TR A4025.02

ELCBAS 90
Factory Acceptance Test Procedures

Nils Eggen, Tormod Lund, Birger Stene

www.sintef.no

SINTEF Energi AS
SINTEF Energy Research

October 2004

SINTEF

TECHNICAL REPORT

SINTEF

SINTEF Energy Research

Address:	NO-7465 Trondheim, NORWAY
Reception:	Sem Sælands vei 11
Telephone:	+47 73 59 72 00
Telefax:	+47 73 59 72 50

www.energy.sintef.no

Enterprise No.:
NO 939 350 675 MVA

SUBJECT/TASK (title)

ELCBAS 90
Factory Acceptance Test Procedures

CONTRIBUTOR(S)

Nils Eggen, Powel ASA , Tormod Lund, ABB,
Birger Stene, SINTEF Energy Research

CLIENTS(S)

Statnett SF

TR NO.	DATE	CLIENT'S REF.	PROJECT NO.
TR A4025.02	2004-10-04	Anders Larsen	11X051

ELECTRONIC FILE CODE	RESPONSIBLE (NAME, SIGN.)	CLASSIFICATION
040819bs102720	Birger Stene [illegible]	Unrestricted

ISBN NO.	REPORT TYPE	RESEARCH DIRECTOR (NAME, SIGN)	COPIES	PAGES
82-594-2753-2		Petter Støa [illegible]	10	47

DIVISION	LOCATION	LOCAL FAX
Energy Systems	Sem Sælandsv. 11	+47 73 59 72 50

RESULT (summary)

This document describes the Factory Acceptance Test (FAT) procedures for the ELCBAS-90 System.

This .02 version is an updated version of the .01 version and include revisions after Windows FAT.

KEYWORDS

SELECTED BY AUTHOR(S)	ELCOM	FAT
	ELCBAS	Communication Protocol

Document Information

Document identity:	Elcbas_FAT_Procedures.doc
Revision:	2

Revision History

Date	Rev.	Notes
950814	0	New Document
020719	1	Common Unix and Windows FAT procedures for class 3
040722	2	Revision after Windows FAT

TABLE OF CONTENTS

Preface

This document describes the test procedures to be used at the Factory Acceptance Test (FAT) of the ELCBAS-90 software to verify conformance with the requirements given in "Software requirements specification of the ELCBAS-90 system" [1]. The tests are also designed to test conformance with the ELCOM-90 User Element Conventions (refer [2]).

The test procedures can be used to test ELCBAS-90 Version 1 Class 2, and ELCBAS-90 Version 1 class 3. All tests shall not be used for ELCBAS-90 Version 1 Class 2.

The test procedures can be used for both Windows and Unix implementation of ELCBAS-90. All tests are not relevant for Windows implementation.

Each test case comprises of four parts:

- A description of the purpose of the test case.
- A definition of the criteria for approving the test.
- A detailed step-by-step description of the test procedure.
- A test protocol.

The test procedures refers to functions rather than describing the exact menu selections in the ELCBAS-90 GUI (Graphical User Interface).

The test protocol form is enclosed in Appendix A.

Related Documents

[1] EFI TR F3911: Software requirement specification of the ELCBAS-90 system.

[2] EFI TR A3825: ELCOM-90 User Element Conventions.

1 GENERAL DESCRIPTION

1.1 Test System

1.1.1 Hardware Configuration
The hardware configurations used during the ELCBAS-90 Factory Acceptance Test shall be specified in the supplied tables in Appendix A.

1.1.2 Software Configuration
The software components under test are listed below. Additional information shall be specified in the supplied tables in Appendix A.

Configuration

System A:	ELCOM-90 Reference Version or compatible ELCBAS-90 System
System B:	ELCOM-90 Reference Version or compatible ELCBAS-90 System

The system under test is <u>ELCBAS-90 system</u> comprising:

ELCBAS-90	Supervisor (GUI)	
ELCBAS-90	Initiator	
ELCBAS-90	Responder	
ELCBAS-90	Watchdog	(Only Unix)
ELCBAS-90	Purger	(Only Unix)
ELCBAS-90	Shutdown	(Only Unix)
ELCBAS-90	Database	(Only Unix)

For the test of ELCBAS-90 the following 4 components shall be used:

Initiator type of Application Server	("iccini"/"t_in_data")
Responder type of Application Server	("iccres"/"t_rs_data"/"t_rs_uns")
Command/Setpoint sending Application Server	("cssend"/"t_in_cmd")
Command/Setpoint receiving Application Server	("csrecv"/"t_rs_cms")

The Applications Servers are used for:

a) testing functions in ELCBAS-90 related to Application Servers.
b) simulating data input to Responder from a Responder Application Server.
c) producing output on the screen of data transferred to an Initiator Application Server.
d) Sending commands and setpoints to the Initiator, and receive results.
e) Receiving commands and setpoints from the Responder, and give adequate answers.

Note that the Application Servers used during the test do not need to access any Application Database.

The ELCBAS-90 API should be tested separately.

These test procedures were originally written for the Unix version of Elcbas-90. All tests may not be relevant for the Windows version of Elcbas-90. I.e. the value type “Text” is not supported in the Windows version. Such cases shall be documented in Appendix A during the test.

In the following it is referred to the use of EQL will be used to verify the contents of the ELCBAS-90 database. For the Unix version, sqlplus is used for Oracle, isql for Sybase. Other means can be used for the same purpose, depending on the software configuration used under test. The Windows implementation of Elcbas-90 does not use a database for data and configuration parameters. For the Windows version other tools can be used to inspect the configuration and results. This information shall be supplied in Appendix A.

1.1.3 Test Data

The test data to be used in the FAT is defined in Appendix B. The data shall be defined before the FAT starts. Some of the groups will have the necessary number of objects to test the extended functionality in ELCOM-90 for support of "large" groups.

Reference data set PC-BC: 8 groups with 30 objects each (group number 21-28)

Reference data set BC-BC: 4 groups with 10 objects each (group number 31-34)

The data sets shall be defined such that all ELCOM datatypes for the ELCOM Class implemented are tested:

* Floating point values	
* Integer values	
* Status values	
* Logical breaker status	
* Binary command	Class 3 only
* Analogue setpoint	Class 3 only
* Digital setpoint	Class 3 only
* Text strings	

One ordinary and one large group shall contain data for 24 hours (hour values).

One partner shall be pre-defined.

For the simultaneity test (refer chapter 4.8) data for 8 partners shall be defined according to the reference data set.

