



# **ClearPath Enterprise Servers**

## **SURE Technical Overview**

SURE Release 8.0

May 2012

8207 4121-000

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THIS DOCUMENT. Any product or related information described herein is only furnished pursuant and subject to the terms and conditions of a duly executed agreement to purchase or lease equipment or to license software. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Notice to U.S. Government End Users: This is commercial computer software or hardware documentation developed at private expense. Use, reproduction, or disclosure by the Government is subject to the terms of Unisys standard commercial license for the products, and where applicable, the restricted/limited rights provisions of the contract data rights clauses.

# Contents

## **Section 1. Introduction**

1.1.	Overview .....	1-1
1.1.1.	Today's Requirements .....	1-1
1.1.2.	Improved Quality and Productivity .....	1-2
1.1.3.	Unique to Unisys ClearPath MCP .....	1-3
1.1.4.	Support for Distributed Environments .....	1-3
1.1.5.	Integrates with IDE's such as Eclipse and Microsoft Visual Studio .....	1-3
1.2.	Features .....	1-4
1.2.1.	Flexible Source Control System .....	1-4
1.2.2.	Integrated Task System .....	1-8
1.3.	What is an Overlapping Task? .....	1-13
1.3.1.	What is Quick Fix? .....	1-14
1.4.	Configurable Environments .....	1-15
1.4.1.	Configuring Workflow .....	1-15
1.4.2.	Workflow Diagram .....	1-16
1.5.	Fully Automated Build Process .....	1-17
1.6.	Modernized MCP Development .....	1-19
1.7.	Seamless Integration with IDEs .....	1-22
1.8.	Transitioning to SURE .....	1-24

## **Section 2. Software Management and Change Control**

2.1.	The First Version of SURE .....	2-1
2.2.	Changes in the Last Decade .....	2-1
2.3.	Control of Development Life Cycle .....	2-2

## **Section 3. Benefits of Using SURE**

3.1.	Enforce Development Standards .....	3-1
3.2.	Independence from Individuals .....	3-3
3.3.	Automated Control of Environments .....	3-3
3.4.	Workbench for MCP Development .....	3-4
3.5.	Use of SURE for PC Development Projects .....	3-5

## **Section 4. Modes of Operation with SURE**

4.1.	Development on ClearPath Libra Servers .....	4-1
4.2.	SURE for Windows GUI Interface .....	4-1

### **Section 5.   Versioning within Life Cycle Support**

5.1.	Using Tasks to Identify Changes.....	5-1
5.2.	What is an Overlapping Task? .....	5-2
5.3.	What is a Quick Fix? .....	5-4
5.4.	How is Integrity Maintained? .....	5-5

### **Section 6.   SURE Development Support**

6.1.	Using the Differences Between Successive File Versions .....	6-1
6.2.	(SURE Development Support) Query Facility.....	6-3
6.3.	(SURE Development Support) Information .....	6-3
6.4.	Run-Time Statistics.....	6-4
6.5.	Event Log.....	6-4
6.6.	Controlled and Automated Management of Environments .....	6-4

### **Section 7.   Applicability of SURE**

7.1.	Development of Internal Software.....	7-1
7.2.	Development of Packages.....	7-2

### **Section 8.   Installation**

8.1.	Installation on the PC.....	8-1
8.1.1.	(Installation) License Key .....	8-1
8.1.2.	(Installation) PC Application Structure .....	8-1
8.1.3.	(Installation) PC Install Procedure .....	8-3
8.1.4.	(Installation) PC Install Program.....	8-4
8.1.4.1.	The First Installation Dialog.....	8-4
8.1.4.2.	The Second Installation Dialog.....	8-5
8.1.4.3.	The Third Installation Dialog .....	8-6
8.1.4.4.	The Fourth Installation Dialog.....	8-7
8.1.4.5.	The Fifth Installation Dialog .....	8-8
8.1.4.6.	The Sixth Installation Dialog .....	8-8
8.1.5.	(Installation) PC Configuration Management.....	8-10
8.1.5.1.	(Installation) PC Standalone Configuration.....	8-10
8.1.5.2.	(Installation) PC Server-Only Configuration .....	8-11
8.1.6.	(Installation) Multiple Installations on One PC .....	8-16
8.1.7.	(Installation) The AW_OBJ.INI File.....	8-17
8.1.8.	Customized Post Installation Actions .....	8-21
8.1.9.	SURE Triggers New Installation According to Versions .....	8-22
8.2.	Installation on the Mainframe .....	8-27
8.2.1.	(Installation) Mainframe Application Structure .....	8-28
8.2.2.	(Installation) Repository INFDB .....	8-28
8.2.3.	(Installation) SURE Objects and Database Compatibility .....	8-29



8.2.3.1.	(Installation) Compatibility at a NEW Installation .....	8-31
8.2.3.2.	(Installation) Compatibility at an UPGRADE Installation .....	8-31
8.2.3.3.	(Installation) Compatibility Direction.....	8-32
8.2.4.	(Installation) RESPECT/TITLES .....	8-32
8.2.5.	(Installation) Object Files .....	8-35
8.2.5.1.	(Installation) SURE Explorer Interface .....	8-36
8.2.5.2.	(Installation) Terminal Emulation Interface .....	8-37
8.2.5.3.	(Installation) RESPECT/LIBRARY .....	8-37
8.2.5.4.	(Installation) Batch Programs .....	8-38
8.2.6.	(Installation) Mainframe Install Procedure .....	8-39
8.2.7.	(Installation) Mainframe Install Program.....	8-41
8.2.7.1.	The First Installation Screen .....	8-41
8.2.7.2.	The Second Installation Screen .....	8-42
8.2.7.3.	The Third Installation Screen .....	8-43
8.2.7.4.	The Fourth and Last Installation Screen .....	8-43
8.2.7.5.	Installation of a First Time User.....	8-46
8.2.7.6.	Upgrade Repository and Install All Objects .....	8-46
8.2.7.7.	Upgrade Repository and Install SURE Explorer Interface.....	8-47
8.2.7.8.	Installation of the SURE Objects .....	8-47
8.2.7.9.	Installation of the SURE Explorer Interface .....	8-47
8.2.7.10.	Creation of an Extra Repository .....	8-48
8.2.7.11.	Creation of a New Repository from Tape .....	8-48
8.2.8.	Definition of the SURE Explorer Interface in COMS .....	8-48

## **Section 9. Configuration and Sizing**

9.1.	(Configuration) Terminology .....	9-1
9.2.	(Configuration) Quick Overview to SURE .....	9-5
9.3.	(Configuration) SURE Configuration .....	9-6
9.4.	(Configuration) Goal .....	9-8
9.5.	(Configuration) Functionality .....	9-8
9.6.	(Configuration) File Entities .....	9-9
9.6.1.	(Configuration) Environment.....	9-10
9.6.2.	(Configuration) System .....	9-12
9.6.2.1.	(Configuration) Work-Location and Object-Location .....	9-12
9.6.2.2.	(Configuration) Private Workspace/Shared Resources .....	9-14
9.6.2.3.	(Configuration) Shared Workspace/Shared Resources .....	9-16
9.6.2.4.	(Configuration) Private Workspace/Private Resources .....	9-17
9.6.2.5.	(Configuration) Multiple Baselines .....	9-18
9.6.2.6.	(Configuration) Hybrid Configuration.....	9-19
9.6.2.7.	(Configuration) Summary .....	9-19
9.6.3.	(Configuration) Project .....	9-21
9.6.4.	(Configuration) File Type.....	9-24

9.6.5.	(Configuration) File .....	9-28
9.7.	(Configuration) Task Entities.....	9-30
9.7.1.	(Configuration) Task .....	9-31
9.7.2.	(Configuration) Task Type .....	9-32
9.7.3.	(Configuration) Task Group .....	9-34
9.7.4.	(Configuration) Task Priority .....	9-35
9.8.	(Configuration) Organization Entities.....	9-35
9.8.1.	(Configuration) Employee Function .....	9-36
9.8.2.	(Configuration) Team .....	9-36
9.8.3.	(Configuration) User .....	9-37

## Section 10. Life Cycle Support

10.1.	Check-In Procedure .....	10-2
10.1.1.	FileVersion Attribute in SURE.....	10-2
10.1.2.	File Attribute RELEASEID.....	10-4
10.1.3.	MarkID.....	10-5
10.1.4.	Source Related Attributes .....	10-5
10.1.5.	Function CHECK-IN .....	10-6
10.2.	Check-Out Procedure.....	10-7
10.2.1.	Check-Out Attributes .....	10-8
10.2.2.	REQUEST .....	10-8
10.2.3.	LINK.....	10-10
10.2.4.	ASSIGN.....	10-10
10.2.4.1.	Assign a File to Another user+task Using Drag/Drop .....	10-11
10.2.5.	Function CHECK-OUT .....	10-11
10.2.6.	UNDO CHECK-OUT .....	10-12
10.2.6.1.	Function UNDO CHECK-OUT Leaves the Source on Disk (renamed).....	10-12
10.2.7.	Manipulate MCP Data Files Using Local Edit .....	10-13
10.3.	Transfer.....	10-14
10.3.1.	FileVersion Attribute.....	10-15
10.3.2.	STATUS Attribute .....	10-15
10.3.3.	Anchored files .....	10-16
10.4.	Enter a New File .....	10-17
10.4.1.	(Life Cycle Support) FileName.....	10-17
10.4.2.	Create a Maintenance Copy of the File .....	10-18
10.4.3.	New/File .....	10-18
10.5.	Load Files in SURE.....	10-18
10.5.1.	Load MCP Files in SURE .....	10-19
10.5.2.	Load PC Files in SURE .....	10-20
10.5.2.1.	Load PC Files in SURE .....	10-21
10.5.2.2.	Load Directory of PC Files in SURE .....	10-22
10.6.	Delete a File .....	10-24

## Section 11. Local Source Editing on PC

11.1.	Compile MCP Files Using the PC.....	11-5
11.1.1.	Compiling from a Windows Application: AS_COMP .....	11-7

11.1.2.	Merging Error Files .....	11-8
11.2.	Editor Support .....	11-9
11.2.1.	ED for Windows .....	11-10
11.2.2.	Microsoft Programmer's Workbench.....	11-11
11.2.3.	Unisys NX/Edit (Release 4.1).....	11-12
11.2.3.1.	NX/EDIT Template Support.....	11-12
11.2.4.	Microsoft Visual Studio C++ (Release 6.0) .....	11-14
11.2.5.	Microsoft Visual C++ (until Release 4.2) .....	11-16
11.2.6.	MultiEdit.....	11-17
11.2.7.	SCC Interface.....	11-19

## Section 12. Windows or UNIX Files in SURE

12.1.	(Windows or UNIX Files) Configuration .....	12-1
12.2.	(Windows or UNIX Files) File Name.....	12-3
12.3.	(Windows or UNIX Files) FILE-TYPE .....	12-4
12.4.	Concurrent Source Maintenance .....	12-6
12.5.	(Windows or UNIX Files) Source Files .....	12-8
12.6.	(Windows or UNIX Files) Detailed Description of Delta File .....	12-8
12.7.	(Windows or UNIX Files) Binary Files.....	12-10
12.8.	(Windows or UNIX Files) Build Support.....	12-10
12.8.1.	(Windows or UNIX Files) Build Support UNIX Extensions.....	12-15
12.8.2.	Make Naming of Build Script Files Configurable.....	12-18
12.8.3.	Starting Build Scripts.....	12-18
12.8.3.1.	Individual Build-Script for Each Source.....	12-18
12.8.3.2.	Common Make-File for the Entire Application .....	12-19
12.8.4.	Integrate Build on Windows with SURE-Batch on MCP.....	12-20
12.9.	(Windows or UNIX Files) Distribution Support .....	12-24
12.10.	SCC Provider.....	12-24
12.10.1.	Offline Processing for SCC Client .....	12-26
12.10.2.	File Extension Associations for the SCC Interface.....	12-28
12.10.3.	Behavior of Get Latest Version Using the SCC Interface.....	12-29
12.11.	JAVA Eclipse CVS pServer Interface .....	12-31
12.12.	Microsoft Visual Basic 6.0 Integration.....	12-41
12.13.	Microsoft Visual C++ 6.0 Integration .....	12-43
12.14.	Microsoft Visual Studio 2003 Integration .....	12-45
12.15.	Microsoft Visual Studio 2005 Integration .....	12-48
12.16.	MicroFocus Cobol Integration.....	12-51
12.17.	PowerBuilder Integration .....	12-52

## Section 13. Query Facility

13.1.	(Query) Relation Structure .....	13-1
13.2.	(Query) Names of Classes Used by the SURE Software .....	13-3

13.3.	(Query) Query Tool Windows GUI.....	13-4
13.4.	(Query) Query Syntax .....	13-5
13.5.	(Query) Macro Facility for the Select Function.....	13-12
13.5.1.	(Query) #SINCE in the MACRO Definition.....	13-13

### Section 14. Text as Information/Documentation

14.1.	Overview .....	14-1
14.2.	Information, Reminder, and Attachment .....	14-2

### Section 15. Copy Files

15.1.	Dependent Files.....	15-1
15.2.	(Copy files) Normal Mode .....	15-2
15.2.1.	(Copy files) Work-Environment .....	15-3
15.2.2.	(Copy files) <Put Copy Files in the Work Directory> .....	15-4
15.2.3.	<Use Never Object Relation> .....	15-5
15.2.4.	(Copy files) <Allow Copy Files From "*" > .....	15-5
15.2.5.	(Copy files) Copy or Include Statements .....	15-5
15.3.	(Copy files) <Use Resource Versions>.....	15-8
15.3.1.	(Copy files) Resource Statements, Syntax, and Qualification .....	15-9
15.3.2.	(Copy Files) Resource Environment .....	15-11
15.3.3.	(Copy Files) Resource Prefix .....	15-11
15.3.4.	(Copy Files) Resource Definition .....	15-11
15.3.5.	(Copy Files) Default Resource Version .....	15-12
15.3.6.	(Copy files) Resource Resolution Algorithm .....	15-12
15.3.7.	(Copy files) Changing Resource Versions .....	15-15
15.3.8.	(Copy Files) Resource Commands.....	15-16
15.3.9.	Overview: Resource Locations, Definitions, and Usage.....	15-17
15.3.10.	Nested Resources in Copy files.....	15-18

### Section 16. Life Cycle Support (Internals)

16.1.	(Internals) Check-In .....	16-1
16.1.1.	Compile as a Result of File Change .....	16-1
16.1.2.	Examine as a Result of a File Change .....	16-2
16.1.3.	Match as a Result of a File Change .....	16-2
16.2.	(Internals) Check-Out.....	16-2
16.3.	(Internals) Transfer .....	16-3
16.3.1.	Object Location as a Result of Transfer.....	16-4
16.3.2.	Delta Files as a Result of a Transfer .....	16-4
16.3.3.	Examine as a Result of Transfer .....	16-5
16.4.	(Internals) Enter a New File.....	16-7
16.5.	(Internals) Logical Delete a File .....	16-8
16.6.	(Internals) Undo Logical Delete a File.....	16-8
16.7.	(Internals) Purge a File in this Environment .....	16-8

**Section 17. Delta Files**

17.1.	Delta Files for Source Files .....	17-1
17.2.	Delta Files for Binary Files .....	17-4
17.3.	(Delta Files) Windows .....	17-5
17.3.1.	(Delta files) Create.....	17-6
17.3.2.	(Delta files) Copy .....	17-6
17.3.3.	(Delta files) Save as .....	17-6
17.3.4.	(Delta files) Write .....	17-7
17.3.5.	(Delta files) Patch.....	17-7
17.3.6.	(Delta files) List .....	17-9
17.3.7.	(Delta files) Roll back .....	17-9
17.3.8.	(Delta files) Compare .....	17-9
17.4.	(Delta files) Resequence .....	17-12
17.5.	(Delta files) Relation Modification .....	17-12
17.6.	(Delta files) Storage of Files and Delta Files.....	17-12
17.7.	(Delta files) Cleaning Old Delta Files .....	17-13
17.8.	(Delta files) RESPECT/SURE/MATCH.....	17-14

**Section 18. Repository Maintenance**

18.1.	Modify .....	18-1
18.1.1.	Allow to Define Multiple Functions for a File .....	18-3
18.2.	Rename (Popup Menu) .....	18-4
18.3.	Renamed PC Files are Removed on the Build Server .....	18-4
18.4.	Relate (Popup Menu).....	18-5

**Section 19. Find and Replace Utilities**

19.1.	Find.....	19-2
19.1.1.	Start the FIND Process .....	19-4
19.1.2.	FIND Results .....	19-5
19.2.	Replace .....	19-6
19.2.1.	Start the REPLACE Process .....	19-8
19.3.	RESPECT/SURE/FIND .....	19-10
19.4.	RESPECT/SURE/REPLACE .....	19-11

**Section 20. ADDS Support**

20.1.	Manual Integrity Trigger Overview .....	20-1
20.2.	Manual Integrity Trigger Example.....	20-1

**Section 21. Patch Files**

21.1.	(Patchfiles) Baselines .....	21-2
21.2.	Integration of Resources, Patch Files, and Compile Queues.....	21-2
21.3.	Merge Patchfile Permanently into the Main Source .....	21-8
21.3.1.	Merge Patchfiles Manually .....	21-8
21.3.2.	Merge Patchfiles Automatically .....	21-13

21.3.3.	Merge Method for Patch-Files .....	21-20
21.4.	Manual Maintenance of Patch Files .....	21-21
21.5.	Batch Function to Repair Patch-File Relations .....	21-23

