.

Location levels are used to support reporting. Standard reports are executed to draw all data from a particular level and all subordinate levels. It is important to note that changing a level value can have severe side effects on the overall location reporting. Review any planned changes with the system manager prior to executing the change.

To store a new location level code:

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

If all of the spaces are filled and more level 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 location level 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 this code value may affect the operation of software elsewhere in the system. See the system manager before making any changes to the code value.
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

Deletion of location level codes is not permitted.

Module Test Plan

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.

- () 8 - Secondary testing complete, no problems found.

Developer _____ Date Complete _____
 Tester Date Complete 12/17/92

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.

- () 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 12/17/92
- () 20 - The transaction is placed into the production environment.
 - () Software files in production directories.

Module Test Plan

- () DMR 550 updated to add transaction.
 - () DMR 900 updated to add transaction.
 - () Transaction added to production usage menu.
- System Architect JA Date Complete 12/17/92

Module Test Case

- 723
- (✓) 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

*Trans. help
executing
misspelled*

- (/ 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: TYPE_LOCATION_LEVEL_CODE
Title: Location Levels

Block Description

Block Test Case

Block Test Results Expected

Field Name: LEVEL_LOCATION

Block: TYPE_LOCATION_LEVEL_CODE
 Title: Location Levels

Field Name: LEVEL_LOCATION

Field Description

Field Test Case

- (✓) Clear an existing field and attempt to update the record.
- (✓) 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.
- 724 () 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 ^{FO}
- 11A () Store a series of records using the SHIFT/^{FO}F7 [duplicate field] key to carry forward field values from the previously stored record.

Block: TYPE_LOCATION_LEVEL_CODE
Title: Location Levels

Field Name: LEVEL_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.

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

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

(✓) 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.

() Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.

Numbers only allowed

(✓) Begin field entry with a space. An error should occur.

OK

Block: TYPE_LOCATION_LEVEL_CODE
Title: Location Levels

Field Name: DESCRIPTION

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 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

Block: TYPE_LOCATION_LEVEL_CODE
Title: Location Levels

Field Name: DESCRIPTION

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.
- (✓) 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: LIBR Library functions

Transaction: LI1005U1 Maintain Location Level Codes

```
=====
Comment ID : 723   Priority: LOW           Date: 16-DEC-92
=====
Status      : ASSIGNED   Date: 16-DEC-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : LORRAINE CAMPBELL
System ID   : 10000000771   Oracle ROWID: 00000049.0009.0018
Last Error  : -0
Last Msg    : FRM-40352 Last row of query retrieved.
Table       : TYPE_LOCATION_LEVEL_CODE
Block       : LOCLEV
Field       : LOCLEV.LEVEL_LOCATION
=====
```

DESCRIPTION

spelling error for executing. spelled "executving"

FIXED

12/16

OK
L

3.4 LI1010U1 MAINTAIN LIBRARY STATUS CODE

LEMIS Module Test Checklist Report

Name: MAINTAIN LIBRARY STATUS CODE
 Short Name: LI1010U1
 Purpose: Maintain the library status codes.

Initial Test Date: 11/30/92 JK

Verification Date: 12/1/92 RSA

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

Block: TYPE_STATUS
 Title: 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 status codes.

Display 12 lines of data. The status code and description are the only data items displayed.

Module Help

This module supports the maintenance and retrieval of library status codes. This table maintains a list of valid codes that describe the

Module Help

types of library status that may occur.

To store a new library 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 library 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 library 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 library 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:

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> </u>	Date Complete <u>12/1/92</u>

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 ZZ 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 CA 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.
- 634 (✓) Transaction help describes:
 - (✓) Purpose of the transaction
 - (✓) Functions performed by the transaction
 - N/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 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 STATUS
Title: 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.
- () Transaction help describes:

Duplicate tests

Block: TYPE STATUS
Title: Status Types

Block Test Results Expected

- () 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.

duplicate tests

Block: TYPE_STATUS
Title: Status Types

Block Test Results Expected

- () 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

Field Test Case

- (✓) 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.
- (✓) 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

Block: TYPE_STATUS
 Title: Status Types

Field Name: TYPE_STATUS

Field Test Results Expected

- 635 (✓) 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
 (✓) 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

(must enter data in description field too)

Block: TYPE_STATUS
Title: Status Types

Field Name: TYPE_STATUS

Field Test Results Expected

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

() 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

(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

(✓) 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.