For the compatibility test with ELCBAS-83 only groups containing data types supported in ELCOM Version 0, class 2 shall be used.

The state of the ELCBAS configuration when the FAT starts, is defined in Appendix B.

1.2 Functional Requirements

All functions defined in the ELCBAS-90 Software Requirement Specification ([1]) shall be verified.

For all tests, ELCBAS-90 Version 1, Class 3 or ELCBAS-90 Version 1, Class 2 shall be used. The tests will also include the new functional unit "Initiator Data Transfer".

The functional tests are given in a sequence which will assure the least amount of work defining data during the test.

1.3 Suspension criteria and resumption requirements

Two cases for suspension criteria and resumption requirements are defined for the FAT of the ELCBAS-90 System:

1) Unexpected result detected in a function which has impact on further testing
2) nexpected result detected in a function which does not have any impact on further testing.

The two cases will be handled as follows:

Case 1: Suspend all test activities until detected error is found and corrected.

Case 2: The test activities can continue, but the test which caused the error is rejected. The test protocols shall describe the outcome of all tests.

2 GENERAL REQUIREMENTS

General requirements for the ELCBAS-90 system are defined in the following sub chapters. A test protocol shall be made for each of the following sub chapters defining tests not specified explicitly.

2.1 Security

ELCOM-90 Security mechanisms are an option not implemented in ELCBAS-90 (refer [2]).

2.2 Robustness

Robustness shall be tested by using illegal values for input parameters. The extent of the parameters in the ELCBAS-90 Supervisor limits the possibility for performing a complete robustness test covering all parameters. As a result robustness shall be tested for a selected number of Supervisor functions.

2.3 Reliability

Test of reliability for the ELCBAS-90 system is defined implicitly in the functional tests defined in chapter 3.

2.4 Performance

Refer chapter 5.

2.5 Recovery

Recovery will be tested by halting the machine (simulation of power failure) and restart the machine. The simulated break-down situation shall remain for 24 hours before the machine is restarted. There are no requirements concerning the effort spent on recovering from a break-down situation. However, the two general conditions apply:

a) If the system boot files for the machine contains proper information to restart the software under test, the ELCBAS-90 system shall be available to the user without any manual actions.
b) The configuration shall be consistent.

When the ELCBAS-90 software restarts, the following shall be verified:

1. All permanent associations are re-established.
2. For periodically requested data, data from the shut-down period shall be collected.

2.6 Help function

A test of the Help functions is not explicitly defined in the test procedures. The Help functions shall, however, be verified during the FAT as part of the specified procedures. The following Help functions shall be tested:

a. Help for each menu and menu item.
b. Help for each dialogue window.
c. Help on all parameters supplied by the user.
d. Information about program versions.

3 FUNCTIONAL TESTS

3.1 Start Initiator

Purpose:

To test that the Initiator can be started by this function.
The function is verified by inspecting the Initiator trace.

Test procedure:

1. Select the function Initiator *Start*.
2. Start all ELCBAS-90 Initiator Functions.
3. Inspect the trace-information and verify that the Initiator starts.
4. Accept/reject test.

3.2 Start Responder

Purpose:

To test that the Responder can be started by this function.
The function is verified by inspecting the Responder trace.

Test procedure:

1. Select the function Responder *Start*.
2. Start all ELCBAS-90 Responder Functions.
3. Inspect the trace-information and verify that the Responder starts.
4. Accept/reject test.

3.3 Create / Modify Partner

Purpose:

To verify that a partner can be defined according to specified characteristics.
The function is verified by checking the configuration with the entered parameters.

Test procedure:

1. Select the function Partner *Create*
2. Define a new partner X25PARTNER of Version 1 Class 2 with three X.25 addresses.
3. Use EQL to verify that the partner is configured.
4. Define another partner TCPPARTNER of Version 1 Class 2 with the same X.25 address as one of the three defined for X25PARTNER..
5. Verify that all three addresses specified for X25PARTNER are rejected.
6. Redefine the partner with three alternative TCP/IP addresses.
7. Use EQL to verify that the partner is configured.
8. Define a third partner with the same TCP/IP address as one of those defined for TCPPARTNER.
9. Verify that all three addresses defined for TCPPARTNER are rejected.
10. Accept/reject test.

3.4 Create/Modify Object

Purpose:

Verify that an object can be defined and changed.
The database contents shall be verified by means of EQL.

Test procedure:

1. Select the function Object *Create*
2. Define 170 objects according to group configuration in Appendix B.
3. Check that the object information is stored correctly in the database by means of EQL.
4. Modify *Object Type* and *Object Description* for one of the objects.
5. Verify by means of EQL that the database is updated with the changes.
6. Accept/reject test.

3.5 Create Group

Purpose:

Ref [2], chapter 6. It shall be verified that one group can be created both in the local and remote database with the necessary parameters.

The entered group information shall be verified by comparing the database contents. Data transfer shall be verified by means of the Responder-log.

The group definition shall be performed on the Initiator side.

Test procedure:

1. Turn Responder-log On.
2. Select the function Group *Define*.
3. Define group 1 as a Global group and group 2 as a Local group
4. Verify by means of the Responder-log that group 1 is transferred to the Responder.
5. Use EQL to check that the group 1 is defined in both the local and remote databases and that group 2 is defined in the local database.
6. Define group 10 as a Global group.
7. Verify by means of the Responder-log that group 10 is transferred to the Responder.
8. Use EQL to check that group 10 is defined in both the local and remote databases.
9. Define group 3, 4, 5, 6, 7 and 8 according to Appendix B as Global groups.
10. Use the Responder-log to verify that the groups are transferred to the Responder.
11. Use EQL to verify the contents of the local and remote databases.
12. Accept/reject test.

3.6 Access to Objects

Purpose:

To test that it is possible to make, modify and delete access control lists for objects related to a specific partner.

Also, it shall be tested that deletion of access to an object already assigned to a Responder-group shall produce a warning.

Test procedure:

1. Define Objects T62 and T63 on the A- and B-side.
2. Define Group 6 consisting of objects T62 and T63 (global)
3. Verify that the group definition results in an error from the remote Responder.
4. Define access to T62 and T63 on B-side
5. Redefine Group 6.
6. Verify that the group definition was successful.
7. Delete access to Object T62 on the B-side.
8. Verify that the deletion produces a warning message.
9. Choose to ignore the warning.
10. Verify by means of EQL that the object access and group definition is deleted.
11. Accept/reject test.