### Section 22. Guard Files

22.1.	RESPECT/SURE/GUARD Overview .....	22-2
22.2.	SURE/GUARD/<dbname> Sample .....	22-2

### Section 23. Compilation and Object Files

23.1.	Daily Batch .....	23-2
23.1.1.	Open Up SURE Batch Overviews .....	23-3
23.1.2.	SURE/WFLINCLUDE .....	23-5
23.2.	Compilation: SURE Commands .....	23-5
23.2.1.	Compile .....	23-5
23.2.2.	Compile <task> .....	23-6
23.2.3.	UNDO Compile .....	23-6
23.2.4.	Purge a Compile Queue .....	23-7
23.2.5.	Compile Interface .....	23-8
23.2.6.	Activate a Compile Queue .....	23-9
23.2.7.	Deactivate a Compile Queue .....	23-9
23.2.8.	Check the Integrity Status .....	23-10
23.2.9.	View the Compilation Listing .....	23-10
23.2.10.	Set the Compile Listing Option .....	23-10
23.3.	Object Attributes .....	23-10
23.4.	Compile Options .....	23-12
23.4.1.	The ReleaseID of the Compiled Object .....	23-14
23.4.2.	The Security of the Compiled Object .....	23-15
23.4.3.	SURE FileVersion nr in Fileattribute "Version" of Objectcode .....	23-16
23.5.	File as Fixed Input for the Compile Process .....	23-18
23.6.	PRE and a POST Compile Job .....	23-19
23.7.	Compiler Control Cards .....	23-19
23.8.	Compiler Translate Tables .....	23-21
23.9.	Compiler Names .....	23-23
23.10.	Object Names .....	23-24
23.11.	Compile Listings .....	23-25
23.12.	Compilation Error Files .....	23-27
23.13.	Multiple Object Files .....	23-27
23.13.1.	Allow to Add Multi-Object Definition in Batch Mode .....	23-28
23.14.	Additional Object with the TADS Option .....	23-29
23.15.	Define your own Compile in SURE .....	23-30
23.16.	Integrity Mechanism .....	23-32
23.16.1.	Integrity Introduction .....	23-33
23.16.2.	Integrity Reason .....	23-34
23.16.2.1.	Integrity Chains Indicated by Tasks .....	23-35
23.16.2.2.	Integrity Chains and Adapted Copy Files .....	23-36
23.16.2.3.	Overlapping Integrity Chains .....	23-37

23.16.2.4.	Overlapping Integrity Chains and Copy Files .....	23-38
23.16.2.5.	Multiple Integrity Overlaps in Copy Files .....	23-39
23.16.3.	Manually Defined Integrity Reason .....	23-39
23.16.4.	Integrity Procedure .....	23-40
23.16.4.1.	Integrity Adaptations in RIS Modules .....	23-41
23.16.5.	Integrity: Loading the Compile Queue .....	23-45
23.16.5.1.	Integrity: Compilation Procedure .....	23-46
23.16.5.2.	Integrity: Various Information .....	23-50
23.16.6.	Batch Function to Check the Resources Used for Compilation .....	23-51
23.17.	RESPECT/SURE/COMPILE .....	23-52
23.17.1.	Compilation Overview .....	23-59
23.17.2.	Notify Users by Email About Compilation Syntax Errors .....	23-62
23.18.	Compiling through a Specific Compile Queue .....	23-64
23.18.1.	Compile Fast Queue .....	23-64
23.18.2.	User Defined Compile Queue .....	23-65
23.19.	RESPECT/SURE/TRANSFER .....	23-67
23.19.1.	Deploy for Each TASK .....	23-70
23.19.2.	Pack-to-pack Copy or BNA Copy .....	23-72
23.19.3.	User-Hook in the Deployment Procedure .....	23-73
23.19.4.	Creation of Transfer Jobs .....	23-74
23.19.5.	Copy or Dump Objects .....	23-79
23.19.6.	Extra Object-Location to Test Quick Fixes .....	23-80
23.20.	Manually Started Compilations by Developers .....	23-81
23.20.1.	Local Compilation Started from CANDE .....	23-81
23.20.2.	Local Compilation Started from SUREforWindows .....	23-82
23.20.3.	Extra Copy of Compiled Object .....	23-83

## Section 24. Binding of Programs

24.1.	(Binding) START-JOB and DRIVER Relations Overview .....	24-5
24.2.	(Binding) START-JOB and DRIVER Relations Examples .....	24-6
24.2.1.	(Binding) Which Start-Job Will Be Used and When Is It Started? .....	24-8
24.2.2.	(Binding) Maintaining DRIVER Relations .....	24-9
24.2.3.	(Binding) Maintaining Start-Job Relations .....	24-10

## Section 25. (Examine) Extracting Source Information

25.1.	(Examine) The Windows Menu Selection .....	25-2
25.1.1.	(Examine) Recognized Statements by the Examining Process .....	25-3
25.1.2.	(Examine) Relation Modification .....	25-4
25.2.	(Examine) RESPECT/SURE/EXAMINE .....	25-4
25.3.	Allow to Define a Site-Specific Examine Function .....	25-5

### Section 26. Backup of Maintenance Sources

26.1.	Backup of Non-Repository Files.....	26-1
26.2.	RESPECT/SURE/SECURE.....	26-2

### Section 27. Maintenance of Development Disk Space

27.1.	Cleanup Disks.....	27-1
27.2.	RESPECT/SURE/CLEANER.....	27-1

### Section 28. SUMLOG Information

28.1.	RESPECT/SURE/LOG.....	28-2
28.2.	Selecting Runinfo from the SUMLOG.....	28-4
28.3.	Cleaning the Runinfo .....	28-6
28.4.	RESPECT/SURE/STARTLOG .....	28-7
28.5.	Analyzing the SURE Run Information.....	28-10
28.5.1.	Run Information of an Individual File .....	28-10
28.5.2.	Run Information of a Group of Files .....	28-10
28.5.2.1.	RUNINFO-LASTRUN.....	28-10
28.5.2.2.	RUNINFO-OVERVIEW .....	28-11
28.5.2.3.	RUNINFO-CHECKRUN .....	28-14
28.5.2.4.	NO-RUNINFO .....	28-15
28.6.	Check the Creation Date of an Object.....	28-16

### Section 29. Software Delivery

29.1.	Deliver Specifications from One Repository to Another Repository .....	29-1
29.2.	Deliver Source Files as an Application System .....	29-2

### Section 30. Initial Load Files

30.1.	RESPECT/SURE/LOAD.....	30-1
30.2.	LOAD using GUI Interface.....	30-4
30.2.1.	Load MCP Sources in SURE .....	30-4
30.2.2.	Load PC Directory of Files in SURE .....	30-5
30.2.3.	Load files in SURE .....	30-6

### Section 31. Information Commands Windows Interface

31.1.	Attachment (Properties/Info) .....	31-1
31.2.	Copy (popup) .....	31-1
31.3.	Compare (popup).....	31-1
31.3.1.	Compare Sources That Come from a Third- Party Vendor .....	31-2
31.4.	Compile (popup, Properties/Info).....	31-3
31.5.	Delta (Properties/Info, Expand).....	31-3
31.6.	Edit/View (popup).....	31-3



31.7.	Errors (popup, Properties/Info) .....	31-4
31.8.	Information (Properties/Info) .....	31-4
31.9.	Integrity (Properties/Info) .....	31-4
31.10.	List (Properties/Info) .....	31-4
31.11.	Log (popup, Properties/Info) .....	31-4
31.12.	Open With .....	31-4
31.13.	Reference (Properties/Info, Expand) .....	31-5
31.14.	Reminder (Properties/Info) .....	31-5
31.15.	Run (popup) .....	31-5
31.16.	Select .....	31-6
31.17.	Expand Browser to Path .....	31-6
31.18.	Status (popup, Properties/Info) .....	31-8
31.19.	Statistics (popup, Properties/Info) .....	31-8
31.20.	Task (Properties/Info, Expand) .....	31-8
31.21.	Write (Properties/Info) .....	31-8
31.22.	Unisys MCP Folder .....	31-8
31.22.1.	Sources (<CANDE Work Usercode>) on <Work-Pack> .....	31-9
31.22.2.	Objects <Directory> on <Pack> .....	31-9
31.22.3.	Batch Reports (<SURE-Batch-Usercode>) on <SURE-Batch-Pack> .....	31-10
31.22.4.	Batch Status Mix: <SURE-Batch-Usercode> .....	31-10
31.22.5.	Printer Output .....	31-11
31.22.6.	Status Mix: <My-Usercode> .....	31-11
31.22.7.	ODT .....	31-11

## Section 32. System and Project Definition

32.1.	System Attributes .....	32-1
32.2.	Project Attributes .....	32-5
32.3.	Hide Directories that Do Not Contain Any Files of My-Project-List .....	32-7

## Section 33. File-Type

33.1.	File Type Classification .....	33-3
33.2.	Object-Usercode/Pack Options .....	33-4
33.3.	Other File Type Options .....	33-6
33.3.1.	Other File Type Options for Mainframe Files .....	33-8
33.3.2.	Other File Type Options for PC Files .....	33-9

## Section 34. Authorization Mechanism and Log On

34.1.	Unique Identification in SURE For Each User .....	34-1
34.2.	The Logon Screen .....	34-2
34.2.1.	Make the Logon Credentials Customizable .....	34-2
34.2.2.	Disable "Save these credentials" at Logon .....	34-7
34.2.3.	Authentication Based On the Windows Account .....	34-8
34.2.3.1.	Anonymous Logon .....	34-10

34.2.4.	Authentication Based On Kerberos.....	34–11
34.3.	Employee Functions .....	34–12
34.3.1.	Employee Function for Routing Purposes.....	34–14
34.4.	Authorization Scheme .....	34–14
34.4.1.	Default Authorization .....	34–15
34.5.	Temporary Authorizations .....	34–17
34.5.1.	Authorizations for Each System/Project .....	34–18
34.6.	Defining Users, Employee Functions, and Authorizations .....	34–20
34.6.1.	User Attributes .....	34–20
34.6.2.	Employee Function Attributes .....	34–22
34.6.3.	Define Authorizations.....	34–22
34.7.	Options of the Authorization System .....	34–23
34.8.	Who is Allowed to Maintain Usercodes and Authorization Maps?.....	34–25
34.9.	Authorization Bits and the Offered Privileges .....	34–26
34.10.	Example Authorization Scheme .....	34–32
34.11.	Overviews .....	34–34
34.11.1.	Authorization Overview.....	34–34
34.11.2.	Overview of Users, Teams, and Projects .....	34–36
34.11.3.	Overview of All the Defined User IDs Plus Attributes .....	34–37

## **Section 35. SURE Options**

35.1.	Global Options.....	35–1
35.1.1.	Global Options, Tab: Task/Global.....	35–1
35.1.2.	Global Options, Tab: SURE.....	35–5
35.1.3.	Global Options, Tab: Security .....	35–11
35.1.4.	Global Options, Tab: History.....	35–13
35.1.5.	Global Options, Tab: SURE compilers .....	35–14
35.1.6.	Global Options, Tab: SURE cleaner .....	35–14
35.2.	Maintain Printer Destinations.....	35–15
35.3.	Maintain Reference Classes.....	35–16
35.4.	SURE Compile Options .....	35–17
35.5.	Compile Options for each Project or System.....	35–21
35.6.	Compile Options for each File Type .....	35–22
35.7.	Transfer Options .....	35–23
35.8.	Resource Definitions .....	35–23
35.9.	Runinfo Usercodes .....	35–24

## **Section 36. Versioning**

36.1.	Terminology .....	36–1
36.2.	Overview and Advice .....	36–3
36.2.1.	Renaming an Environment.....	36–4
36.2.2.	Environment.....	36–5
36.2.3.	Branching of an Environment.....	36–8
36.2.4.	Work-Environment.....	36–13
36.3.	Security .....	36–15

## Section 37. Task

37.1.	Task Function .....	37-1
37.2.	(Task) Organizations Not Using SURE.....	37-2
37.3.	(Task) Tasks and Environments .....	37-3
37.4.	(Task) Type .....	37-4
37.5.	(Task) Definition .....	37-11
37.5.1.	(Task) Definition .....	37-11
37.5.2.	(Task) Attributes .....	37-13
37.5.3.	Alternative Task Definition Screens .....	37-21
37.5.4.	Hyperlinks to an URL or the SharePoint Page .....	37-21
37.6.	(Task) Commands and Click Actions .....	37-22
37.7.	(Task) Authorization Mechanism .....	37-28
37.8.	(Task) Flow of a Task in the EDP Department .....	37-29
37.8.1.	(Task) Validation .....	37-30
37.8.2.	(Task) Routing.....	37-31
37.8.3.	(Task) Assignment .....	37-32
37.8.3.1.	(Task) Assign a task to multiple individual users .....	37-32
37.8.4.	(Task) Adapting the Application System .....	37-34
37.8.5.	(Task) Verification .....	37-35
37.8.6.	(Task) Overlap Analysis.....	37-37
37.8.7.	(Task) Delink.....	37-39
37.8.8.	(Task) Ready .....	37-39
37.8.9.	(Task) Approve.....	37-39
37.8.10.	(Task) Transfer .....	37-42
37.8.10.1.	(Task) Transfer Multiple Selected Tasks as One Unit.....	37-45
37.8.11.	(Task) Reactivation .....	37-46
37.8.12.	Send Email for Assign, Approve, or Transfer .....	37-50
37.9.	(Task) Changing to Status Solved .....	37-52
37.10.	(Task) Master Tasks and Dependent Tasks .....	37-53
37.11.	(Task) Quick Fix .....	37-54
37.11.1.	(Task) Quick Fix Preface .....	37-54
37.11.2.	(Task) Quick Fix Definition .....	37-56
37.11.2.1.	(Task) Quick Fix: Urgent Errors .....	37-56
37.11.2.2.	(Task) Quick Fix: Overlap with Other Tasks .....	37-56
37.11.2.3.	(Task) Quick Fix: Adaptations in a Historical Release.....	37-59
37.11.3.	(Task) Quick Fix Work-Environment .....	37-60
37.11.4.	(Task) Quick Fix Procedure .....	37-61
37.11.4.1.	(Task) Quick Fix: Different Work- Environment.....	37-61
37.11.4.2.	(Task) Quick Fix: Changes Status .....	37-62
37.11.4.3.	(Task) Quick Fix: Reprocess Modifications .....	37-62
37.11.5.	Example: Reprocess Quick Fix Using Check Out .....	37-65
37.12.	(Task) Delivery .....	37-67
37.12.1.	(Task) Deliver Commands .....	37-69
37.12.2.	(Task) Delivery Overlap.....	37-69
37.12.3.	(Task) Delivery Final or Test .....	37-70
37.12.4.	(Task) Delivery Batch Procedure .....	37-71
37.13.	(Task) Overviews.....	37-74

37.13.1.	(Task) Customer Overview .....	37-74
37.13.2.	(Task) Overview "Tasks to handle by ..." .....	37-76
37.13.3.	(Task) Overview "Tasks solved during a period" .....	37-77
37.13.4.	(Task) Overview "Tasks active on a date" .....	37-79
37.13.5.	(Task) Overview "Available tasks" .....	37-80
37.13.6.	(Task) Overview "Tasks transferred from an Env" .....	37-80
37.13.7.	(Task) Overview "Task dump status for each environment" .....	37-81
37.13.8.	(Task) Summary "Reported, solved and still active tasks" .....	37-81
37.13.9.	(Task) Overview "Overtime tasks" .....	37-83
37.13.10.	(Task) Overview "History of a file" .....	37-83
37.13.11.	(Task) Overview of task overlaps .....	37-83
37.13.12.	(Task) RESPECT/TASK/LIST .....	37-85
37.13.12.1.	How to customize task overviews.....	37-88
37.13.13.	(Task) Overview "Various RTF (Rich Text Format) reports" .....	37-90
37.13.14.	Multiple Task Forms Using RTF.....	37-91
37.13.15.	Reporting Using Microsoft Excel.....	37-92
37.14.	(Task) Overview of ID that has to be Tested .....	37-93
37.14.1.	Task-Mode .....	37-94
37.14.2.	Dump-file-mode.....	37-94
37.15.	(Task) Options .....	37-96

### **Section 38. Remove, Recover and Rollback**

38.1.	Remove Logically .....	38-2
38.2.	Restore Logically Removed id .....	38-2
38.3.	Purge .....	38-3
38.4.	Deleted PC Files are Removed on the Build Server .....	38-4
38.5.	Recover From the Next Environment.....	38-4
38.6.	RollBack .....	38-5

### **Section 39. Names Standards**

39.1.	Function .....	39-1
39.2.	(Name-Standard) Implementation.....	39-1
39.2.1.	(Name-Standard) Definition .....	39-1
39.2.2.	(Name-Standard) Syntax .....	39-2
39.2.3.	(Name-Standard) Types .....	39-6
39.2.3.1.	Pre-Defined Name-Standards .....	39-6
39.2.3.2.	Variable Name-Standards .....	39-9

### **Section 40. Repository**

40.1.	(Repository) Environment .....	40-1
40.1.1.	(Repository) Implementation .....	40-1
40.1.2.	(Repository) Location .....	40-2
40.1.3.	(Repository) Why Multiple Repositories.....	40-2

40.1.4.	(Repository) Dump/Load Procedures between Repositories .....	40-2
40.2.	(Repository) Structure.....	40-3
40.2.1.	(Repository) The Structure of the Database (INFDB) .....	40-4
40.2.2.	(Repository) Datasets in Database INFDB .....	40-5
40.2.3.	(Repository) The Use of Relations .....	40-10
40.2.4.	(Repository) Detailed Information about Relations .....	40-13
40.2.5.	RELATION Functionality in SURE Browser .....	40-15