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

Block: TYPE_STATUS
Title: Status Types

Field Name: TYPE_STATUS

Field Test Results Expected

record from the database.

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

NA () 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*

(✓) 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_STATUS
Title: 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.
 - (✗) 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.

Nulls not allowed

11

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 STATUS
 Title: 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 COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT/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.
- 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.
- (✓) 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

} NO no blanks allowed

11

Block: TYPE_STATUS
Title: 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.

Transaction: LI1010U1 Maintain Library Status Codes

```
=====
Comment ID : 635   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   : 10000001231   Oracle ROWID: 00000062.0003.0018
Last Error  : -0
Last Msg    : FRM-40400 Transaction complete -- 1 records posted an
Table       : TYPE_STATUS
Block       : TYPEVS
Field       : TYPEVS.TYPE_STATUS
=====
```

DESCRIPTION

Help for block, update section: item 4 is truncated.

FIXED.

12/1/92

✓

3.5 LI4041U1 LOCATION MAINTENANCE

LEMIS Module Test Checklist Report

Name: LOCATION MAINTENANCE
 Short Name: LI4041U1
 Purpose: Maintain location data.

Initial Test Date: 12/15/92 *RC*

Verification Date: 12/17/92 *RC*

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

Block: LOCATION
 Title: 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	CHAR	2	()	()	()	()	()

Block: LOCATION
Title: 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	()	()	()	()	()
TYPE_LOCATION	CHAR	1	()	()	()	()	()
LEVEL_LOCATION	CHAR	2	()	()	()	()	()
HANFORD_COORD	CHAR	15	()	()	()	()	()
TYPE_SITE	NUMBER	3	()	()	()	()	()
LAMBERT_EW_COORD	NUMBER	11	()	()	()	()	()
LAMBERT_NS_COORD	NUMBER	11	()	()	()	()	()
FLAG_TEXT	CHAR	1	()	()	()	()	()

Block: LOCATION_NEST
Title: Locations Associated With Another Locati

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

Block: LOCATION_ORGAN_J
Title: Organizations Related to a Location

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

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.
 Procedures may be attached to the location.
 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

Module Test Plan

Module Test Case

no testing available
no module help.

Block: LOCATION
 Title: Location

Block Description

Block Test Case

Block Test Results Expected

Field Name: IDENT_LOCATION

Field Description

The field contains the unique identifier of a location, the user viewed name.

Field Test Case

(/) Begin field with a space.

Block: LOCATION
Title: Location

Field Name: IDENT_LOCATION

Field Test Case

- (✓) 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
- NA () 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.

Block: LOCATION
Title: Location

Field Name: IDENT_LOCATION

Field Test Results Expected

- () 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 enter data as lower case. ~~An error should result.~~
- (✓) 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: LOCATION
Title: Location

Field Name: TITLE_LOCATION

Field Description

This field is a 1-40 character descriptive title of the location.

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

Block: LOCATION
Title: Location

Field Name: TITLE_LOCATION

Field Test Results Expected

- 66*
- (✓) 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.
 - (✓) 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: LOCATION
Title: Location

Field Name: TYPE_LOCATION

Field Description

This field identifies the type of location. Valid codes are:

W - waste location
D - document location
M - management offices
P - physical location

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.
- (/) 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: LOCATION
Title: Location

Field Name: TYPE_LOCATION

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/^{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.
- ☒ 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: LOCATION
Title: Location

Field Name: LEVEL_LOCATION

Field Description

This field is validated by LOCATION_LEVEL_CODE. The location level ranges from 0 - the world to 9 - specific site location.

Field Test Case

- N/A* () Clear an existing field and attempt to update the record.
- () 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.
- N/A* () Attempt to enter negative value. No error should occur.

no desc for test

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.
- N/A* () 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: LOCATION
Title: Location

Field Name: LEVEL_LOCATION

Field Test Results Expected

- 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.
- 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 () Delete an existing value.
- () 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.
- ✓ () Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.
- ✓ () Begin field entry with a space. An error should occur.