3.7 Inspect Group

Purpose:

Ref [2], chapter 6. It shall be verified that a defined group can be read and presented both from the local database, and from a remote database.

The information presented on the screen shall be compared with the information in the databases. Data transfer shall be verified using Responder-log.

Test procedure:

1. Select function Group *Read*
2. Select group 10.
3. Verify by means of the Responder-log that the data is transferred from the Responder.
4. Verify by means of EQL that the presented information conforms with the Responder database contents.
5. Select function Group *Read*
6. Select group 2.
7. Verify that the selection results in an error (group 2 is a local group).
8. Accept/reject the test.

3.8 Modify Group

Purpose:

Ref [2], chapter 6. It shall be verified that group information can be modified in both the local and remote database.

The entered group information shall be verified by comparing the database contents. Data transfer shall be verified by means of the Responder-log.

The group modification shall be preformed on the Initiator side.

Test procedure:

1. Select the function Group *Modify*.
2. Modify group 2 to become a Global group
3. Verify by means of the Responder-log that group 2 is transferred to the Responder.
4. Check with EQL that the remote database is updated for group 2.
5. Try to modify the parameters for group 2 with illegal values:
   Define global group with *Owner*=Responder
   Change global group where *Static*=true
6. Accept/reject the test.

3.9 Log

This function is used to turn on and off logging facilities in the Initiator and Responder. The mode of the log-system is set to the same as before the log was activated. The information from the log is ASCII text, and can be inspected using an editor.

Purpose:

It shall be controlled that the log for the Initiator and Responder may be turned off and on, and that the log contains information about executed functions in the Initiator and Responder.

Test procedure:

1. Select the function Log
2. Turn both the Initiator and the Responder Logs Off in Systems A and -B to reset the log files.
3. Turn the logs on again.
4. Perform the function Read Group, group 1, Global.
5. Turn the logs off
6. Verify that the Initiator-log contains information about the Group definition, and that the Responder-log is empty in System A.
7. Verify that the Responder-log contains information about the Group definition, and that the Initiator-log is empty in System B.
8. Accept/reject test.

3.10 Stop Initiator

Purpose:

To test that Initiator can be stopped by means of this function and that all active functions are stopped before the Initiator process is halted.

The function is verified by inspecting the Initiator-log and trace.

Test procedure:

1. Select function Stop Initiator
2. Stop all ELCBAS-90 functions
3. Inspect the Initiator-log and trace, and verify that all started ELCBAS functions are stopped before the Initiator process is halted.
5. Accept/reject test

3.11 Stop Responder

Purpose:

To test that Responder can be stopped by means of this function and that all active functions are stopped before the Responder process is halted.

The function is verified by inspecting the Responder-log and trace.

Test procedure:

1. Select function Stop Responder
2. Stop all ELCBAS-90 functions
3. Inspect the Responder-log and trace, and verify that all started ELCBAS functions are stopped before the Responder process is halted.
4. Accept/reject test

3.12 Requested Data Transfer (Fetch Data)

Purpose:

Ref [2], chapter 7.1. To verify that transfer of object values for a selected group can be requested by the Initiator.

EQL shall be used to verify the contents of the database, and Initiator-log shall be used to check that data transfer is performed for the selected group.

The values for the selected objects must be different in the Initiator and Responder Databases before the test is started.

Test procedure:

1. Turn Initiator-log On.
2. Define Application Server for Initiator.
3. Specify Requested Data Transfer for group 1, with transfer of 6 sets of time values.
4. Verify that data transfer is performed for the selected group and time value sets (Initiator-log).
5. Verify that the values presented on the screen correspond to the values fetched from the Responder database.
6. Verify that the Initiator type Application Server is activated for the transferred data.
7. Modify some of the quality codes for the data in group 1.
8. Retransmit group 1 by repeating item 3, and verify the transfer by repeating items 4 through 6.
9. Repeat items 3 through 6 for group 10 (large group).
10. Repeat items 3 through 6 for group 2 (Status values).
11. Repeat items 3 through 6 for group 3 (Integer values).
12. Repeat items 3 through 6 for group 4 (Logical Breaker status values).
13. Repeat items 3 through 6 for group 5 (Text messages).
14. Turn Initiator-log Off.
15. Accept/reject the test.

3.13 Delete Object

Purpose:

Verify that an object can be deleted.

The database contents shall be verified by means of EQL.

1. Define an object T60 as type text message.
2. Try to delete an object assigned to a group (T39).
3. Verify that the deletion is rejected.
4. Delete on object which is not assigned to a group (T60).
5. Verify by means of EQL that the object is deleted in the database.
6. Accept/reject test

3.14 Delete Partner

Purpose:

To verify that all related information is deleted when the partner is deleted, and that the user is given a cancel opportunity before the deletion takes place.

The function shall be verified by checking the contents of the database after the removal of a partner is executed. Furthermore, it shall be verified that the Initiator and Responder receive a Restart command after the deletion.

Test procedure:

1. Start Initiator-log and Responder-log.
2. Choose partner TCPPARTNER.
3. Define a local group for the partner.
4. Select the function Partner *Delete*
5. Verify that a warning is presented to the user.
6. Cancel the deletion.
7. Use EQL to verify that no deletion is performed in the database.
8. Execute the Partner Delete function again, this time confirm the deletion.
9. Use EQL to verify that the partner characteristics and the group assignment are deleted in the database.
10. Use the Initiator and Responder log to verify that both servers have received Restart command.
11. Accept/reject test.

3.15 Create Periodically Requested Data Transfer

Purpose:

It shall be verified that the parameters for Periodically Requested Data Transfer can be defined.

EQL shall be used to check the data in the database.

Test procedure

1. Define periodically requested transfer for group 1.
2. Use EQL to verify the parameters in the database.
3. Accept/reject the test.

3.16 Modify Periodically Requested Data Transfer

Purpose:

It shall be verified that the parameters for Periodically Requested Data Transfer can be modified.

EQL shall be used to verify the modification of the data in the database.

Test procedure

1. Modify the Periodically Requested Data Transfer parameters for group 1.
2. Use EQL to verify the changes in the database.
3. Accept/reject the test.

3.17 Delete Periodically Requested Data Transfer

Purpose:

It shall be verified that parameters for Periodically Requested Data Transfer can be deleted.

EQL shall be used to verify the changes in the database.

Test procedure:

1. Delete periodically requested transfer for group 1.
2. Use EQL and check that the parameters for group 1 are deleted in the database.
3. Accept/reject the test.