## Section 41. API Interfaces

41.1.	Overview .....	41-1
41.2.	OLE and DDE Interfaces to SURE for Windows .....	41-1
41.2.1.	DDE Interface .....	41-2
41.2.1.1.	DDEExecute .....	41-3
41.2.1.2.	DDEInitiate .....	41-3
41.2.1.3.	DDEPoke .....	41-4
41.2.1.4.	DDERequest .....	41-4
41.2.2.	OLE Interface .....	41-5
41.2.2.1.	CreateObject .....	41-5
41.2.2.2.	SetValue .....	41-6
41.2.2.3.	GetValue .....	41-6
41.2.2.4.	MethodForObject .....	41-6
41.2.2.5.	Connect .....	41-7
41.2.2.6.	Disconnect .....	41-7
41.2.2.7.	InitTab .....	41-8
41.2.2.8.	SetTabName .....	41-8
41.2.2.9.	SetTabValue .....	41-9
41.2.2.10.	GetTabValue .....	41-9
41.2.2.11.	GetTabOccurs .....	41-10
41.2.2.12.	GetTabFieldCount .....	41-10
41.2.2.13.	GetTabFieldName .....	41-10
41.2.2.14.	Example Source .....	41-11

## Section 42. Terminal Emulation Interface

42.1.	Overview .....	42-1
42.2.	Screen Handling for the On-line Functions.....	42-1
42.2.1.	Function Line: Top Line .....	42-3
42.2.2.	Command Line: Second Line .....	42-3
42.2.2.1.	Specify on ITEM .....	42-4
42.2.2.2.	Specify on FILE .....	42-6
42.2.2.3.	Specify on TYPE .....	42-7
42.2.2.4.	Specify on NEXT .....	42-7
42.2.2.5.	Specify on PREV .....	42-7
42.2.2.6.	Specify on FIRST .....	42-7
42.2.2.7.	Specify on LAST .....	42-7
42.2.2.8.	Specify on RELATION .....	42-7
42.2.2.9.	Specify on INFO .....	42-10

42.2.2.10.	Specify on LOG .....	42-13
42.2.2.11.	Specify on HIST .....	42-13
42.2.2.12.	Specify on HELP .....	42-14
42.2.2.13.	Specify on Tasks .....	42-16
42.2.2.14.	Specify on PRINT .....	42-16
42.2.3.	Status Information .....	42-16
42.2.4.	Current Task .....	42-19
42.2.5.	Function Specific Part: Line Three to Twenty- Three.....	42-19
42.2.6.	Error Field: Last Line .....	42-20
42.2.7.	Ctrl Codes.....	42-21
42.2.8.	GO Command.....	42-22
42.2.9.	Printing Information .....	42-22

### Section 43. Windows Interface

43.1.	Logon.....	43-1
43.2.	SURE Explorer .....	43-2
43.2.1.	Explorer Folders .....	43-2
43.2.2.	Hiding Main Folders .....	43-6
43.2.3.	Expanding Objects .....	43-6
43.2.4.	Context-Sensitive Popup Menus .....	43-10
43.2.5.	Windows Layout .....	43-11
43.2.6.	Refresh Strategy .....	43-14
43.2.7.	Clipboard Data.....	43-15
43.2.8.	Tools .....	43-15
43.2.9.	Queues.....	43-15
43.3.	Customizing the SURE Explorer .....	43-16
43.3.1.	Help Menu.....	43-16
43.3.2.	Report Menu .....	43-17
43.3.3.	Email Server.....	43-23
43.3.4.	Multiple Concurrent Versions .....	43-23
43.3.5.	Task Entry Form.....	43-23
43.3.6.	Alternative Task Definition Screens .....	43-24
43.3.7.	Configuring as BUILD Server.....	43-28
43.3.8.	User Pictures .....	43-29
43.3.9.	Open With .....	43-29
43.3.10.	Date Format According to the Regional Settings .....	43-31
43.4.	Keyboard and Mouse Support.....	43-33
43.4.1.	Keyboard Support on Data Entry Windows .....	43-33
43.4.2.	Keyboard Support in the SURE Browser .....	43-34
43.4.3.	Mouse Support.....	43-34
43.4.4.	Drag and Drop Operations.....	43-35
43.4.5.	Explanation of Icons .....	43-36
43.5.	Installation of "backupfile" in MS Word.....	43-38

### Section 44. SURE Datamodel

### Section 45. SURE Process Model

**Section 46. Program Structure**

46.1.	Online Programs (Windows) .....	46-1
46.2.	Batch Programs (Windows).....	46-2

**Section 47. Program Index**

**Section 48. Command Index**

**Section 49. Help Files**

**Section 50. Software Delivery Old Style**





# Section 1

## Introduction

SURE provides life cycle support for software development on Unisys ClearPath Libra Series systems that involve components from multiple operating environments including MCP, Windows, UNIX, and LINUX. With SURE, you can achieve complete control over software development on these platforms.

SURE provides a unique solution for the consolidation and application of repeatable processes across all elements of an enterprise to ensure the integrity of the development process for “composite” applications. Composite applications involve development artifacts from multiple operating environments and require special discipline to achieve business objectives.

SURE is also an excellent tool for software houses, providing support for the development of packages and for the distribution of releases. SURE can also be used by clients who receive packages from software vendors and then apply local changes to customize the release.

The core systems of an enterprise often run on central servers. To enhance and guarantee the quality of this vital functionality, a comprehensive system for software management and change control is essential. Today, many major organizations depend on SURE to maintain the quality and integrity of their “mission critical” systems.

This document describes the Unisys ClearPath SURE solution for process-centric, workflow-enabled software management and change control.

### 1.1. Overview

#### 1.1.1. Today's Requirements

Today's software development and IT organizations are represented by the following roles within a company:

- Developer who gets assigned tasks and who works with popular development tools.
- Tester who approves or denies tasks.
- Team leader who initiates work and assigns it to his team members.
- Project leader who coordinates and monitors projects.
- Release coordinator who manages and controls application systems.
- Management who is responsible for budgeting and procedures.

Now looking into organizations, often different development tools are accompanied with different procedures and tools. Furthermore, different project leaders introduce different procedures for their projects. Local spreadsheets, individual emails, local databases, different task tracking systems, and various project management systems will not benefit the productivity within an IT organization.

SURE originally started as a source control system and has grown to an IT workflow-based system that allows integration of desktop tools and IT procedures. This allows for a common IT workflow within an organization, improving quality and productivity.

### 1.1.2. Improved Quality and Productivity

Currently, many organizations using ClearPath systems do not use a software control management system. Instead, they use their own procedures to archive source modules, control changes, and take software in the production environment. If these procedures are well designed and well implemented, they may seem satisfactory to an organization. However, such procedures are often rigid and restrictive, providing only limited control and functionality and are more consumptive of scarce budget resources. With SURE, you may upgrade the software management and change control services provided on any ClearPath server in order to improve the quality of your software.

This document discusses topics, provided by SURE that helps in increasing the quality and the productivity of IT departments. The SURE product does not prescribe any development methodology like waterfall or rapid application development; it simply helps automating, controlling, and managing the development methods and IT procedures used.

Changes within an IT department are considered as a risk, and therefore, an organization often keeps on working the way they always did unless they are forced to change. Therefore, consider the next list and decide whether it contains beneficial functions.

- Can a developer easily (within a minute) compare sources from different stages and inspect why (for what task, incident, or requirement) changes are made?
- Can you make changes at any stage of the development without complex procedures and without the risk of losing work?
- Can you reproduce the build process at any stage?
- Can you guarantee that an executable correspond to a specific version of a source and inspect the reasons why it was changed?
- Does your current mechanism produce impact analysis and does it detect impact overlap?
- Can two developers work concurrently on the same source without the risk of losing work?
- Is the turnover procedure fully automated, audited, and without any paper work?

- Does your current mechanism provide spreadsheets that can be used by project leaders allowing them to verify and discuss the status of a project?
- Is the release letter generated by your current mechanism?

Most of the SURE customers were using a turnover paper prior to their implementation on SURE. In general, every turnover paper takes more than 2 hours of processing for someone to fill and update it, route it to the authorized persons who need to sign the form and follow up the process. This is beside the lack of auditing and the additional time required when things must be done quickly. With SURE, the turnover process is totally automated and electronically coordinated.

### **1.1.3. Unique to Unisys ClearPath MCP**

The MCP operating system provides a variety of functions specific to the architecture of the ClearPath Series. Therefore, generic software control management systems can only provide limited support for the MCP platform. SURE is a unique product for Unisys ClearPath Series in that it provides functionality tailored and optimized to run on the MCP operating environment.

SURE seamlessly integrates with the MCP security system, file system, disk pack organization, networking facilities, usercode structure, SUMLOG, and other ClearPath Libra Series specific mechanisms.

### **1.1.4. Support for Distributed Environments**

Not only does SURE allow control and management of MCP software changes and projects, but it also works in controlling Windows, UNIX, and LINUX environments. As organizations progress towards implementation of Web-based extensions to their core business applications, this capability becomes increasingly vital. And, if there is contemplation of a Services Oriented Architecture involving the integration of many software components from different technologies, SURE is a mandatory solution.

### **1.1.5. Integrates with IDE's such as Eclipse and Microsoft Visual Studio**

SURE supports every type of file on the WINDOWS or UNIX operating system environment. Furthermore, a seamless integration is achieved between SURE and IDEs such as Java Eclipse and Microsoft Visual Studio. This integration allows developers to use the SURE functions out of the IDE, meaning that they can check out and check in files from menu choices within these IDEs. Therefore, they do not need to switch an application that eases the developer activities.

## 1.2. Features

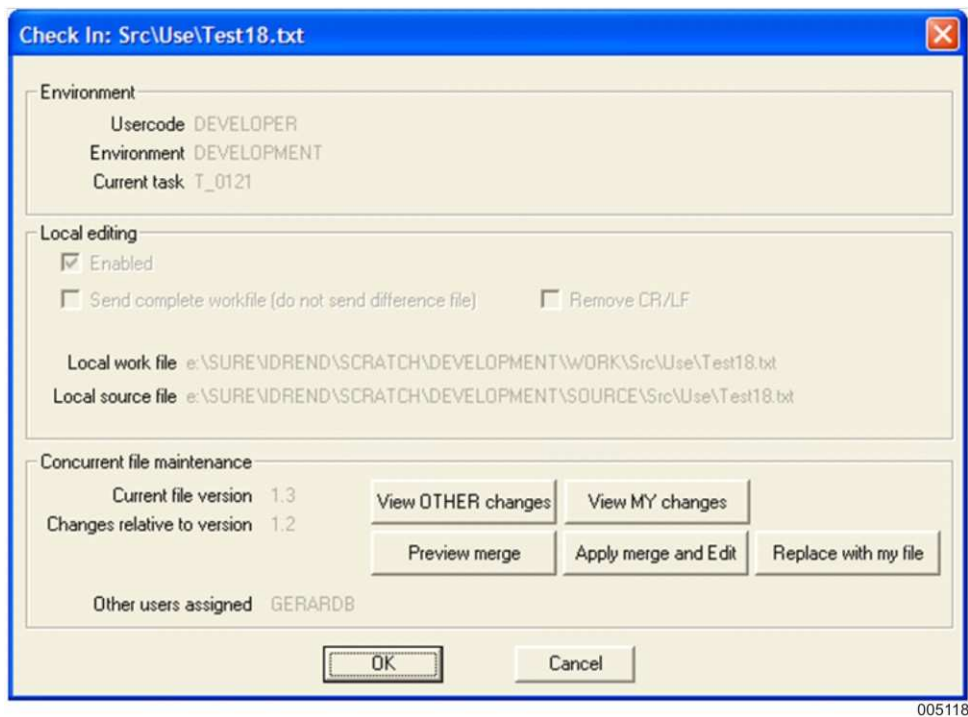
SURE offers the following features to simplify software management tasks.

### 1.2.1. Flexible Source Control System

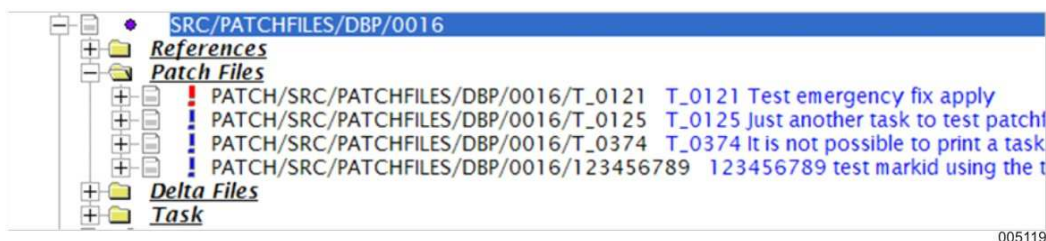
#### Parallel Development

SURE contains a flexible source control system that allows concurrent source maintenance. For PC or WINDOWS based files, SURE supports concurrent source maintenance where it integrates “other changes” in the source. This mechanism avoids rework and errors. For MCP files, SURE supports PATCH mechanism that allows parallel development and total control of deployment.

The following figure illustrates the Check In dialog that allows for concurrent source maintenance and support.



The following figure illustrates the SRC/PATCHFILES/DBP/0016 file (containing patch files) that allows parallel development and different deployment schedules.



### Information Attached to a Source

The source control system allows connecting various information to a source. This varies from notepad style documentation to zip files, containing complete analysis in Word and Visio.

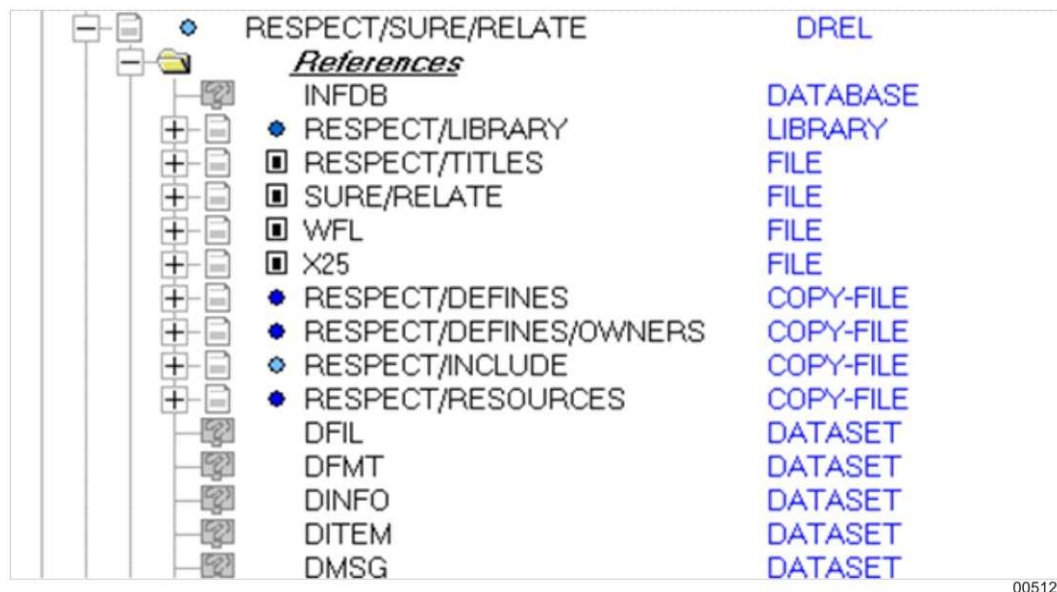
The following figure illustrates a file containing an attachment and a reminder.



### Dynamic Cross Reference

The source control system is the foundation of the SURE system. This source control system is built on top of a true repository system that keeps track of all the source-related information. This includes a dynamic build cross-reference between sources and data structures, stored as references. For example, a database administrator uses this information to issue a compile command for all sources using a specific dataset.

The following figure illustrates a RESPECT/SURE/RELATE source file with its references.



### Delta Files and Differences

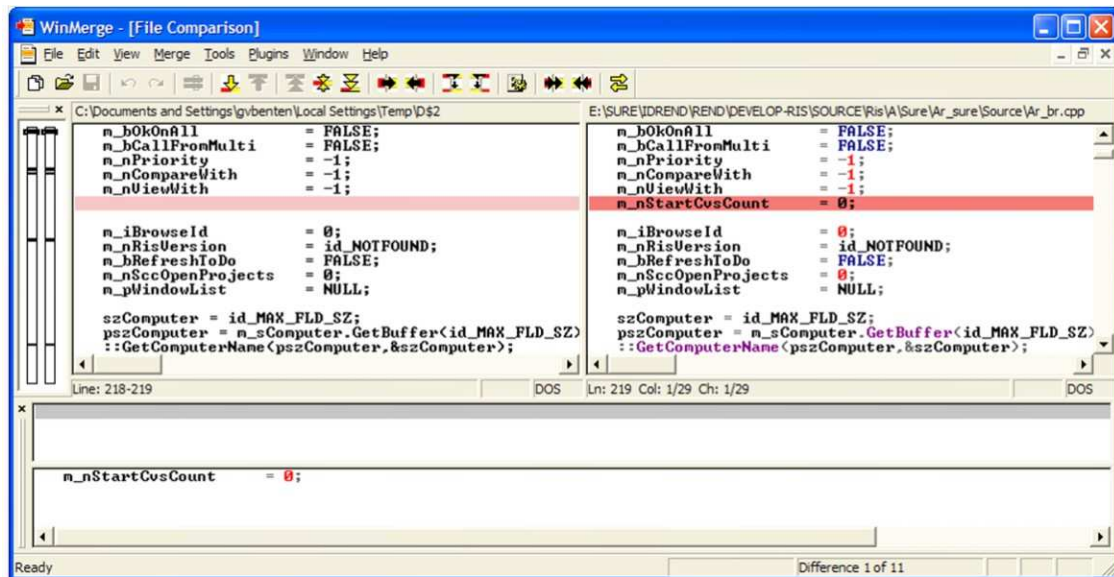
The source control system uses a dynamic history configuration that allows a company to define how long different historical objects, such as delta files, need to be archived. These delta files allow for graphical comparisons of files and concurrent source maintenance. Furthermore, SURE has an inbuilt procedure for hot fixes on "production" software.

The following figure illustrates a file containing delta files used for graphical comparison or rollback purposes.