*Non
Updatable
Field*

Block: LOCATION
Title: Location

Field Name: HANFORD_COORD

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.
- 7/2 () 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: LOCATION
Title: Location

Field Name: HANFORD_COORD

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/⁸⁶~~87~~ [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.
- (✓) Any alphanumeric characters may be entered into the field.
- (✓) Attempt to enter data as lower case. ~~An error should result.~~

Block: LOCATION
Title: Location

Field Name: TYPE_SITE

Field Description

This field uses a LOV derived from the WIDS system.

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.

1707
Null
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.
- 113 () 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: LOCATION
Title: Location

Field Name: TYPE_SITE

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~~^{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.
- ☒ 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. *non null*
- ☒ Delete an existing value. The field should be cleared and stored as a null value. *non null*
- ☒ Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted. *allows change after then col*
- ☒ Begin field entry with a space. ~~An error should occur.~~

Block: LOCATION
Title: Location

Field Name: LAMBERT_EW_COORD

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.
- ☒ 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

Block: LOCATION
Title: Location

Field Name: LAMBERT_EW_COORD

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 COM00103 (if no LOV).
 - () Value above valid range should display message COM00101 or
- (✓) Store a series of records using the SHIFT ^{fg} [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 dataout 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.
- (✓) Numeric data (0-9), comma (,), plus (+), minus (-), and period (.) are permitted.

does not accept comma

Block: LOCATION
Title: Location

Field Name: LAMBERT_EW_COORD

Field Test Results Expected

(6) Begin field entry with a space. An error should occur.

Block: LOCATION
Title: Location

Field Name: LAMBERT_NS_COORD

Field Description

A dated, written communication to a customer, providing price or prices, for product or products, possibly specifying the minimum volume, and necessarily specifying an expiry date

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.
- NA* (☒) Override the default with an invalid data value.
- 7/8* (☒) 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.

NA

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

Needs description

Block: LOCATION
Title: Location

Field Name: LAMBERT_NS_COORD

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/^{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.
- ☐ 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: LOCATION
Title: Location

Field Name: LAMBERT_NS_COORD

Field Test Results Expected

11/ () Numeric data (0-9), comma (,), plus (+), minus (-), and period (.)
are permitted.

(~~/~~) Begin field entry with a space. ~~An error should occur.~~

Block: LOCATION
 Title: Location

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: LOCATION
Title: Location

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
- () 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.

Block: LOCATION
Title: Location

Field Name: FLAG_TEXT

Field Test Results Expected

- () 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: LOCATION_NEST
Title: Locations Associated With Another Locati

Block Description

Block Test Case

Block Test Results Expected

no testing

Block: LOCATION_ORGAN_J
Title: Organizations Related to a Location

Block Description

This table provides a link between organization and location data. An organization may be the manager of a specific location, usually a building or area but may extend down to a waste stream.

An occurrence of this table identifies the association between organization and defined location.

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.
- N/A () 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.
- 720 () 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: LOCATION_ORGAN_J
Title: Organizations Related to a Location

Field Name: RELATIONSHIP

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/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.

*non null
not
updatable*

() 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/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.

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

() Modify each default value with an invalid alternative.

Block: LOCATION_ORGAN_J
Title: Organizations Related to a Location

Field Name: RELATIONSHIP

Field Test Results Expected

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

() Attempt to enter data as lower case. An error should result. *Def*

(✓) 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: LI4041U1 Location Maintenance

```
=====
Comment ID : 706   Priority: LOW           Date: 15-DEC-92
=====
Status      : ASSIGNED   Date: 16-DEC-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : LORRAINE CAMPBELL
System ID   : 10000001156   Oracle ROWID: 00000021.0011.0018
Last Error  : -0
Last Msg    : FRM-40350   Query caused no records to be retrieved.
Table       : LOCATION
Block       : LOCATE
Field       : LOCATE.IDENT_LOCATION
=====
```

DESCRIPTION

no module testing available.

NOT PROB

12/16

TEST BOOK PROBLEM

Programmer's Bug Report

Application: LIBR Library functions

Transaction: LI4041U1 Location Maintenance

=====

Comment ID : 707 Priority: LOW Date: 15-DEC-92

=====

Status : ASSIGNED Date: 16-DEC-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : LORRAINE CAMPBELL

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40102 Record must be entered or deleted first.