3.18 Start Periodically Requested Data Transfer

Purpose:

Ref [2], chapter 7.2. It shall be tested that periodically requested data is transferred correctly from the ELCBAS Responder to the ELCBAS Initiator. Several cycles shall be transferred. It shall be verified that the function starts for groups which are configured for periodically requested data transfer and that an Application server connected to a transferred group is activated correctly.

EQL will be used to verify the data transfer in the database. Initiator-log will be used to confirm the data transfer.

Test procedure:

1. Turn Initiator-log On.
2. Define Periodically Requested Data Transfer for group 1.
3. Define Initiator Application Server name for Periodically Requested Data Transfer.
4. Select the function Periodically Requested Data Transfer *Start*
5. Select a time to start the transfer some minutes ahead.
6. Check the Initiator-log to verify that the data is transferred periodically.
7. Use EQL to verify that data is transferred.
8. Verify that the Initiator Application Server is activated with information about which data that has arrived.
9. Repeat items 2 through 7 for group 10 (large group).
10. Stop Periodically Requested Data Transfer for group 10.
11. Accept/reject test.

3.19 Stop Periodically Requested Data Transfer

Purpose:

It shall be verified that the function stops periodically requested transfer for an activated group.

Initiator-log shall be used to verify that the requested data transfer is stopped.

Test procedure:

1. Select the function Periodically Requested Data Transfer *Stop*
2. Stop Periodically Requested Data Transfer for group 1.
3. Turn Initiator-log Off.
4. Check the Initiator-log to verify that data transfer is stopped.
5. Accept/reject test.

3.20 Create Unsolicited Data Transfer

Purpose:

To verify that the parameters for Unsolicited Data Transfer can be created.

EQL shall be used to check that the information in the database is updated.

Test procedure:

1 Select the function Unsolicited Data Transfer *Define*.
2. Define Unsolicited Data Transfer for group 2.
3. Use EQL to verify the parameters in the database.
4. Accept/reject the test.

3.21 Modify Unsolicited Data Transfer

Purpose:

To verify that the parameters for Unsolicited Data Transfer can be modified.

EQL shall be used to check that the information in the database is updated.

Test procedure:

1. Select the function *Unsolicited Data Transfer Change*.
2. Change the parameters for group 2.
3. Use EQL to verify the changes in the database.
4. Accept/reject the test.

3.22 Delete Unsolicited Data Transfer

Purpose:

To verify that the parameters for Unsolicited Data Transfer can be deleted.

EQL shall be used to check that the information in the database is updated.

Test procedure:

1. Select the function Unsolicited Data Transfer *Delete*.
2. Delete the information for group 2.
3. Verify that the information for group 2 is deleted in the database.
4. Accept/reject the test.

3.23 Start Unsolicited Data Transfer

Purpose:

Ref [2], chapter 7.4. To verify that unsolicited data is transferred correctly from the ELCBAS Responder to the ELCBAS Initiator. It shall be checked that the function starts for groups which are configured for unsolicited data transfer.

EQL and Initiator-log shall be used to verify the function.

Test procedure:

1. Turn Initiator-log On.
2. Define Unsolicited Data Transfer for group 2.
3. Select the function Unsolicited Data Transfer *Start*.
4. Define Initiator Application Server name for Unsolicited Data Transfer.
5. Start Unsolicited Data Transfer for group 2.
6. Verify that the complete set of (initial) object values for group 2 is transferred.
7. Use the Responder type Application Server to modify the object values assigned to group 2.
8. Use the Initiator-log to verify that the modified objects are transferred.
9. Use EQL to check that the transferred object values in the Initiator database are the same as those manually entered in the Responder database.
10. Verify that the Initiator Application Server is activated with correct information about the data transferred.
11. Define Unsolicited Data Transfer for group 10.
12. Repeat items 4 through 10 for group 10 (large group).
13. Stop Unsolicited Data Transfer for group 10.
14. Repeat items 11 through 13 for group 4 (Logical Breaker status values).
15. Accept/reject the test.

3.24 Stop Unsolicited Data Transfer

Purpose:

To verify that the function stops unsolicited data transfer.

EQL and Initiator-log shall be used to verify the function.

Test procedure:

1. Start Unsolicited Data Transfer for group 4.
2. Select the function unsolicited data transfer *Stop*.
3. Stop Unsolicited data transfer for group 2.
4. Use EQL to modify some of the object for group 2 and 4 in the responder database.
5. Verify by means of the Initiator-log that only the modified objects for group 4 are transferred.
6. Turn Initiator-log off.
7. Stop Unsolicited data transfer for group 4.
8. Accept/reject the test.

3.25 Mixed Data Transfer

Purpose:

Ref [2], chapter 7.5. To verify that mixed data transfer is performed when the priority class for a group is set correctly.

EQL and initiator-log shall be used to verify the function.

Test procedure:

1. Change priority class for group 4 to 1, start Unsolicited transfer for the group.
2. Change data and verify that data is transferred as spontaneous data.
3. Stop Unsolicited transfer for the group; change priority class to 2; start unsolicited transfer for the group.
4. Change data and verify that data is transferred as high priority mixed data.
5. Stop unsolicited transfer for the group; change priority class to 0; start unsolicited transfer for the group.
6. Change data and verify that data is transferred as low priority mixed data.
7. Stop unsolicited transfer for the group.
8. Accept/reject the test.

3.26 Create Periodic Data Transfer

Purpose:

To verify that the parameters for Periodic Data Transfer can be created.

EQL shall be used to verify the contents of the database.

Test procedure:

1. Select the function Periodic Data Transfer *Create*.
2. Define Periodic Data Transfer for group 1.
3. Use EQL to verify the contents of the database.
4. Accept/reject the test.

3.27 Modify Periodic Data Transfer

Purpose:

To check that the parameters for Periodic Data Transfer can be modified.

EQL will be used to verify the contents of the database.

Test procedure:

1. Start Periodic Data Transfer for group 1.
2. Select the function Periodic Data Transfer *Modify*.
3. Try to modify the parameters for group 1.
4. Verify that the changes are rejected.
5. Stop periodic transfer for group 1.
6. Modify the parameters for group 1.
7. Use EQL to verify the contents of the database.
8. Accept/reject the test.

3.28 Delete Periodic Data Transfer

Purpose:

To verify that Periodic Data Transfer can be deleted for a group.

EQL will be used to verify the function.