005122

The following figure illustrates the graphic compare utility.

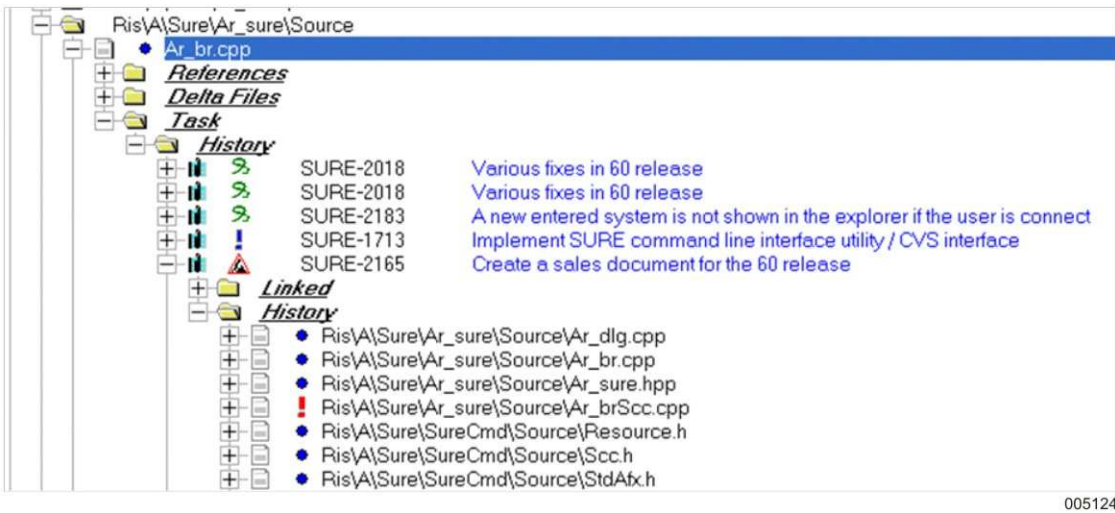


005123

Integrated Task History

The source control system is integrated with the task tracking system. Therefore, a task history is maintained.

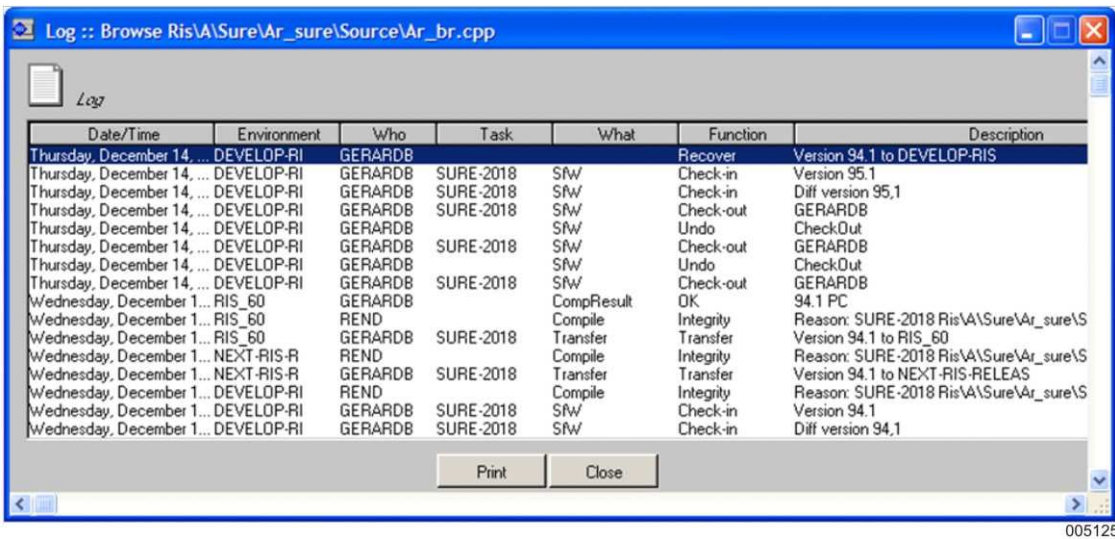
The following figure displays the Ar\_br.cpp file that contains a set of tasks indicating why the file was changed. It also lists under the History folder all the files that are changed for the task SURE-2165.



Audit Log

The source control system uses extensive logging for audits. This logging answers questions about who changed what, why, and when.

The following figure illustrates the command log on a file used for auditing purposes.



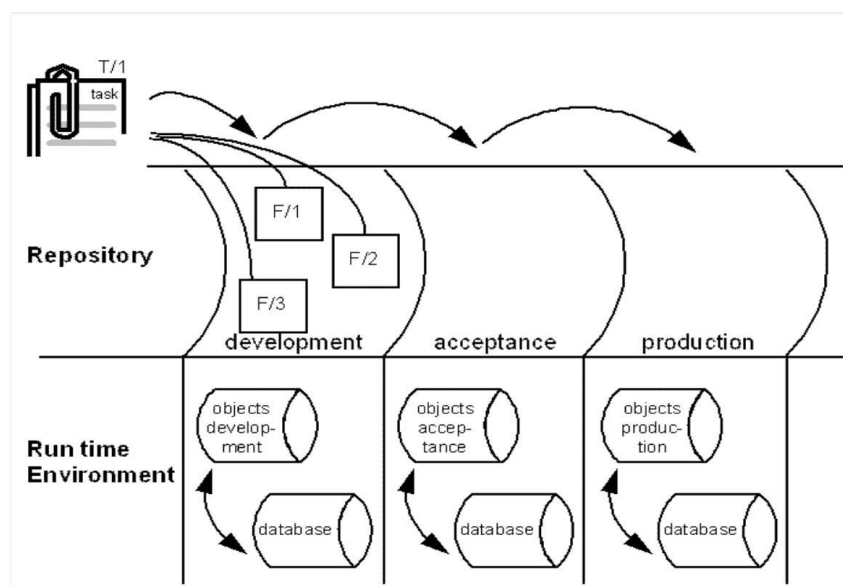
### 1.2.2. Integrated Task System

SURE contains an integrated task (incident or requirement) system that drives and enforces the procedures. An organization uses an incident management system and a requirement registration or planning system. The SURE task system contains basic functionality that is also present in the systems that are listed previously. If you do not have these systems in place, you could use the SURE task functionality. If you have these systems in place and you are satisfied, then you can link the SURE task system to these external systems.

The advantage of integration is that SURE connects all the change sources and software components to a task that identifies an incident or a requirement. This task can then be promoted to another stage in development that implicitly promotes the change sources or software components, starts the build process, and deploys the new software. For this reason, you can release incidents or requirements. This integration will automatically update the status of the tasks that eliminate the need for project leaders to manually update local spreadsheets.

- Integrated task administration in order to group changes together: This implicitly allows “non-technical personnel” to release tasks into production. A task in SURE describes a logical unit of work.
- A provision for multiple distinct environments, each containing a complete software system that matches the status of that environment, allowing quality assurance of production software to be made independent of development: An environment represents a “slot” or stage of the application development cycle (development, test, acceptance, production), and SURE has a provision for up to eight separate environments, but typically most clients require far fewer.

The following figure illustrates SURE with three configured environments: development, acceptance, and production. Each of these environments contains its source modules in a “repository slot,” with the resulting objects and matching database(s) in the corresponding run-time environment.





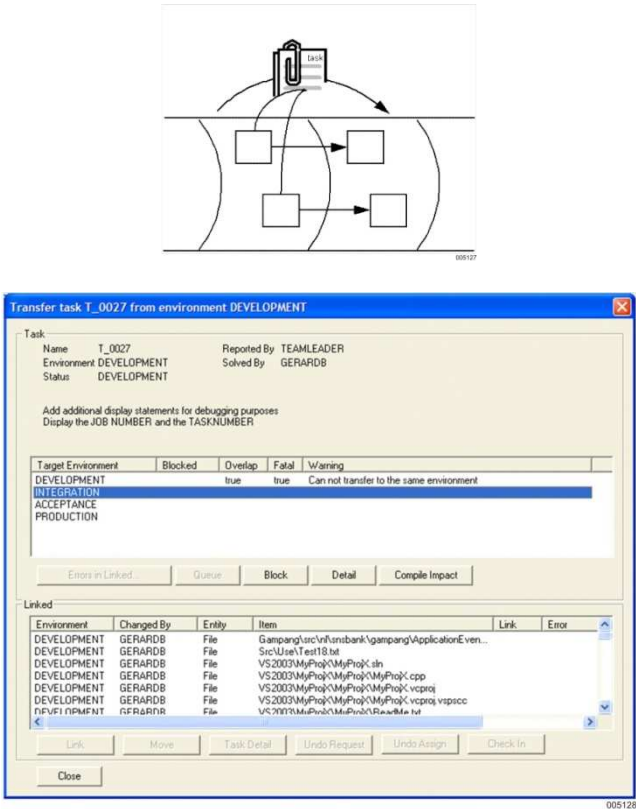
A task identifies not only a project, but also the associated developers. Thus, defining a task will implicitly define the workload for the various developers working on that project. A task can represent the work needed for any project or incident report that needs to be resolved. Tasks in SURE are hierarchical, and therefore, linked to represent any degree of project scope.

Workflow between the stages of the development process occurs when tasks are moved to another slot. SURE has inbuilt queues and other mechanisms that make it easier to facilitate workflow automation.

Task Promotion

If a developer checks out a source file, SURE automatically creates a link between the task and the source file, enabling maintenance of a complete history of multiple developers checking out and working on multiple source modules. Transferring the changes to another environment (for example, acceptance) is performed by transferring the task to the new environment. Therefore, the task is the actual unit of work and the sole mechanism to control changes. In SURE, tasks are the managed unit, and developers and administrators do not manage files. It is the responsibility of SURE to record changes in files in the task. SURE can then maintain a complete picture of what needs to be transferred to the next slot, relieving individuals of this responsibility. This results in much higher productivity as developers can concentrate on creating new business functionality and leave the administration details to SURE.

The following figure illustrates the Task promotion dialog that contains all the linked changes.



## Introduction

In doing so, the task is the driving force within the IT turnover procedure. That is not different from the way normally worked at most organizations; however, the turnover procedure is normally a piece of paper with a lot of manual activities. While in SURE it is a fully automated process with mechanisms for approval by different employee roles and maintenance of an electronic journal for auditing purposes.

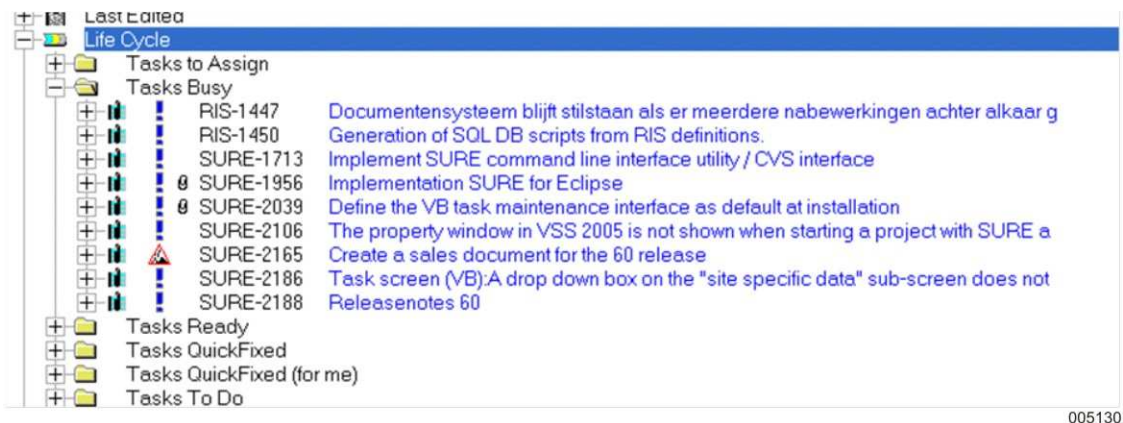
The following figure illustrates the part of the TASK popup menu that allows different functions, such as assign, close, ready, priority, and approve, according to the status in the workflow and the role of the employee.



## Workflow Folders

SURE groups these tasks in different workflow folders that allow different employee roles to view a list of tasks in a status of their interest. These lists can be imported in any Microsoft tool, such as Excel, for communication purposes. Often these lists are used by project or team leaders during their weekly meetings. This avoids time spend by the project or team leaders for maintaining similar spreadsheets.

The following figure illustrates the different workflow folders that SURE maintains, tailored for every employee.



## Extract Task Information in Excel

The following figure illustrates the SURE list where the information is extracted to Microsoft Excel.

	A	B	C	D	E	F	G
1	Report:	Active tasks of type			Make report		
2							
3	Argument	Value	Field name				
4	Task type	ERROR	TaskType				
5	Environment	DEVELOP-RIS	Environment				
6							
7	Macro text						
8		PROBLEM-TYPE(<<TaskType>>) AND STATUS(<<Environment>>)					
9							
10	Task Name	Short description	Status	Assigned	Entered	Priority	Delivery
12	RIS-1447	Documentensysteem blijft stilstaan	DEVELOP-RIS	SIMON	20061101		
13	SURE-2106	The property window in VSS 2005 is	DEVELOP-RIS	GERARDB	20060905		
14	SURE-2165	Create a sales document for the 60 r	DEVELOP-RIS	GERARDB	20061122		
15	SURE-2186	Task screen (VB):A drop down box o	DEVELOP-RIS	GERARDB	20061204		

005131

## Customized Reports

SURE allows creation of customized reports; therefore, a paper version of the turnover form can easily be created and used in the transition phase to SURE.

The following figure illustrates the task report that can be customized to specific installations.

12/18/06 3:53 PM SURE task report : SURE-2039

**Task** **SURE-2039**  
Define the VB task maintenance interface as default at installation

**Attributes**

Status	DEVELOP-RIS
Environment	DEVELOP-RIS
Reported by	GERARDB
Project	SURE
Department	R&D-TEAM
Group	SURE/FEATURE
Type	FEATURE

**Status**

Handled By	GERARDB
Entered at	2006/05/10
Opened at	2006/12/08
Ready at	2006/12/04
Assigned to User	GERARDB

**Description :**

Define the VB task maintenance interface as default at installation

**Linked entities :**

DEVELOP-RIS	Ris\A\Sure\TaskMaintenance\MSSCCPRJ.SCC
DEVELOP-RIS	Ris\A\Sure\TaskMaintenance\SiteForm.fm
DEVELOP-RIS	Ris\A\Sure\TaskMaintenance\TaskMnt.vbp

005132

### Customizable Task Form

SURE contains a customizable Visual Basic program for task registration and maintenance. This, combined with the custom definable task properties, allows you to customize the task maintenance according to the customer requirements.

The following figure illustrates the Customizable Visual Basic form for specific installations.

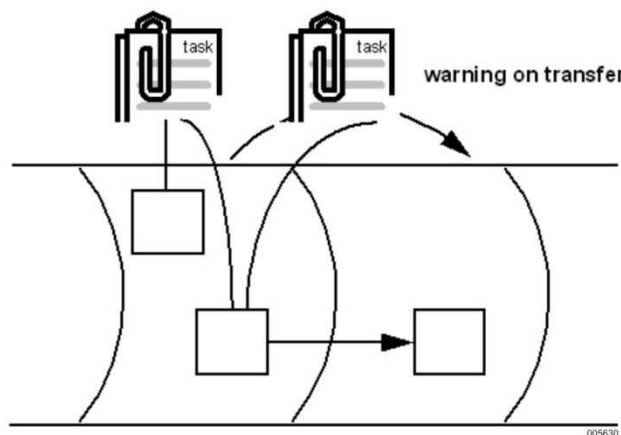
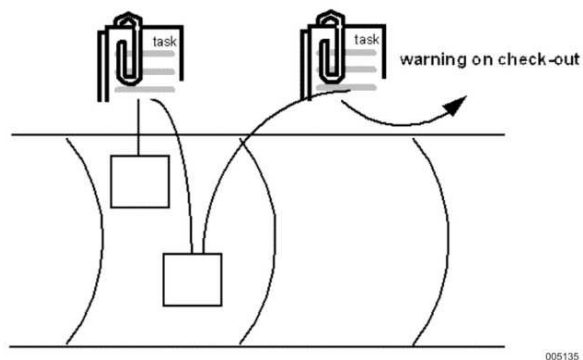
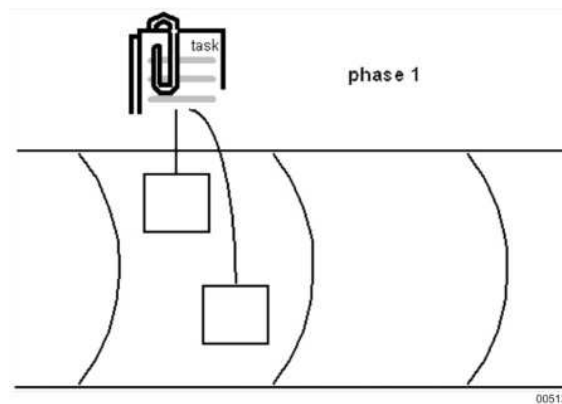
The screenshot displays a Windows application window titled "SURE\VDREND\REND - SURE-2039". The form is divided into several sections:

- Top Section:** Includes a dropdown for "The selected task", a "SURE status" dropdown set to "DEVELOP-RIS", and a "Master task" dropdown.
- Identification Section:** Contains input fields for "Identification" (filled with "SURE-2039"), "Project" (filled with "SURE"), "Type" (filled with "FEATURE"), "Priority", "Compile Queue", and "Resources".
- Reporting Section:** Includes a "Reported by" dropdown (filled with "GERARDB"), an "Entered" date field (filled with "5/10/2006"), and an "Effort estimated" field.
- Approval Section:** Features a "Delivery" field, an "Extended Data" button, a "Site Specific Data" button, and a line for "Approved by".
- To handle by Section:** Contains radio buttons for "I start working for this task", "Not assigned", "Usercode" (selected), "Function", and "Team". It also includes a "Usercode" dropdown (filled with "GERARDB") and checkboxes for "Inform assignment via email" and "Inform status change via email".
- Information Section:** Includes radio buttons for "Description" (selected), "Solution", "Documentation", and "Reminder". A text area below contains the text "★ Define the VB task maintenance interface as default at installation".
- Bottom Section:** Contains buttons for "New", "Ready", "Approve", "Apply", and "Close". To the right is an "Attachment" label with icons for a folder, a document, and a printer.