Table : LOCATION

Block : LOCATE

Field : LOCATE.LEVEL_LOCATION

DESCRIPTION

there are no descriptions with the LOV.FIXED ✓
12/15OK
L

Application: LIBR Library functions

Transaction: LI4041U1 Location Maintenance

=====

Comment ID : 708 Priority: LOW Date: 15-DEC-92

=====

Status : ASSIGNED Date: 16-DEC-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By : LORRAINE CAMPBELL

System ID : Oracle ROWID:

Last Error : -0

Last Msg : FRM-40739 Clear_Form with Full_Rollback not allowed i

Table : LOCATION

Block : LOCATE

Field : LOCATE.TYPE_SITE

DESCRIPTION

no descriptions for LOV

FIXED 12/15 ✓
OK fa

Programmer's Bug Report

Application: LIBR Library functions

Transaction: LI4041U1 Location Maintenance

```
=====
Comment ID : 712   Priority: LOW           Date: 15-DEC-92
=====
Status      : ASSIGNED   Date: 16-DEC-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : LORRAINE CAMPBELL
System ID   : 10000004390   Oracle ROWID: 0000011D.000A.0018
Last Error  : -0
Last Msg    : FRM-40350   Query caused no records to be retrieved.
Table       : LOCATION
Block       : LOCATE
Field       : LOCATE.HANFORD_COORD
=====
```

DESCRIPTION

an example for the Hanford Coordinates would be helpful here.

✓ NOT PROBLEM 12/16 ✓
SUFFICIENT DESCRIPTION
FOR OPTIONAL FIELD

16-DEC-92

WHC-SD-WM-TP-164 REV 0
Programmer's Bug Report

Page 1 of 1

Application: LIBR Library functions

Transaction: LI4041U1 Location Maintenance

=====
Comment ID : 715 Priority: LOW Date: 15-DEC-92
=====

Status : ASSIGNED Date: 16-DEC-92 Class:
Type : PROBLEM
Assigned To: TODD ADAMS
Entered By : LORRAINE CAMPBELL
System ID : 10000004390 Oracle ROWID: 0000011D.000A.0018
Last Error : -0
Last Msg : FRM-50016 Legal characters are 0-9 - + E .
Table : LOCATION
Block : LOCATE
Field : LOCATE.TYPE_SITE

DESCRIPTION

=====
FIELD IS A LOV FIELD. FIELD SHOULD NOT ALLOW MINUS, COMMA, PERIOD OR PLUS TO BE ENTERED AND ACCEPTED WHEN TABBING TO NEXT FIELD. The only one that accepted was the Plus

FIXED

12/16

help
OK

Application: LIBR Library functions

Transaction: LI4041U1 Location Maintenance

=====

Comment ID : 718 Priority: LOW Date: 15-DEC-92

=====

Status : ASSIGNED Date: 16-DEC-92 Class:

Type : PROBLEM

Assigned To: TODD ADAMS

Entered By: LORRAINE CAMPBELL

System ID : 10000004390 Oracle ROWID: 0000011D.000A.0018

Last Error : -0

Last Msg : FRM-50016 Legal characters are 0-9 - + E .

Table : LOCATION

Block : LOCATE

Field : LOCATE.LAMBERT_NS_COORD

DESCRIPTION

lower case letters accepted. Comma not accepted

NOTPROB (NUMERIC FIELD)

12/16

Programmer's Bug Report

Application: LIBR Library functions

Transaction: LI4041U1 Location Maintenance

```
=====
Comment ID : 719   Priority: LOW           Date: 15-DEC-92
=====
Status      : ASSIGNED   Date: 16-DEC-92   Class:
Type        : PROBLEM
Assigned To : TODD ADAMS
Entered By  : LORRAINE CAMPBELL
System ID   : 10000004390   Oracle ROWID: 0000011D.000A.0018
Last Error  : -0
Last Msg    : FRM-40407   Transaction complete -- posted records comm
Table       : LOCATION
Block       : LOCATE
Field       : LOCATE.LAMBERT_NS_COORD
=====
```

DESCRIPTION

needs description help

FIXED

12/16

✓

OK

This page intentionally left blank.