Test procedure:

1. Start Periodic Data Transfer for group 1.
2. Select the function Periodic Data Transfer *Delete*.
3. Try to delete the information for group 1.
4. Verify that the deletion is rejected.
5. Stop Periodic Data Transfer for group 1.
6. Delete the information for group 1.
7. Use EQL to verify that the information is deleted in the database.
8. Accept/reject the function.

3.29 Start Periodic Data Transfer

Purpose:

Ref [2], chapter 7.3. To test that periodic data is transferred correctly from the ELCBAS Responder to the ELCBAS Initiator. Several cycles shall be transferred. It shall be verified that the function starts for groups which are configured for periodic data transfer.

EQL and Initiator-log will be used to verify the function.

Test procedure:

1. Turn Initiator-log on.
2. Select the function Periodic Data Transfer *Start*.
3. Assure that the information for group 1 is different in the Initiator and Responder databases using EQL.
4. Define Periodic Data Transfer for group 1.
5. Define Initiator Application Server name for Periodic Data Transfer.
6. Start Periodic transfer for group 1.
7. Check the Initiator-log to verify that periodic transfer is active for group 1.
8. Use EQL to verify that transferred data is written into the Initiator database.
9. Verify that the Initiator Application Server is activated with correct information about the data transferred.
10. Repeat items 4 through 9 for group 10 (large group).
11. Repeat items 4 through 9 for group 2 (status values).
12. Repeat items 4 through 9 for group 4 (logical breaker status values).
13. Accept/reject the test.

3.30 Stop Periodic Data Transfer

Purpose:

To verify that the function stops Periodic Data Transfer for a selected group.

Initiator-log and EQL shall be used to verify the function.

Test procedure:

1. Select the function Periodic Data Transfer *Stop*.
2. Stop periodic transfer for group 1.
3. Wait at least one period (see next test).
4. Check the Initiator-log to verify that periodic transfer is stopped for group 1 only.
5. Stop Periodic Data Transfer for group 2, 4 and 10.
6. Accept/reject the test.

3.31 Define Period Length for Periodic Data Transfer

Purpose:

To verify that the period length can be modified while periodic data transfer is active.

Initiator-log and EQL will be used to check the function.

Test procedure:

1. Select the function Define Period Length
2. Modify Responder period length
3. Check the Initiator-log to verify that the active Periodic Data Transfer for group 1, 2, 4 and 10 is stopped and restarted with new period length.
4. Accept/reject the test.

3.32 Delete Group

Purpose:

Ref [2], chapter 6. To verify that a group can be deleted both locally, remote and global.

The function shall be checked by inspecting the contents of the remote and local databases. Data transfer for global group definitions shall be verified with the Responder-log.

The group definition shall be performed on the Initiator side.

Test procedure:

1. Start Periodic Requested Data Transfer for Group 3.
2. Select the function Group *Delete.*
3. Try to delete Group 3 Global.
4. Verify that the deletion is rejected.
5. Stop Periodic Requested Data Transfer for Group 3.
6. Delete Group 3 Global.
7. Verify that transfer to the Responder takes place (Responder Log).
8. Use EQL to check that the group is deleted in both the local and remote databases.
9. Delete group 2 Locally.
10. Verify that no data transfer to the Responder takes place.
11. Verify that the group is deleted in the Initiator database.
12. Redefine Group 3 Global.
13. Delete group 3 Remote.
14. Verify that transfer to the Responder takes place (Responder Log).
15. Use EQL to check that the group is deleted in the remote database.
16. Try to enter some illegal values:
    Delete group with *Owner*=Responder, and *Where to be deleted* = R(emote).
    Delete group with *Permanent*=yes, and *Where to be deleted* = R(emote).
17. Redefine group 2 and 3 as they where before the test started.
18. Accept/reject test.

3.33 Initiator Data Transfer (Send Data)

Purpose:

To verify that transfer of object values for a selected group can be transmitted on request.

EQL shall be used to verify the contents of the database and Responder-log shall be used to check that data transfer is performed for the selected group.

The values for the selected objects should be different in the Initiator and Responder Databases before the test is started.

Test procedure:

1. Define Application Server Name for Responder.
2. Specify Initiator Data Transfer for group 1, with dT = 1, Time Unit = 4, and Periods = 24.
3. Use Initiator type Application Server in System B to verify that data transfer is performed for group 1.
4. Verify with EQL that the values in the Responder database is updated with correct values.
5. Verify that the Initiator type Application Server is activated with correct information about the transferred data.
6. Repeat items 2 through 5 for group 10 (large group).
7. Specify Initiator Data Transfer for group 5 (text message), with dT = 1, Time Unit = 4, Periods = 1.
8. Use Initiator type Application Server to verify that data transfer is performed for group 5.
9. Verify with EQL that the Responder database is updated with correct text messages.
10. Verify that the Initiator type Application Server is activated with correct information about the transferred data.
11. Accept/reject the test.

3.34 Command Transfer

Purpose:

Ref [2], chapter 7.6. To verify that commands can be sent to a remote responder application server.

Special test versions of initiator and responder applications servers are used for this test. These give printouts, which are used to verify the test, together with Initiator/Responder log.

Test procedure:

1. Define Application Server for Responder. Make sure path is blank, and start the *csrecv* program manually in the responder system. Make sure group 6 is configured.
2. Send a command for one object in group 6 with *cssend*. Verify with printout from *csrecv* and Initiator/Responder logs that the command is transferred, and the answer is sent back. The following Commands shall be transmitted:
   "Immediate Execute"
   "Checkback", and then "Execute"
   "Checkback", and then "Inhibit"
3. Accept/reject the test.

3.35 Setpoint Transfer

Purpose:

Ref [2], chapter 7.6. To verify that setpoints can be sent to a remote responder application server. Special test versions of initiator and responder applications servers are used for this test. These give printouts, which are use to verify the test, together with Initiator/Responder log.
EQL can also be used to verify data transfer, as setpoints are stored in the database.

Test procedure:

1. Make sure csrecv is running in the responder system, and that group 7 and 8 are configured.
2. Store values for a floating point setpoint and send with *cssend*. Verify with printouts from *csrecv*, Logs and EQL that the setpoint is transferred and that an appropriate answer is returned.
3. Repeat with a range of setpoints in the group
4. Repeat 2 for integer setpoints
5. Repeat 3 for integer setpoints
6. Accept/reject the test

3.36 Communication Status

Purpose:

It shall be verified that Initiator and Responder respond to inquiries about transfer status, and that the result is presented on the screen.