005133

### 1.3. What is an Overlapping Task?

The relationship between task developers and task source files is “many-to-many.” For this reason, it is possible to modify a source file for multiple different tasks. If one of these tasks is transferred to another environment, any dependent source file that might contain changes for other tasks is also transferred to that environment. The SURE system allows transfer of such “overlapping tasks” but warns a programmer when he checks out a dependent source file. It also warns any employee transferring the task.



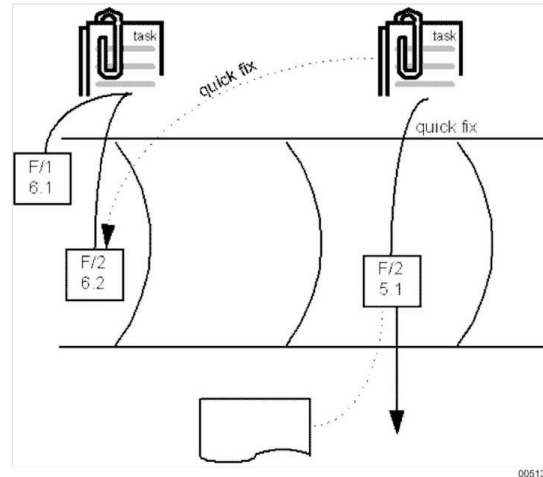
### 1.3.1. What is Quick Fix?

Overlapping tasks result in a program that contains changes for different tasks. However, there may be situations where this overlap is unacceptable. For example, in the case of solving fatal production errors, it must be possible to get rid of the fatal error without taking any unwanted new features into production. SURE notifies the developer when such changes have occurred.

By default, developers make software changes in the lowest hierarchical environment and then transfer these changes to higher environments (for example, from development to acceptance to production). In the case of a quick fix, a different approach is used wherein a developer makes the changes in a higher environment. SURE then reports to the lower environments that these changes have been made. By ensuring that changes that occur in higher level slots are reported at lower levels, SURE avoids a common problem of having the same error recur because an emergency fix in production was not applied to new versions in the development.

A quick fix is the ability to make changes to source modules in a higher environment without disturbing the development process for production environments or releases. The SURE system provides the appropriate implementation and feedback mechanisms to support quick fixes. SURE also provides a mechanism to merge patches in a lower environment.

The following figure illustrates an example of a quick fix.

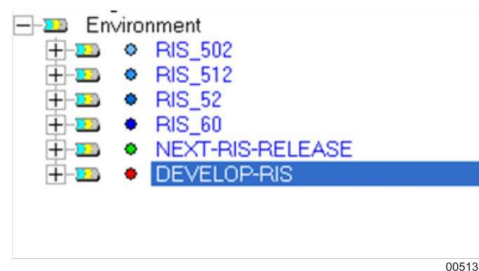


The file F/2 was modified in development. A production error required modification of the file F/2. However, during the check-out procedure, the system indicated overlapping tasks and offered a quick fix option. This option allowed the file F/2 to be changed in production so that the actual production version was fixed without the new functionality. Thereafter, the file F/2 shows the presence of a quick fix, and it is also reported in a batch listing. It is likely that the changes to F/2 made in production also need to be incorporated into F/2 in the development. After this incorporation, the quick fix notification disappears.

## 1.4. Configurable Environments

SURE allows for a configurable amount of environments (stages such as develop, test, acceptance, and production) that match the requirements described in the procedures within a company. An environment contains all software components for a company. If a company owns different application systems, each with different versions, then an environment contains the specific versions of the application systems in that stage. An environment can also be considered as a predefined company baseline when using other source control system's terminology.

The following figure illustrates the environments for a software supplier supporting historical releases.



The following figure illustrates the environments for an organization supporting ongoing software development.



### 1.4.1. Configuring Workflow

What is the difference between baselines created after finishing software and predefined company baseline? Using predefined company baselines allows defining the workflow, the authorization, and the approval process for different stages. For example, you can configure that an employee role named "release manager" must approve a task before it is deployed to the environment named "production." Therefore, by defining the workflow, authorization and approval, the company procedures are automated, audited and enforced by the SURE software.

The following figure illustrates the task type that allows specific workflow procedures. It also defines which role has to approve in which environment.

Task type DailyChanges

Task type

Type: DailyChanges

Task type | TransferTask | Approval | Assignment | Add | Miscellaneous

Environment	Approve task Role
ACCEPTANCE	ReleaseCoordinator
INTEGRATION	TeamLeader

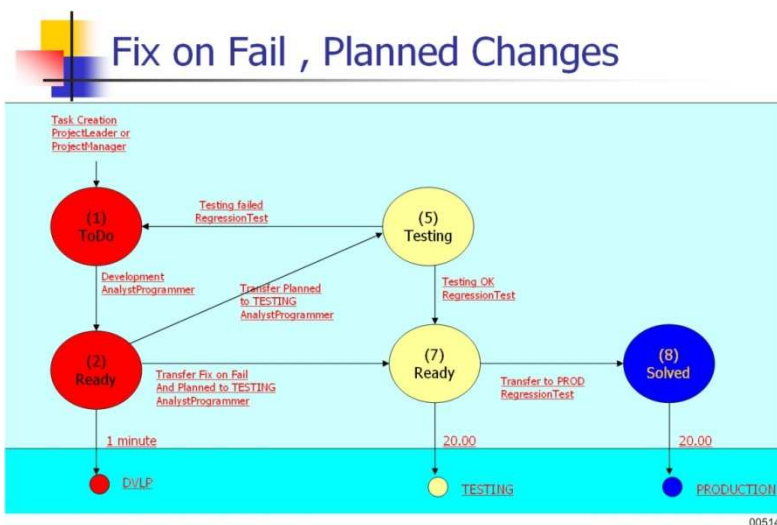
Apply | Insert Line | Delete Line | Print | Close

005139

### 1.4.2. Workflow Diagram

Using this configurable model allows creating a workflow per task type that exactly defines how the procedure is implemented within an organization. These workflow diagrams are used as the help information for SURE users.

The following figure illustrates the simple workflow diagram that shows the activities and the roles performing different functions.





## 1.5. Fully Automated Build Process

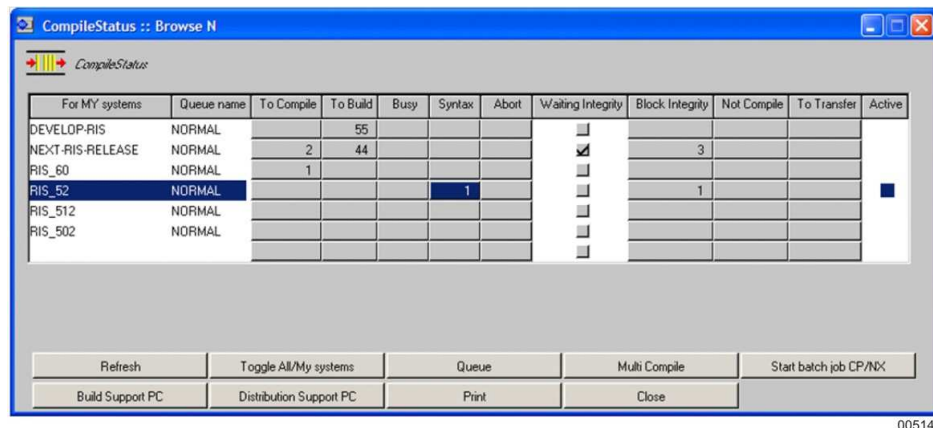
Software development projects often depend on the skills and creativity of individuals. However, this creativity should be concentrated on the development of the software, not on the control of it. SURE provides control measures to protect any project from dependency on individuals for this function. One such measure is the unique registration in the repository of file locations, security aspects, non-standard object names, or object characteristics (for example, privileged programs). The same applies to other aspects, such as program binding or any special action upon that SURE compile software acts afterwards.

SURE allows a completely automated build process based on the information in the repository. This build process initiated with a single mouse click may produce MCP object files, PC executables, and UNIX executables, all controlled from the central SURE repository. All the information about these builds such as timestamps and error reports are stored in the SURE repository and accessible from the SURE explorer.

### Daily Build Process

The review of some Microsoft papers about the software development process in their factory speaks about a "daily build" and the follow up on this daily build that identifies the sources failing in this daily build. SURE supports this daily build process for every defined environment based on the changed files or other software components. This means that SURE controls, starts, and logs the build process so that this information is always available and accessible from every workstation.

The following figure illustrates the compile console of SURE that holds all files changed for different environments and that trigger compilation.



The screenshot shows a window titled "CompileStatus :: Browse N". Inside, there is a table with columns: "For MY systems", "Queue name", "To Compile", "To Build", "Busy", "Syntax", "Abort", "Waiting Integrity", "Block Integrity", "Not Compile", "To Transfer", and "Active". The table lists several build tasks, with "RIS\_52" selected. Below the table are buttons for "Refresh", "Toggle All/My systems", "Queue", "Multi Compile", "Start batch job CP/NO", "Build Support PC", "Distribution Support PC", "Print", and "Close".

For MY systems	Queue name	To Compile	To Build	Busy	Syntax	Abort	Waiting Integrity	Block Integrity	Not Compile	To Transfer	Active
DEVELOP-RIS	NORMAL		55				<input type="checkbox"/>				
NEXT-RIS-RELEASE	NORMAL	2	44				<input checked="" type="checkbox"/>	3			
RIS_60	NORMAL	1					<input type="checkbox"/>				
RIS_52	NORMAL			1			<input type="checkbox"/>	1			<input checked="" type="checkbox"/>
RIS_512	NORMAL						<input type="checkbox"/>				
RIS_502	NORMAL						<input type="checkbox"/>				

Buttons: Refresh, Toggle All/My systems, Queue, Multi Compile, Start batch job CP/NO, Build Support PC, Distribution Support PC, Print, Close.

Today's software systems have a modular character, implicitly making many software modules dependent on one another. For example, checking in a copy file requires recompilation of all modules using that copy file. Only through this process, we can maintain the integrity of the application system.

# Introduction

The repository used by SURE has a “self-learning” mechanism with regard to dependencies between source modules. Each time a file is checked into the SURE environment, parsing by an internal syntax scanner takes place (based on the file kind), check and SURE automatically stores all relevant dependencies in its repository.

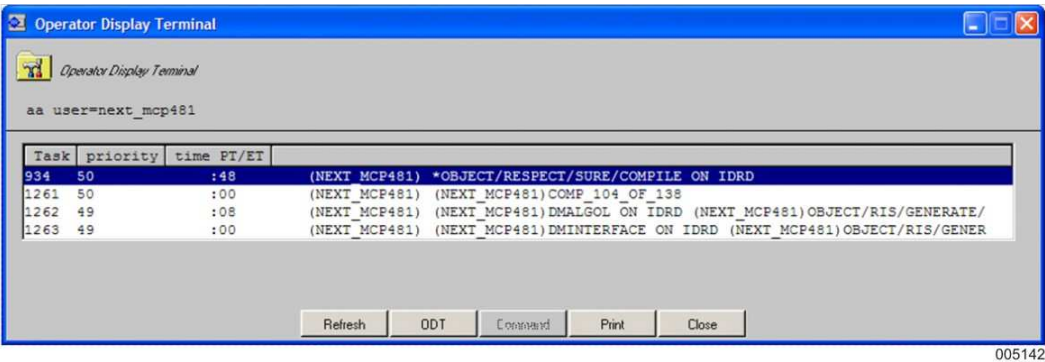
Except this self-learning mechanism, SURE supports a manual dependency declaration that is often used for multiple platform development with common interfaces.

The information extracted from source modules is used to maintain the integrity of the application system. Therefore, checking in a copy file will result in the compilation of all modules using that copy file. Of course, the status in one environment (for example, development) may be different from that in another environment (for example, acceptance). In contrast to manual processes, SURE maintains these environments electronically, assuring visibility of all stages of the development processes.

## Build MCP

In the SURE software, the build for Unisys MCP is fully automated with a minimal effort for the configuration.

For MCP, a program compiles all change files. If a copy or include file is changed, all related files are compiled.

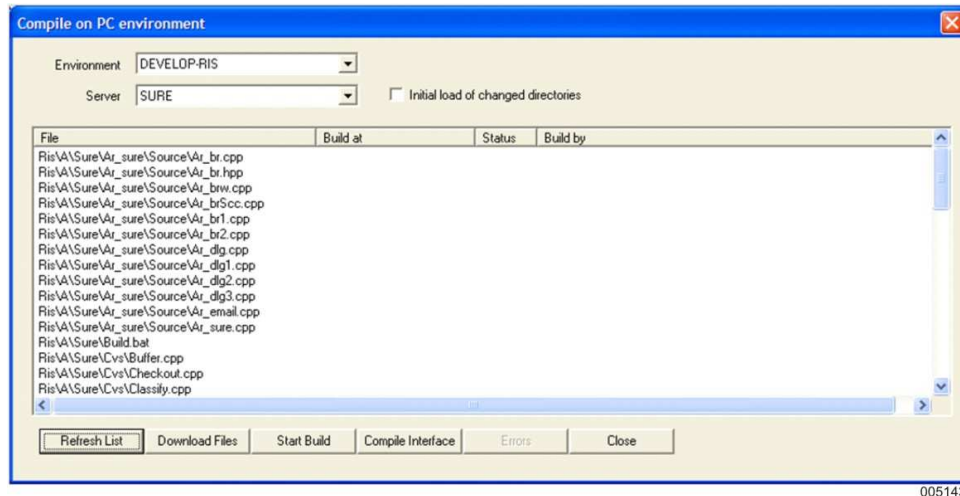


005142

## Build WINDOWS or UNIX

For Windows and UNIX, SURE uses the current build procedures using Make, Nmake, (N)Ant, or any other scripts in place. SURE can easily handle a project-based setup and allow for generic script files.

For WINDOWS or UNIX files, SURE runs scripts that are connected to logigac file types or build.bat files. These scripts can access any MAKE, NMAKE, ANT, NANT, BAT, or SHELL script. The errors or results are reported back in the repository.



005143

The end result of the build and deployment configuration is that everyone (authorized) can start the build process with a single click (or time scheduled) and everyone can see the results and errors if applicable.

## 1.6. Modernized MCP Development

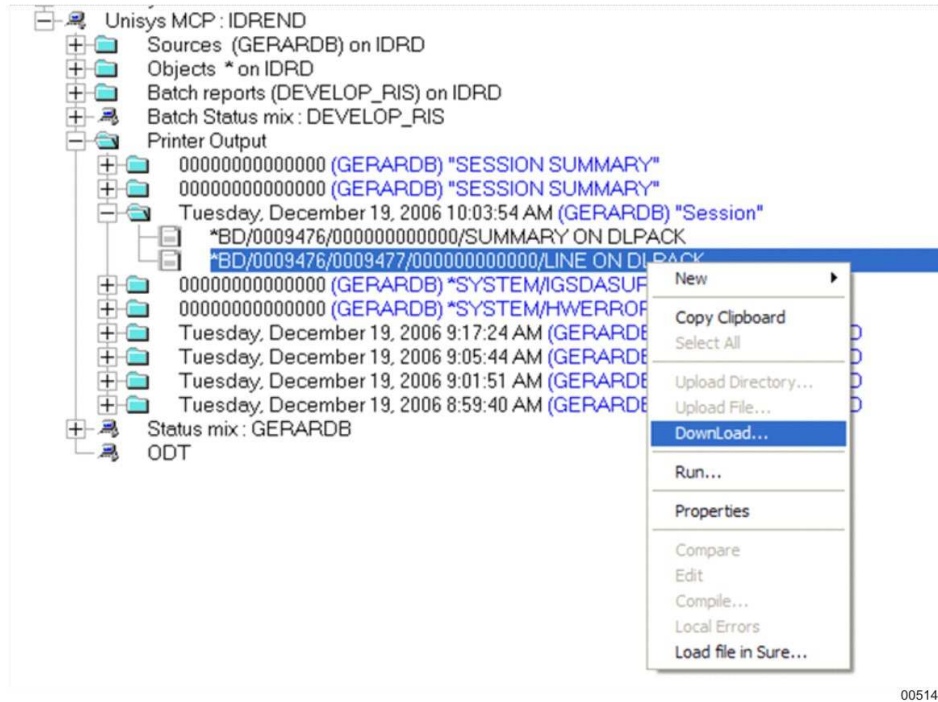
As stated before, many organizations run their core systems on MCP servers. Modernizing this development environment increases the lifetime of these systems. Therefore, writing off software application systems in 15 years instead of 10 years will save a substantial amount of money.

### Workbench

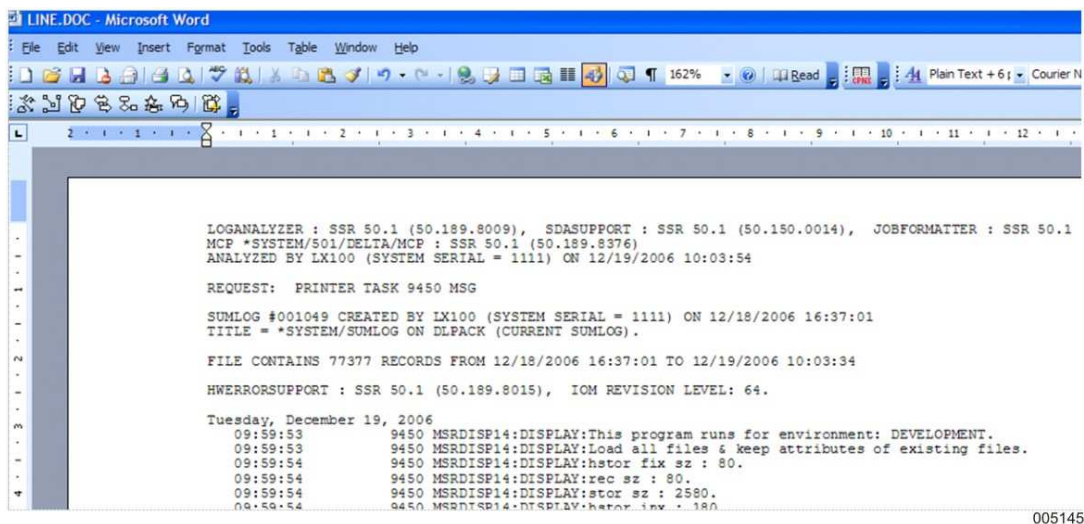
The SURE software contains a workbench that allows viewing the status on the MCP system, downloading and viewing print files, and editing source files using PC-based editors.

## Introduction

The SURE MCP workbench provides easy functions for a developer. The following example shows how to open a printer backup file in Word from the SURE MCP workbench.



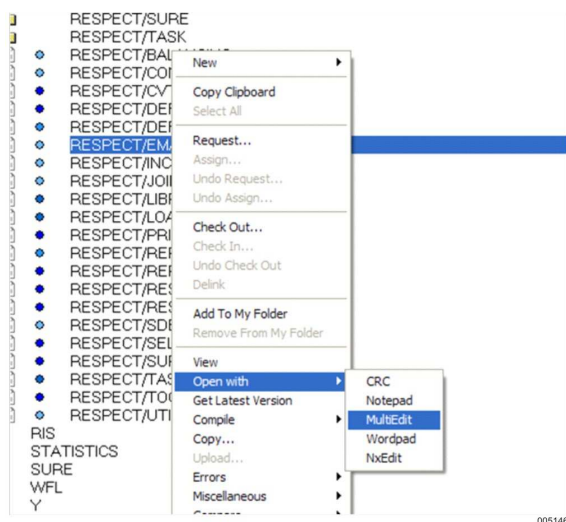
The following example shows a printer backup file in Word. The SURE MCP workbench streamlines the process of opening this file.



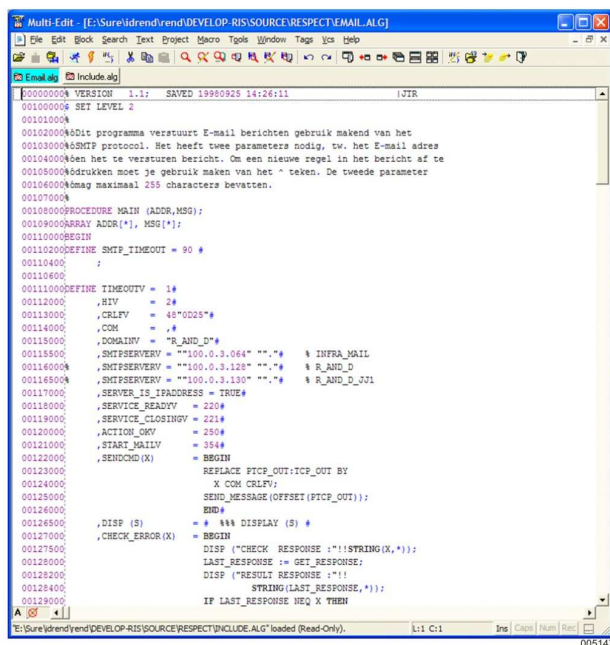
## Windows Based Editing

The SURE software allows for the coding of source software using industry standard editors. The programmer is given the choice of using Unisys Programmer Workbench or a priced item such as MultiEdit from American cybernetics. Using a PC-based editor, SURE takes care of the required file transfers and the synchronization with the MCP file system.

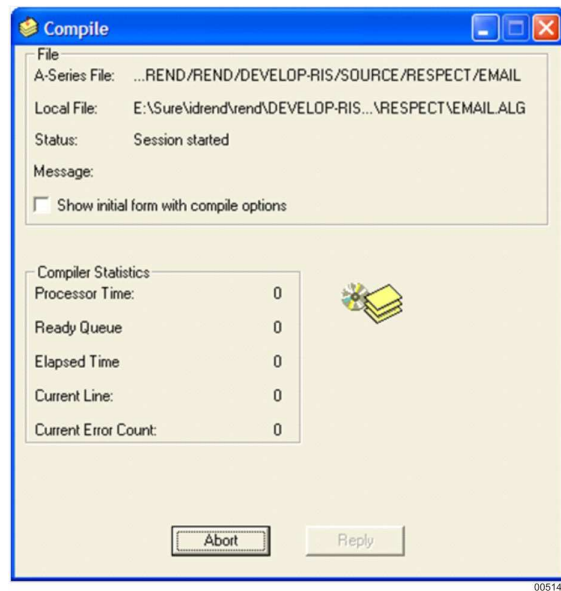
The SURE software allows opening an MCP file using any configured editor on the PC. The SURE software downloads and converts the file so that it is easy to maintain in this PC editor.



Editing an MCP file in a commercial editor allows opening multiple files and making use of syntax coloring facilities.



The SURE software includes utilities that allow initiation of MCP compilation and error feedback in the editor, making the developer's work easy.



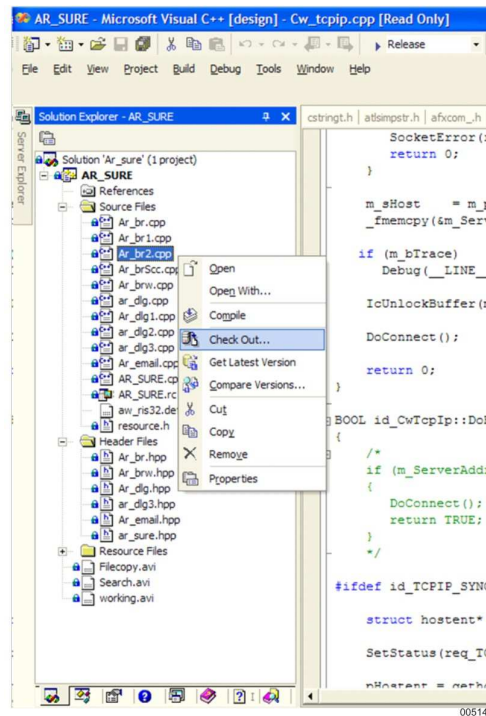
## 1.7. Seamless Integration with IDEs

SURE supports a seamless integration with the current main IDEs, such as Microsoft Visual Studio and Java Eclipse, and the Relativity tool. The current mainstream of development consists of Microsoft Visual Studio .NET and Java Eclipse listed in random order. Furthermore, there are developers using various different flavors of editors such as MultiEdit, NXEdit, Notepad, and Unisys MCP editing. SURE offers a seamless integration with all of these development tools. This allows the developers to check out and check in the files from their IDE without switching to other applications.

### Microsoft Visual Studio

For Microsoft .NET, SURE uses the Microsoft SCC interface for communication. Using this interface allows the use of Microfocus COBOL, Powerbuilder, and Visual Basic tools.

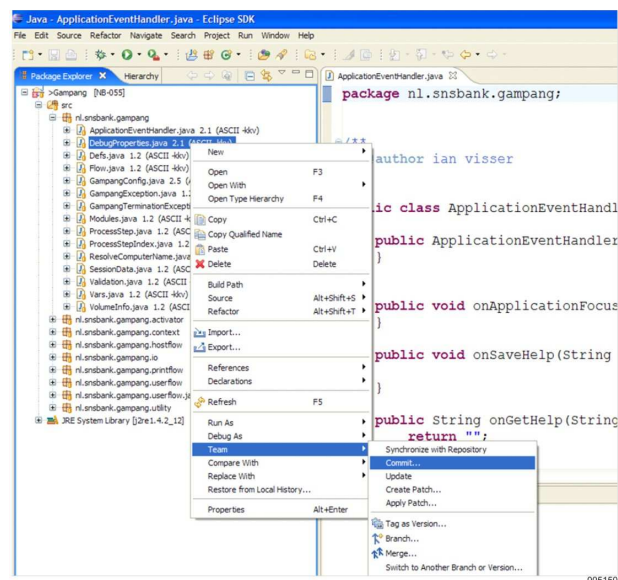
Microsoft Visual Studio links up to the SURE software using the SCC interface. This allows the developer to access the SURE function from within Microsoft Visual Studio.



## Java Eclipse

For Java Eclipse, SURE uses the CVS pServer protocol. Currently, this interface is available for Eclipse versions 3.1.2 and 3.2.

Java Eclipse links to the SURE software using the CVS pServer protocol. This allows the developer to access the SURE function from within the Eclipse software.

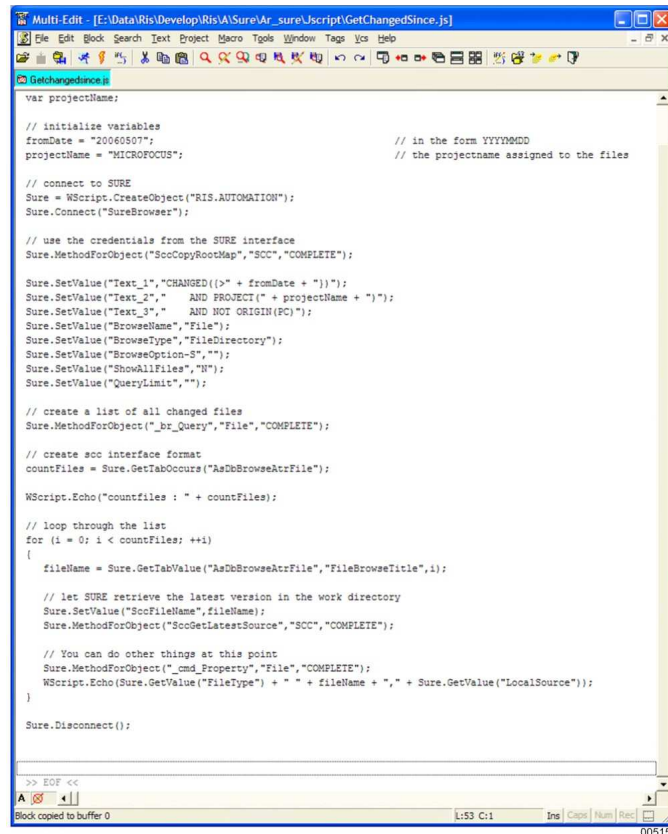




### Relativity

Furthermore, allows SURE interaction using VB script or JAVA script using an exported object model. External tools, such as Relativity, use this interface for integration purposes.

SURE supports an OLE interface that allows calling all functions within the SURE interface. This small Java script retrieves all the files that are changed since the last run.



```
var projectName;

// initialize variables
fromDate = "20060507"; // in the form YYYYMMDD
projectName = "MICROFOCUS"; // the projectname assigned to the files

// connect to SURE
Sure = NScript.CreateObject("RIS.AUTOMATION");
Sure.Connect("SureBrowser");

// use the credentials from the SURE interface
Sure.MethodForObject("SocCopyRootMap", "SOC", "COMPLETE");

Sure.SetValue("Text_1", "CHANGED(>" + fromDate + ")");
Sure.SetValue("Text_2", " AND PROJECT(" + projectName + ")");
Sure.SetValue("Text_3", " AND NOT ORIGIN(PC)");
Sure.SetValue("BrowseName", "File");
Sure.SetValue("BrowseType", "FileDirectory");
Sure.SetValue("BrowseOption-5", "");
Sure.SetValue("ShowAllFiles", "N");
Sure.SetValue("QueryLimit", "");

// create a list of all changed files
Sure.MethodForObject("Br_Query", "File", "COMPLETE");

// create soc interface format
countFiles = Sure.GetTabOccurs("AsDbBrowseAttrFile");

NScript.Echo("countfiles : " + countFiles);

// loop through the list
for (i = 0; i < countFiles; ++i)
{
    fileName = Sure.GetTabValue("AsDbBrowseAttrFile", "FileBrowseTitle", i);

    // let SURE retrieve the latest version in the work directory
    Sure.SetValue("SocFileName", fileName);
    Sure.MethodForObject("SocGetLatestSource", "SOC", "COMPLETE");

    // You can do other things at this point
    Sure.MethodForObject("cmd_Property", "File", "COMPLETE");
    NScript.Echo(Sure.GetValue("FileType") + " " + fileName + " " + Sure.GetValue("LocalSource"));
}

Sure.Disconnect();

>> EOF <<
Block copied to buffer 0
L:53 C:1
Ins Caps Num Rec
005151
```

## 1.8. Transitioning to SURE

Most organizations have developed custom processes for managing development projects. Successful projects are defined within an environment that has evolved over many years. Therefore, there is reluctance to disturb what has been working for a long time. New framework represents business risks that should be accessed before undertaking a transformation.

With SURE, the approach to this transformation mitigates most of the risk of change. SURE is a flexible architecture for application development. SURE adapts the client's existing processes and provides complete automation for any manual procedures. Every SURE installation is completely customized to the unique requirements of each organization. In all but the very largest development organizations, the transition to SURE is measured in days and weeks, not in months.



## Section 2

# **Software Management and Change Control**

### **2.1. The First Version of SURE**

SURE was originally designed to track the changes and to prevent the loss of changes while modifying the copies of source modules. In 1983, the first version was released and was designed for Unisys CP/NX Series systems. Since then, SURE has controlled the performance of development environment. For example, check-in and check-out procedures, task registration, control of compilations, and the transfer of objects.

### **2.2. Changes in the Last Decade**

During the last decade, the software development process had many changes; some of them had a direct impact on the way the software development must be controlled:

- Now, modular approach is utilized in almost every development environment irrespective of whether traditional programming languages or fourth generation languages are being used.
- Software systems have become critical in most organizations that extensive and formal test procedures are introduced prior to the acceptance of in-house developed software systems. Different IT subgroups exist for this reason, such as development, acceptance, and production.
- Current systems span multiple hardware and software platforms, each with their own applications. Together, these applications form an integrated system, requiring significant additional testing to ensure correct functioning across a heterogeneous environment.
- The size and complexity of software systems have grown in-line with phenomenal increases in hardware performance.

Due to these changes, SURE has been extended to support the software development process by following this scenario:

- A task identifies and describes the functionality that must be changed in a software system. This task might cause changes in multiple distinct source modules. These changes are transferred as a group to the acceptance environment.
- The acceptance team receives the task description and performs an integrated test of the revised functionality. After approval, the acceptance team transfers the changes to the production environment.

### 2.3. Control of Development Life Cycle

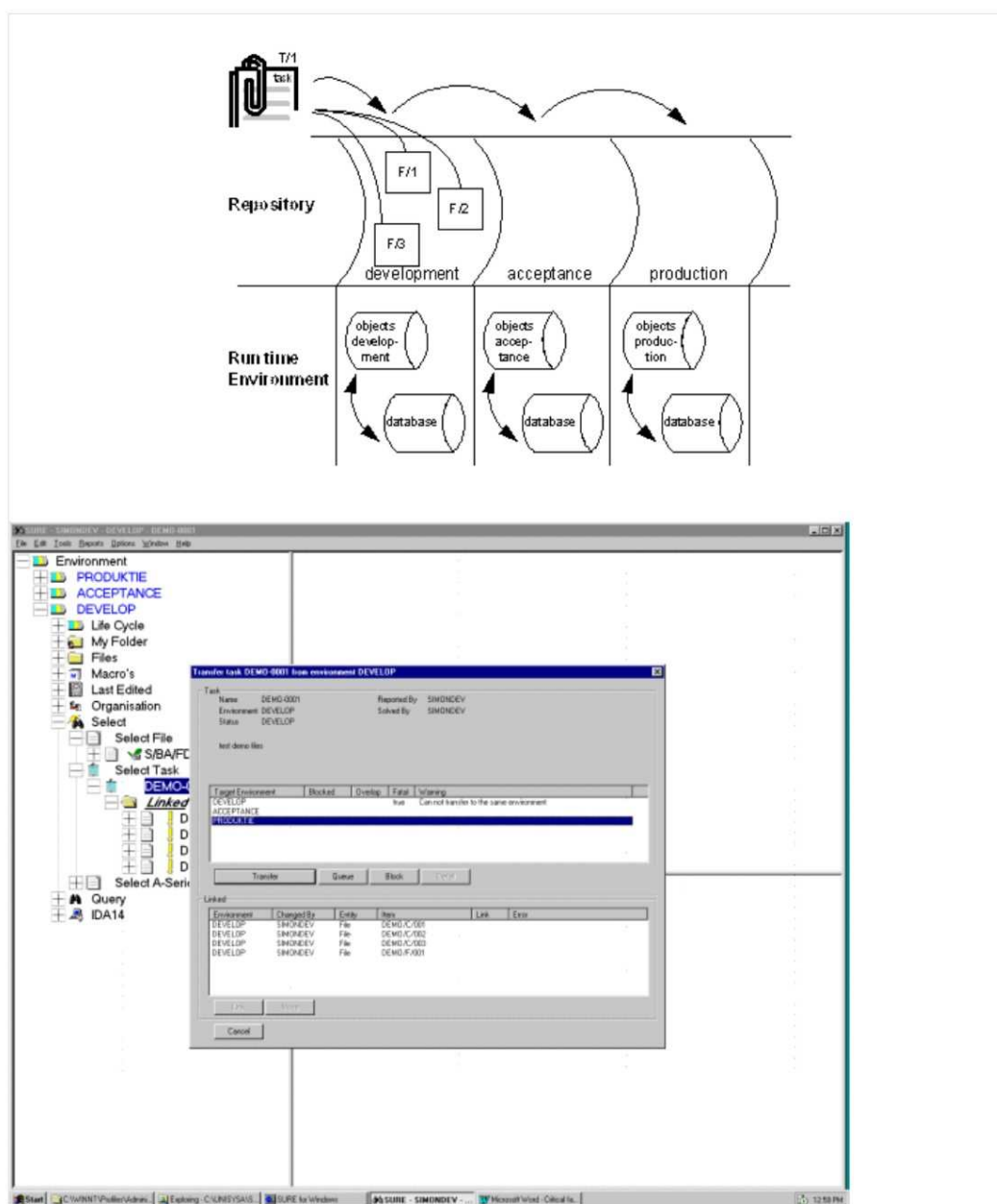
You can perform the following functions using SURE:

- Control the changes to the source module.
- Provide unlimited roll-back.
- Restore a source module to any previous state.