The function is controlled by checking Initiator- and Responder-log and by comparing the presented information from A- and B-side.

Test procedure:

1. Turn Initiator-log On.
2. Start Unsolicited Data Transfer for Group 2.
3. Select the function *Get Communication Status.*
4. Perform status check from A-side.
5. Verify that the status is transmitted from the B-side and that the communication status is presented on the screen.
6. Perform status check from the B-side and verify that the Unsolicited Data Transfer Function Unit is active for Responder.
7. Compare the presented information with that presented on the A-side and verify the correctness of the information.
8. Turn Initiator-log Off.
9. Accept/reject test.

3.37 Reconfigure All Initiator Groups

Purpose:

The function is used to bring the group configuration in the Responder into accordance with the corresponding Initiator group configuration, e.g. after a database backup is installed .

The function is verified by controlling that data transfer is stopped, and do not start again until the responder group is reconfigured from the Initiator.

Test procedure:

1. Start Initiator-log and Responder-log.
2. Stop all data transfer.
3. Select the function Initiator Reconfiguration
4. Verify by means of the Responder-log that the groups are reconfigured from the Initiator.
5. Use EQL to verify that group configuration is the same in the Initiator and Responder.
6. Accept/reject test

3.38 Reconfigure All Responder Groups

Purpose:

The function is used to delete the Responder group configuration to make the remote Initiator reconfigure the group by next association establishment.

The function is verified by controlling that all data transfer is stopped, and that the responder group configuration is deleted in the database. Furthermore, it shall be checked that the responder group configuration is reconfigured when the association to the partner is reestablished.

Test procedure:

1. Start Initiator-log in System A and Responder-log in System B.
2. Start Periodically Requested Data Transfer from System B for group 1 with Cyclic dT = 5, Cyclic Time Unit = 5 (minutes), and Periods = 1.
3. Select the function Responder Reconfiguration on System B.
4. Use EQL to verify that the Control Field (CF) of all R-groups belonging to the partner is cleared.
5. Use Initiator-log to observe the change in the CF field of the user data of the Connect Request at next attempted periodically requested data transfer.
6. Use the Initiator-log to verify that a new association is established for Group Configuration.
7. Use the Responder-log to verify that the Initiator sends a Group Management with function "Delete-all-groups" on all R-groups belonging to the partner (refer item 4 above).
8. Verify by means of the Responder-log that all groups (refer above) are reconfigured by the Initiator.
9. Observe that Periodically Requested Data Transfer restarts when the reconfiguration is completed (the missing period shall be collected first).
10. Accept/reject test

3.39 Automatic Reconfiguration

Purpose:

Ref [2], chapter 8.1. To verify that automatic reconfiguration is performed when the Control Field of a group is different in System A and System B.

The log function is used to verify that reconfiguration takes place.

Test procedure:

1. Start the Initiator-log on System A.
2. Start Unsolicited Data Transfer for group 1.
3. Destroy the Control Field (CF) for the partner in the database on system B.
4. Restart the Responder in System B.
5. After some time, inspect the Initiator-log and verify that automatic reconfiguration has been performed, and that Unsolicited Data Transfer is restarted for group 1.
6. Stop the Initiator-log in System A.
7. Accept / reject test.

3.40 Automatic Data Deletion function

Purpose:

This function is only implemented in the Unix implementation of Elcbas-90.

It shall be tested that the Automatic Data Deletion function operates with default timeout values and with user supplied timeout values for some objects. Furthermore, it shall be verified that the delete function depend on the data type and the parameter that defines the number of values that shall remain after the deletion.

The function is verified by checking the data in the database before and after the deletion shall take place according to defined timeout values.

Test procedure:

1. Check that the timeout values for object T10 and T11 are 48 hours (default).
2. Check that Number of Status values and Number of Text values is n (n = number of values to remain after deletion).
3. Set the default timeout value to 2 hours.
4. Modify timeout value for objects T10, T20 and T50 to 1 hour.
5. Perform n+1 data transfers of group 1, 2 and 5.
6. Stop the data transfer.
7. Check that the data is stored in the database.
8. After 1 hour check that:
   - the data for T10 is completely deleted, except latest value.
   - n incarnations of objects T20 and T50 remains in the database.
9. After 2 hours, check that the data for T11 is deleted.
10. Accept/reject test.

3.41 Watchdog function

Purpose:

This function is only implemented in the Unix implementation of Elcbas-90.

To verify that the Watchdog function performs periodically status checks on the availability of the Initiator and Responder, and performs Test Connection service calls to all of the Initiators Partners.

Test process:

1. Turn Initiator-log On.
2. Use the Initiator-log to verify that the Watchdog function performs Test Connection Service call to all of the Initiators Partners with predefined intervals.
3. Make one of the Initiator's defined partners unavailable.
4. Verify that the Watchdog function reports the status of the unavailable partner.
5. Stop the Initiator.
6. Verify that the Watchdog function reports the new Initiator status.
7. Stop the Responder.
8 Verify that the Watchdog function reports the new Responder status.
9. Accept/reject the test.

3.42 Switch Over

Purpose:

For each partner three alternative lower level addresses can be specified. If the ELCBAS Initiator fails to establish an association to a partner, the next address will be tried until all three addresses have been tried.

The switch-over function will be verified by defining three lower addresses for a partner, where the two first addresses are incorrect. The Initiator shall try to establish an association to the partner using each of the three addresses with one minute interval, finally succeeding with address number 3. If all fails, wait one minute before retry.

Test procedure:

1. Start Initiator-log.
2. Select the function Partner *Create*
3. Define a partner with two fictitious (unreachable) and one correct (reachable) lower level addresses.
4. Define a group on System A and transfer the group definition to System B.
5. Start periodic transfer for the group.
6. Stop the Responder.
7. Use the Initiator-log to verify that the Initiator tries to establish the association to the partner using each of the three defined addresses with one minute intervals.
8. Wait one minute for the final timeout after three failures.
9. Start the Responder.
10. Verify that the Initiator succeeds to establish the association to the partner with the last of the three lower level addresses defined for the partner.
11. Accept/reject the test.

3.43 Print Functions

Purpose:

This function is only implemented in the Unix implementation of Elcbas-90.

The print function shall be verified by checking that Elcbas-90 configuration is written to a print file. This function is not implemented on all platforms.

The function is verified by comparing the printed information with the information in the database.