However, this does not provide sufficient functionality to control software development methodology followed today. As a result, the following two mechanisms must be added to the scenario:

- Integrating task administration to group changes must be made mandatory. This allows “non-technical personnel” to release tasks into production.
- Provisioning multiple and distinct environments; each environment containing a complete software system that matches the status of that environment. Thus, quality assurance of production software can be made independent of development process.

These two mechanisms are the key features of the latest SURE system.



005152

This picture shows SURE with three configured environments—Development, Acceptance, and Production. Each of these environments contains source modules in a “repository slot” with the resulting objects and matching database(s) in the corresponding run-time environment.

The task T/1 describes the required functionality and results in changes to the files F/1, F/2, and F/3 as follows:

- Transferring task T/1 from Development to Acceptance results in the transfer of F/1, F/2, and F/3 to the Acceptance environment.  
SURE will recompile these files resulting in a change to the run-time environment for Acceptance.
- Transferring task T/1 from Acceptance to Production, results in the transfer of F/1, F/2 and F/3 to the Production environment.  
SURE will recompile these files resulting in a change to the run-time environment for Production.

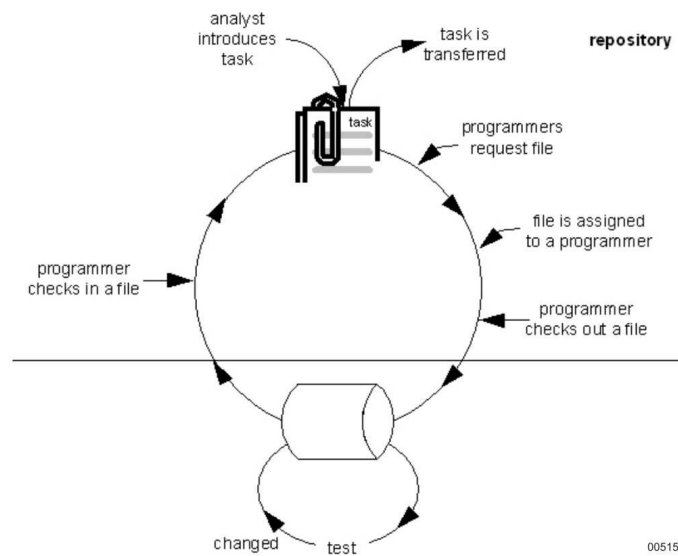
This example shows only the basic structures of the SURE environment. In the following sections, SURE capabilities are described in more detail.

## Section 3

# Benefits of Using SURE

### 3.1. Enforce Development Standards

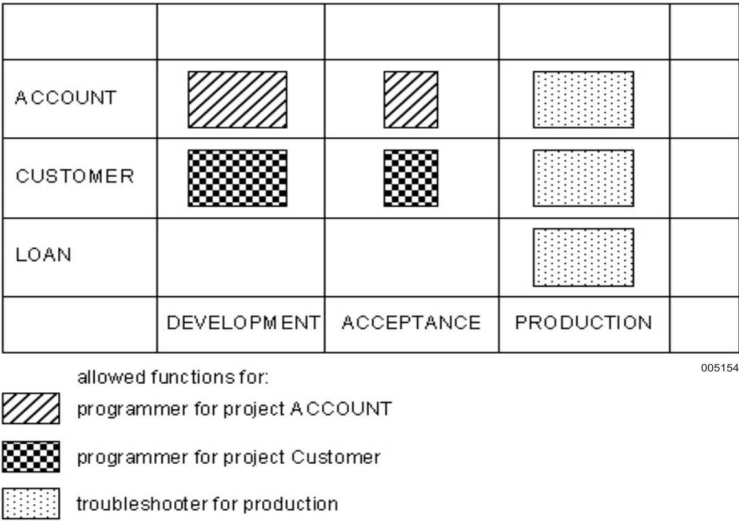
Development procedures and standards are usually documented in most development environments. However, the application of these procedures and standards often depends on the knowledge and willingness of individual programmers. With SURE, you can enforce these procedures. For example, check-in and check-out procedures, enforcing naming standards to ensure standard file naming conventions within any system.



**Benefits of Using SURE**

---

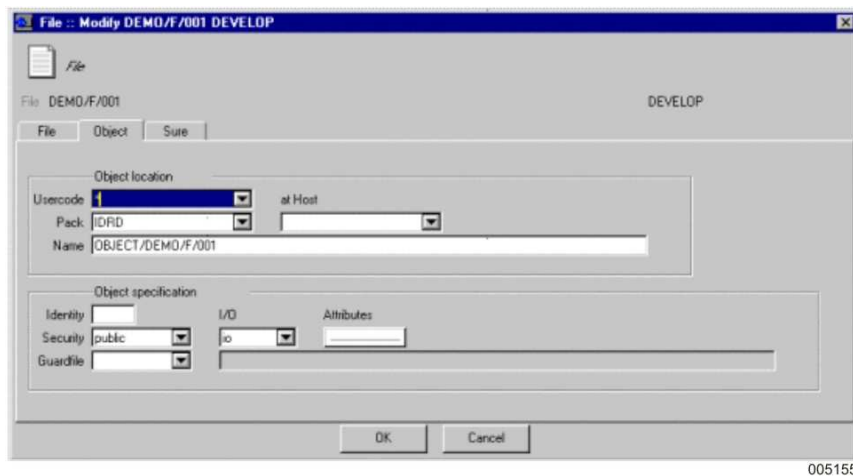
Within SURE, every file is related to a project and is identified by an arbitrary name that often reflects a sub-system. Security within SURE is applied at the levels of employee functions, environments, and projects. With this flexible security mechanism, it is possible to define, implement, and enforce security procedures suited to the requirements of any specific development process or organization.



The size of the square defines the amounts of functions allowed.

## 3.2. Independence from Individuals

Software development projects often depend on the skills and creativity of individuals. However, this creativity must be concentrated on the development of the software, and not on the control of the software. SURE provides control measures to protect any project from dependency on individuals as far as control is concerned. An example of such a measure is the unique registration in the repository of object-locations, security, non-standard object names or object characteristics, for example, privileged programs. This also applies to other features such as program binding, or any other special action upon which the SURE compiles software acts. In general, it is always possible to reproduce the complete set of objects in the run environment from SURE.



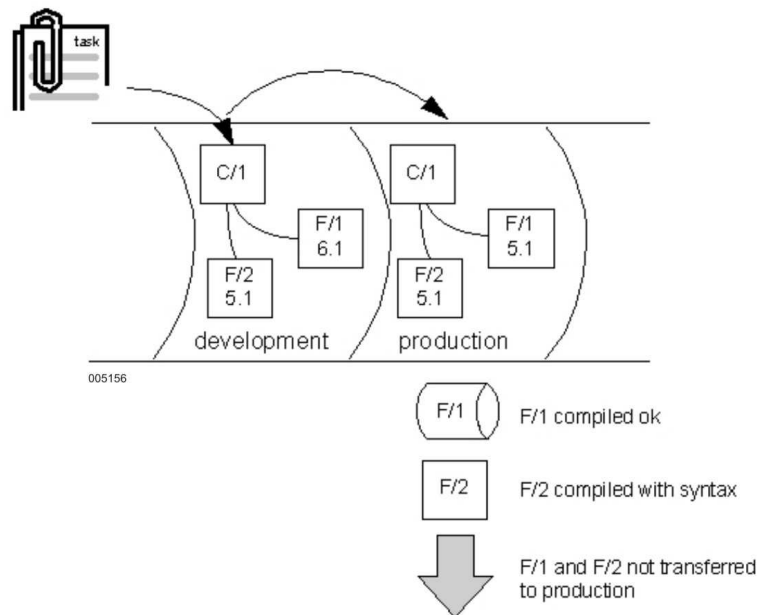
005155

## 3.3. Automated Control of Environments

As stated earlier, the software systems today have a modular character, implicitly making many software modules dependent on each other. For example, checking in a copy file requires recompilation of all modules using that copy file. Only in this way we are able to maintain the integrity of the application system.

The repository used by SURE has a “self learning” mechanism about dependencies between source modules. Each time a file is checked in to the SURE environment, based on the type of the file, the file is parsed by an internal syntax scanner and SURE automatically stores all relevant dependencies in its repository.

The information extracted from source modules is used to maintain the integrity of the application system. So, if you check in a copy file, all modules using the copy file are compiled. The status in one environment (for example, Development) might be different to that in another environment (for example, Acceptance).



This integrity mechanism in SURE can handle situations that are even more complicated. Suppose that C/1 is a copy file defining a record layout used by the programs F/1 and F/2. The versions of file F/1 are different in Development and Production, since one of them might have been changed by another task. After checking in the file C/1, the system will compile F/1 and F/2 and they might compile correctly.

Transferring the task connected to C/1 will cause a compilation of the files F/1 and F/2 in the production environment. The file F/1 in production might result in a syntax error while, at the same time, the file F/2 might compile correctly. At this point, the object file F/2 will not be deployed to the run environment as the dependency between F/2 and F/1 through the copy file C/1. SURE recognizes this situation and handles it accordingly.

### 3.4. Workbench for MCP Development

The source files for MCP development on ClearPath Libra Servers were traditionally edited using editors such as CANDE or the editor utility, SYSTEM/EDITOR.

Editing the files using these mechanisms offer a relatively primitive user interface, compared to PC editors, along with the advantages of using development tools running on the same platform for which development is being done.

Since choosing the "best editor" is a personal preference, SURE provides the capability to edit and compile on a PC with any PC based editor. This functionality allows MCP source files to be edited on a PC, and then compiled on a ClearPath server through a PC "pseudo compiler." The error files are merged into the symbolic code after compilation. Optimal performance is also obtained by sending "merging files" only to the ClearPath server, reducing transmission times.





The original SURE product was designed to support software development on Unisys CP/NX Series systems only. As a result, the current compilation and transfer mechanisms are only available for the ClearPath Libra platform, as it is the current generation of MCP mainframe. However, all other SURE functionality is also applicable to PC based projects.

Using SURE for PC applications allows similar task integration and environment control as for ClearPath Enterprise Server applications. Except for lifecycle support, SURE provides build and distribution support for the Windows platform.

The build support allows automated creation of executables on the Windows platform because of the changes made to the source files. Optionally, the resulting executables are loaded in the repository.



## **Benefits of Using SURE**

---

The distribution support allows creation of a directory of files based on system, sub-system, or destination location. This directory of files can be used by a distribution tool such as, Microsoft SMS, for the actual file distribution in the network.

The SURE syntax scanner extracts the dependencies in the MAKE file located in the path "C:\PR\F1.MAK." Checking out this MAKE file might copy the dependent files, and then development and editing of files takes place on the PC. The files are checked in to the SURE environment only after completing the task. Transferring the task also transfers the files to the next environment.

## Section 4

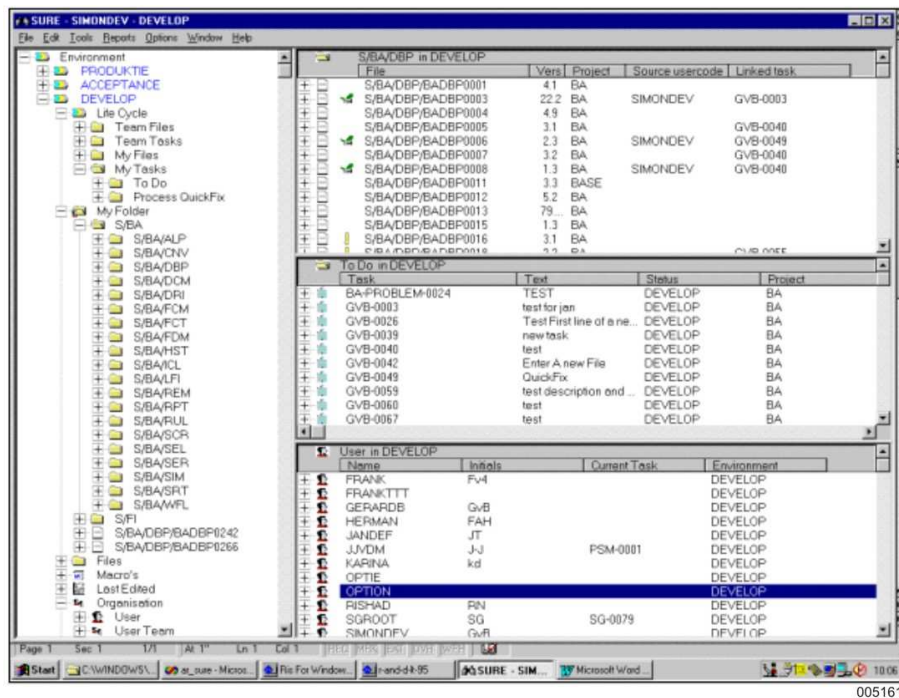
# Modes of Operation with SURE

### 4.1. Development on ClearPath Libra Servers

SURE is used for check-in, check-out, and inquiry purposes. There are no restrictions or implications for the development methods used on ClearPath Libra Servers.

### 4.2. SURE for Windows GUI Interface

SURE for Windows is a client-server application. A Windows application is connected to a ClearPath server application through Winsock, TCP/IP, or InfoConnect. The Windows interface is similar to the Windows Explorer; multiple windows on the right side of the interface might contain different information. The following example shows a left Explorer style window and three windows on the right side with files, tasks, and users. You can use this interface to control sources of PC applications and ClearPath server applications.





## Section 5

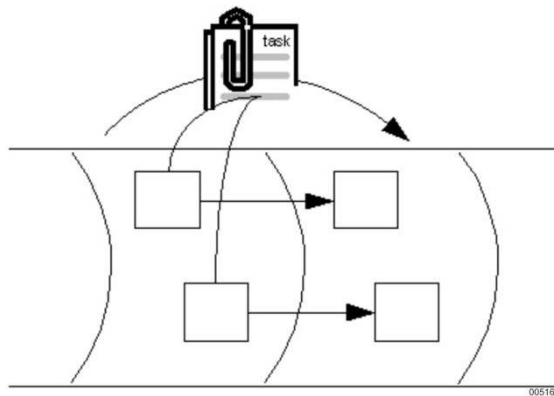
# Versioning within Life Cycle Support

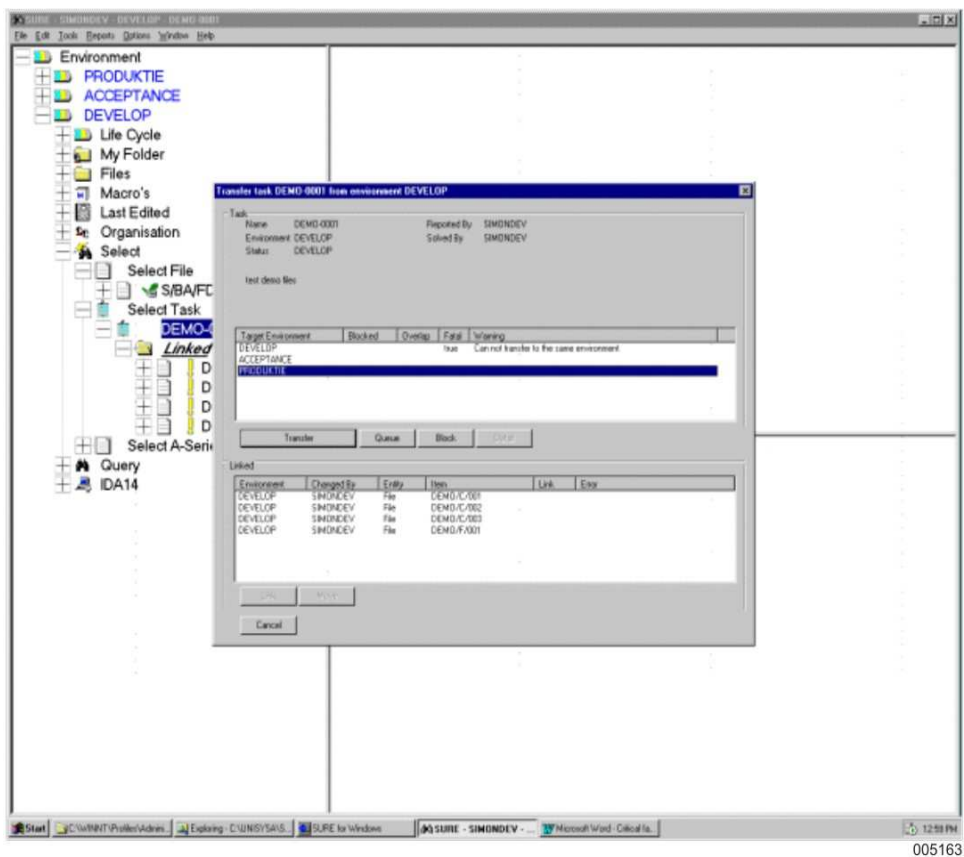
### 5.1. Using Tasks to Identify Changes

A unique concept in SURE is to use a task as a unit of work and to integrate task administration with the control of software development. Entering a task in the system requires a description and results in identification of a unique task.