Test procedure:

1. Select the function *Print Partner Information.*
2. Compare the printed information with the database contents by means of EQL.
3. Select the function *Print Object Information.*
4. Compare the printed information with the database contents by means of EQL.
5. Select the function *Print Group Information.*
6. Compare the printed information with the database contents by means of EQL.
7. Select the function *Print Access Information.*
8. Compare the printed information with the database contents by means of EQL.
9. Accept/reject test

3.44 Shutdown function

Purpose:

This function is only implemented in the Unix implementation of Elcbas-90.

The shutdown function is used to stop the ELCBAS System. The following ELCBAS components are stopped by the function:

- Purger
- Watchdog
- Initiator
- Responder

The shutdown function is implemented in a program started from the command prompt.

The purpose of the test is to verify that the shutdown function stops ELCBAS activity and stops the Purger, Watchdog, Initiator and Responder processes.

Test process:

1. Start Initiator- and Responder-log in System A.
2. Start Periodically Requested Data Transfer for Group 1 in System A.
3. Configure Group 2 for Periodic Data Transfer.
4. Start Periodic Data Transfer for Group 2 in System B.
5. Start the shutdown program in System A.
6. Verify that the Initiator and Responder in System A detaches all attached Service Access Points.
7. Verify that the Purger, Watchdog, Initiator and Responder processes are stopped by the shutdown function.
8. Accept/reject the function.

4 CPU LOAD TEST

The CPU load test is performed by transferring the data reference quantity (refer Appendix B) with one minute delay between each transfer for 15 minutes. Data for about 15 days will be required for the test.

The test will only be performed for a Regional Control Centre configuration, due to the extent of data needed in the test.

The relative CPU load by the ELCBAS-90 System shall be monitored during a 15 minute period. The UNIX command *time* shall be used to measure the time spent in user and kernel mode during the test period. This command may also be used for the Windows implementation, i.e. using the Cygnus Shell from a command window.

The relative load is calculated as the relationship between the sum of the time spent in user and kernel mode and the elapsed time.

The time spent in User mode represents the execution time of process "user" code (opposed to system call code).
The time spent Sys mode represents the time the kernel used for processing system calls made by the process.

The total CPU time in User and Sys mode shall be measured during the 15 minutes period and shall not exceed 10% of elapsed time.

4.1 Regional Control Centre Test

Purpose:

Reference data quantity PC-BC is transferred for 15 minutes to the Initiator. At the same time the reference data quantity BC-BC shall be transferred from the local Responder. CPU time for both Initiator and Responder shall be measured during the transfer.

The sum of CPU time (User and Sys mode) for the Initiator and Responder shall not exceed 10% of the elapsed time. Measurements will be performed by running both Initiator and Responder under the system call time.

Test procedure:

1. Start Initiator and Responder with the UNIX command time.
2. Start Periodically Requested Data Transfer of the actual groups for a period ahead in time.
3. After 15 minutes, stop the Initiator and Responder.
4. Calculate total CPU time used by the Initiator and Responder from the information printed by the time call.
5. Calculate the percentage CPU time for Initiator and Responder in the period and compare with requirement.
6. Accept/reject test

5 PERFORMANCE TEST

The purpose of the performance tests is to measure the throughput for the Initiator and Responder. Two tests are defined in the following sub chapters; one for the Initiator and one for the Responder.

The tests shall be run on an unloaded computer.

5.1 Initiator

Purpose

The Initiator ELCOM Time, **TIE** as defined below, shall be measured.

This is done by checking the log files from the Initiator with reference to when the data passes through the program interfaces.

The reference data quantity PC-BC shall be transferred.

The **Initiator ELCOM time** is defined as **TIE = T2 - T1**, where **T1** is the time when ADTI is called in the Initiator. This measurement corresponds to when that data is available from the ELCOM Application layer, before the ELCBAS Initiator reads the data from the ADTI service primitive. **T2** is the time when the data in the ADTI service primitive is written in the database. This measurement is performed after the data has passed the ELCBAS - database interface.

The Initiator ELCOM time is calculated by the Initiator, and the results are printed to a log-file. This function is controlled by the environment variable "ELCBAS_FAT". Mean time and standard deviation should be calculated separately.

Test procedure:

1. Copy the test data to the databases on system A and B.
2. Stop all logs and trace.
3. Define a point in time for the transfer of the reference data, and start the data transfer.
4. Stop the data transfer after the reference data has been transferred.
5. Make a print-out of the results.
6. Calculate the times, and fill in the results in the table below.
7. Compare the results with the requirements.
8. Accept/reject the test.

TIE *Measured*	**TIE** *Reference*	**Standard deviation** *Measured*	**Standard deviation** *Reference*
	150 msec		20%

5.2 Responder

Purpose:

The Responder ELCOM time, **TRE** as defined below, shall be measured.

The test is performed by comparing log files with respect to when data passes through the interfaces.

The reference data quantity BC-BC shall be transferred.

Responder ELCOM time is defined as **TRE = T1 - T2**, where **T1** is the time after ADTRQ is called by Responder. This time refers to when the data is transferred to the ELCOM Application Layer. **T2** is the time before the data is read from the database. This is also measured before data for the group has passed the ELCBAS/database interface.

The Responder ELCOM time is calculated by the Responder, and the results are written to a log file. This function is controlled by the environment variable "ELCBAS_FAT". The mean time and standard deviation should be calculated separately.

Test procedure:

1. Copy the test data to the databases on the A- and B-systems.
2. Stop all logs and trace.
3. Define a point of time for data transfer of the reference data.
4. Start the data transfer.
5. Stop the data transfer after the reference data quantity is transferred.
6. Print out the results.
7. Calculate the mean time and standard deviation and fill in the table below.
8. Compare the measured times with the requirement.
9. Accept/reject the test

TRE *Measured*	**TRE** *Reference*	**Standard deviation** *Measured*	**Standard deviation** *Reference*
	100 msec		20%

6 MULTI-PARTNER COMMUNICATION TEST

The similarity test is performed by simulating communication with several partners over the local area network. Communication with several partners will be possible by defining the same lower level address for several partners. When defining the address for a partner in the ELCBAS Supervisor the entered address must be unique. Thus, EQL must be used for defining the addresses.

Several sets of the same groups must be defined on both sides, with different partners as "owner".

The test shall be performed for National Control Centre configuration, but may also be performed for the Regional Control Centre.

6.1 National Control Centre Configuration

Purpose:

8 times the reference data quantity BC-BC shall be transferred, i.e. to 8 (local) partners. The data transfer to all partners shall be started within a timeframe of 10 seconds. The data shall be transferred correctly and completely and without time-out times being exceeded.

The ELCBAS logs shall be used to measure time when transfer is started, and EQL shall be used to verify the contents of the database. The relevant parts of the database shall be deleted before the test starts, in order to verify that the data has been transferred.

Test procedure:

1. Delete the relevant parts of the database.
2. Copy the test data to the database on the A- and B-system.
3. Start periodically requested data transfer for all 8 partners from a given time ahead in time
4. Start the ELCBAS logs.
5. Wait till the data transfer starts.
6. Use EQL to check when the data transfer is completed, and then stop the Initiator.
7. Inspect the Error log to verify that no fatal errors has happened.
8. Use EQL to verify that data is transferred correctly.
9. Inspect the ELCBAS logs to verify that the transmission has started within the time requirement (10 seconds).
10. Accept/reject test.

6.2 Regional Control Centre Configuration

Purpose:

One time the reference data quantity PC-BC shall be transferred. At the same time 5 times the reference data quantity BC-BC shall start, i.e. to 5 logical partners. At the same time it shall be possible to configure 8 groups of 30 objects each. These actions shall start within a timeframe of 10 seconds. The data shall be transferred correctly and completely without time-out values being exceeded.

The ELCBAS logs are used to measure the time when transmission starts and EQL is used to verify the contents of the database. The related database tables shall be deleted before the test, in order to verify that the data is transferred.

Test procedure:

1. Configure 8 groups of 30 objects locally.
2. Delete the relevant database tables.
3. Copy the test data to the database on the A and B-system.
4. Start periodically requested data transfer for all 5 partners from a given point ahead in time on B side, and for 1 partner on the A side.
5. Start the ELCBAS logs.
6. Wait until the transfer starts.
7. Transfer the pre-defined groups (refer item 1 above).
8. Use EQL to check when the data transfer is completed, and then stop the Initiator on A- and B side.
9. Inspect the Error log to verify that no fatal errors has occurred.
10. Use EQL to verify that data is transferred correctly.
11. Use the ELCBAS logs to verify that the transmission started within the required timeframe (10 seconds).
12. Accept/reject test

7 APPENDIX A TEST PROTOCOL FORM

The test protocol form for the ELCBAS-90 FAT is displayed on the next pages. A test protocol shall be made for each test performed. The test procedures shall be referenced by the corresponding chapter number in this document (refer item *Test Case* in the form).

Hardware Configuration

(ELCBAS-90 / ELCBAS-90)

System A	**Description**
Machine type	
Memory	
Disk	
Communication hardware	
Additional information	

System B	**Description**
Machine type	
Memory	
Disk	
Communication hardware	
Additional information	

Software Configuration

(ELCBAS-90 / ELCBAS-90)

System A	**Description**
Operating System	
Database	
EQL Inspect tool	
Application Servers	
Support software used	
Additional information	

System B	**Description**
Operation System	
Database	
EQL Inspect tool	
Application Servers	
Support software used	
Additional information	

ELCBAS-90 Factory Acceptance Test

Test Protocol

Test case: ....

Comment:

☐	Approved	Sign: ....
☐	Approved with comment	Sign: ....
☐	Rejected	Sign: ....

8 APPENDIX B TEST DATA

8.1 Data for functional tests

Partner:

PID = 1
TCP/IP lower level address and port number:
X25 address:
Standard suffix.

Object Identifiers:

Oname	Type	Comment
T10 . . T19	Float	1 hour resolution
T20 . . T29	Status	
T30 . . T39	Integer	1 hour resolution
T40 . . T49	Logical breaker	
T50 . . T59	Text	
T201 . . T320	Float	1 hour resolution

Group Configuration:

Ordinary groups:

Group 1 :	Objid10 -	Objid19	(Floating point values)
Group 2 :	Objid20 -	Objid29	(Status values)
Group 3 :	Objid30 -	Objid39	(Integer values)
Group 4 :	Objid40 -	Objid49	(Logical Breaker status values)
Group 5 :	Objid50 -	Objid59	(Text messages)
Group 6 :	Objid20 -	Objid29	(Command group)
Group 7 :	Objid10 -	Objid19	(Floating point setpoint group)
Group 8 :	Objid30 -	Objid39	(Integer setpoint group)

<u>Large group:</u>
Group 10 : Objid201 - Objid320 (Floating point values)

Data for group 1 shall be stored in the B-system for 48 hours.

Data for group 2, 4, 5 and 10 shall be stored in the B-system for 24 hours.

Data for group 3 shall be stored in the B-system for 24 hours as forecast data.

8.2 Data for CPU load, Performance and Multi-Partner Tests

Partner with PID = 1 is predefined.
All objects are of type float.
Data shall be stored in the Responder system for 24 hours.

Reference data quantity **PC-BC** is predefined in the following groups:

Group 21	Objid1	= PC1.1
	Objid2	= PC1.2
	.	
	.	
	.	
	Objid30	= PC1.30
Group 22	Objid1	= PC2.1
	Objid2	= PC2.2
	.	
	.	
	.	
	Objid30	= PC2.30
.		
.		
.		
Group 28	Objid1	= PC8.1
	Objid2	= PC8.2
	.	
	.	
	.	
	Objid30	= PC8.30

Reference data **BC-BC** is predefined as follows:

Group 31	Objid1	= BC1.1
	Objid2	= BC1.2
	.	
	.	
	.	
	Objid10	= BC1.10
Group 32	Objid1	= BC1.1
	Objid2	= BC2.2
	.	
	.	
	.	
	Objid10	= BC2.10
.		
.		
.		
Group 34	Objid1	= BC4.1
	Objid2	= BC4.2
	.	
	.	
	.	
	Objid10	= BC4.10

8.3 Initial State of ELCBAS System when FAT starts

B.3.1 Unix version database.
The following database tables shall be empty on the A-side when the FAT starts:

ELCOBJ
ELCACC
ELCGROUP
ELCGROBJ
ELCFREQ
ELCTMOUT
ELCCYCLE
ELCSERV
ELCMAIL
ELCIR-VL
ELCST-VL
ELCTX-VL

The following database tables shall contain data for local configuration and pre-defined partner when the FAT starts:

ELCADM
ELCPART
ELCSUFF
ELCADR

B3.2 Windows version configuration.

The Elcbas configuration shall only contain local configuration when the FAT starts. One predefined remote partner shall however be configured.