A task identifies a project and the associated developers. Thus, defining a task implicitly defines the workload for the various developers working on that project.

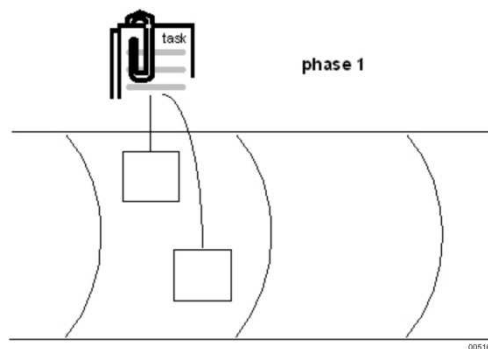
If a developer checks out a source file, SURE automatically creates a link between the task and the source file. This enables history to be maintained over time of multiple developers checking out and working on multiple source modules. The changes are transferred to another environment, by transferring the task to the new environment, for example, Acceptance. As a result, the task is the actual unit of work and is the sole mechanism to control changes.

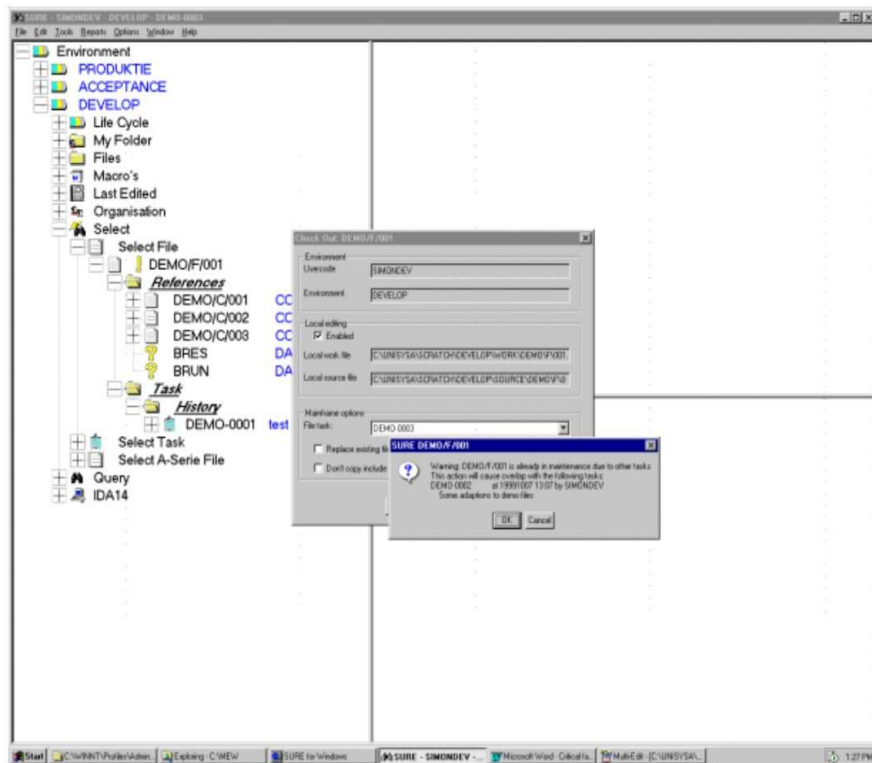
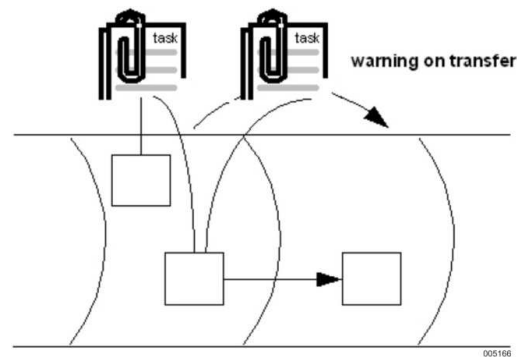
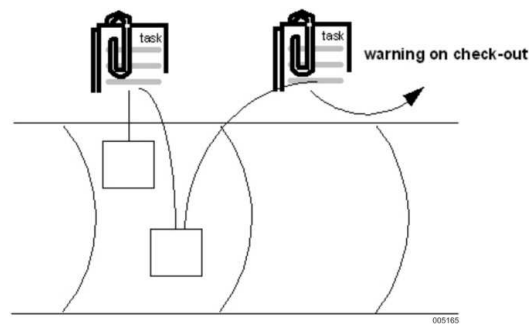




## 5.2. What is an Overlapping Task?

The relationship between task developers and task source files is a “many-to-many” relationship. So, it is possible to modify a source file for multiple tasks. In other words, such a source file contains changes for multiple tasks. If one of these tasks is transferred to another environment, any dependent source file that might also contain changes for other tasks is also transferred to that environment. The SURE system allows transfer of such “overlapping tasks” but warns a programmer when the programmer checks out a dependent source file, and also warns anyone transferring the task.



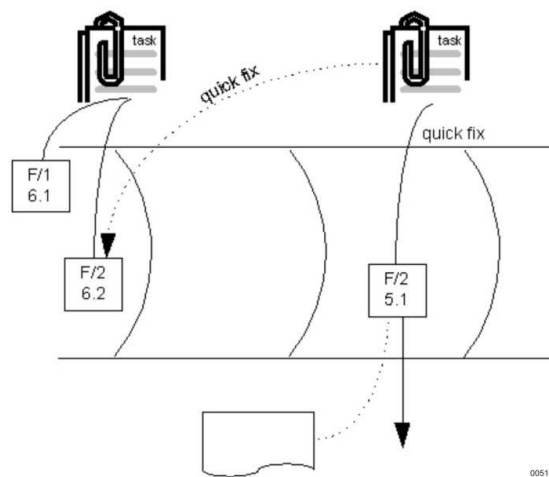


### 5.3. What is a Quick Fix?

Overlapping tasks result in a program that contains changes for different tasks and it is not possible to record the changes for any one task separately into production. Alternatively, there might be situations where this overlap is unacceptable, for example, solving fatal production errors. In such cases, it must be possible to get rid of the fatal error without taking any unwanted new features into production. SURE notifies the developer when such changes have occurred.

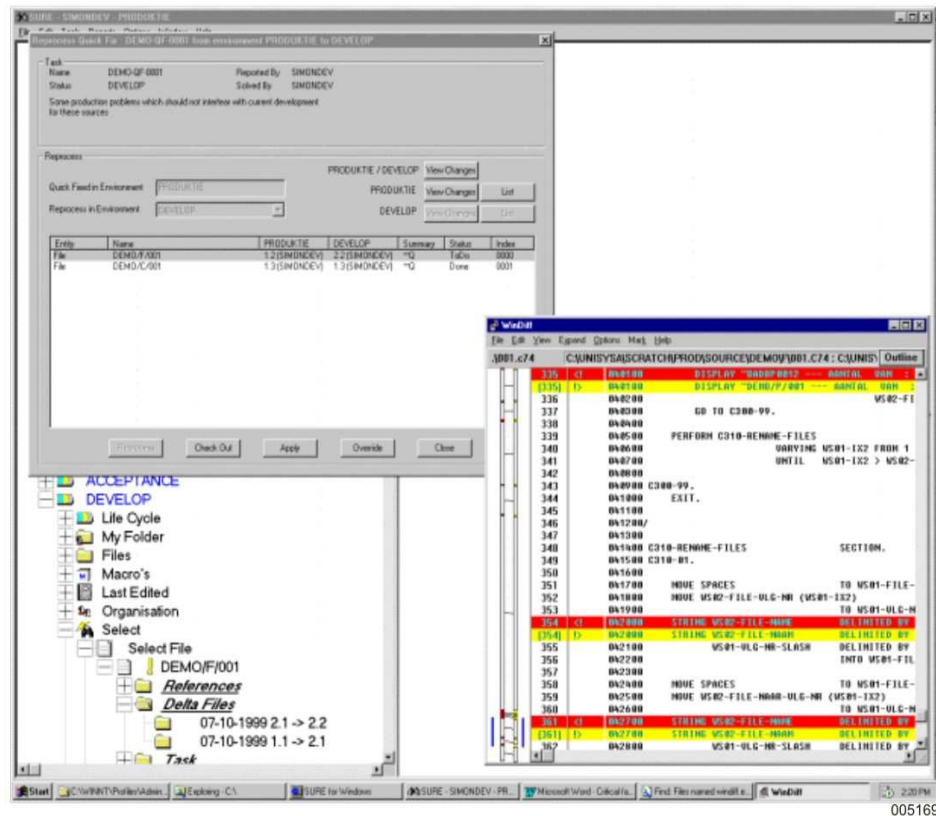
By default, developers make software changes in the lowest hierarchical environment and then transfer these changes to higher environments, for example, transferring the changes from development to acceptance and acceptance to production. In the case of a “quick fix” a different approach is used, the developer makes the changes in a higher environment. SURE reports to the lower environments that these changes have been made.

A “quick fix” is therefore the ability to make changes to source modules in a higher environment without disturbing the development process for production environments or releases. The SURE system provides the appropriate implementation and feedback mechanisms to support quick fixes. SURE also provides a mechanism to merge patches in a lower environment.



The file F/2 was modified in development. A production error required modification of the file F/2. However, during the check-out procedure, the system indicated overlapping tasks and offered a quick fix option. This quick fix allowed the file F/2 to be changed in production, so that the actual production version was fixed without the new functionality. Thereafter, the file F/2 shows the presence of a quick fix and it is also reported in a batch listing. The changes made in production for the file F/2 must be incorporated into the file F/2 in development. When the change is incorporated in the file F/2 in development, the quick fix notification disappears.



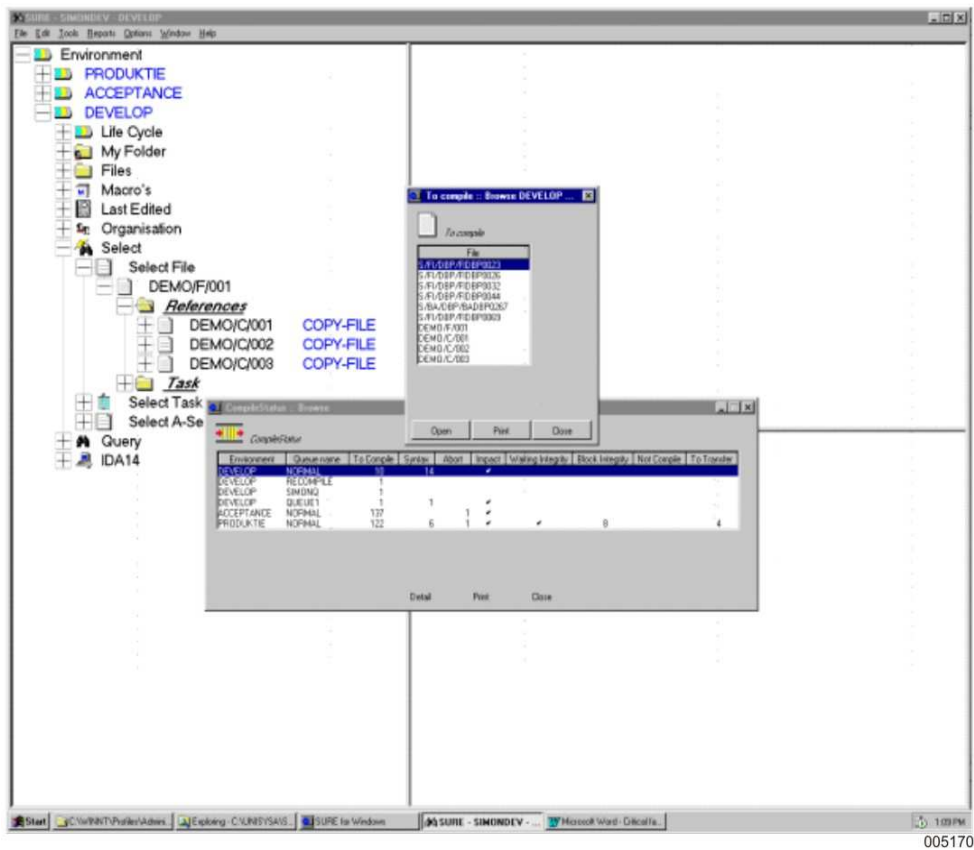


## 5.4. How is Integrity Maintained?

A highly sophisticated mechanism is required to maintain the software integrity of modular systems within multiple environments. SURE handles this requirement through its syntax scanning facility that creates dependencies between programs based on “technical” links, such as, copy files. The integrated task administration also creates dependencies between programs based on “functional” links. For both types of dependency, SURE can ensure that all programs belonging to a task or to a technical unit are transferred to production as one consistent group.

Even though all programs are transferred to production as one group, syntax errors might occur in the production environment. This might be because of different description files, or because of other environmental software. In other words, if a program contains syntax errors, the technical or functional group of files to which this program belongs is not complete. The SURE transfer mechanism makes it possible to distribute files automatically and to ensure software integrity in the production environment. This is achieved by transferring the group of files only if they are compiled without syntax errors.

## Versioning within Life Cycle Support



## Section 6

# **SURE Development Support**

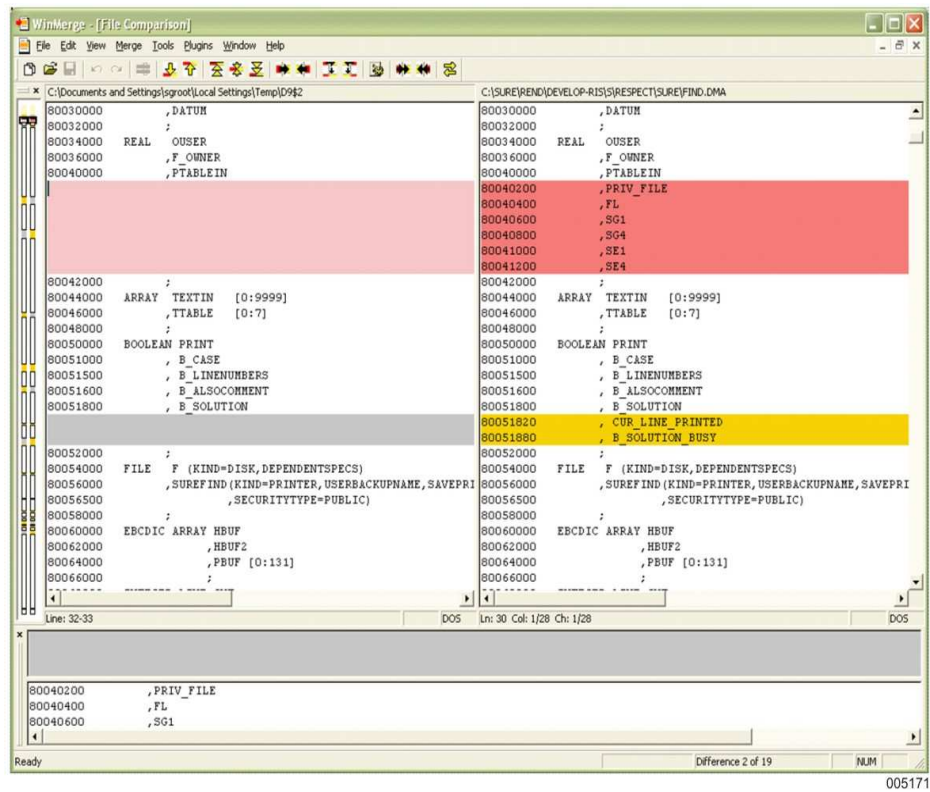
### **6.1. Using the Differences Between Successive File Versions**

Formal procedures for software changes and a trace facility to find out “who did what, when, and why” are often mandatory from the viewpoint of an Electronic Data Processing (EDP) auditor.

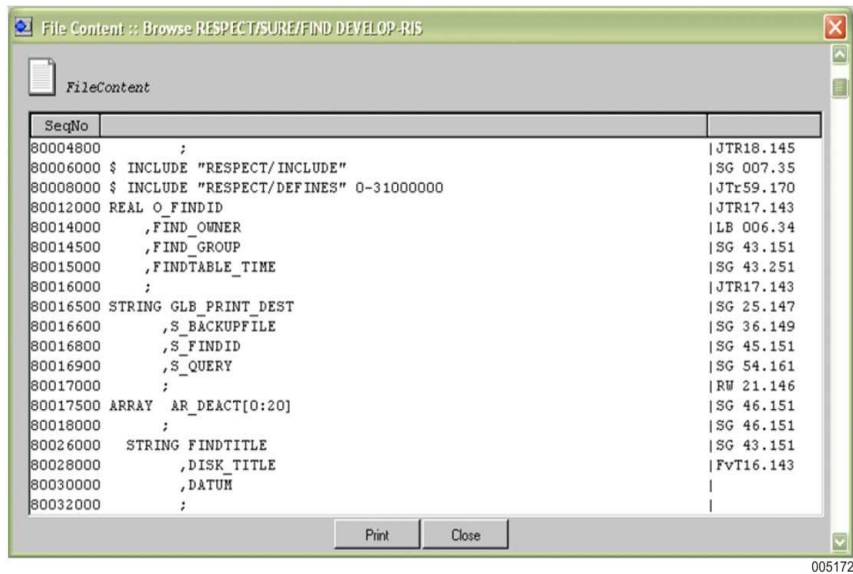
For purposes of software debugging, this functionality is also beneficial. “Fixing errors” becomes easier, when additional information is available about a problem and about how various program lines have attained their current state, through knowledge of when and by whom changes have been made.

If a developer checks in a source file, SURE creates a “delta” file that indicates the differences between successive versions. For each environment, a source file contains its symbolic code and multiple delta files. The SURE system uses these delta files for different purposes, for example, to restore a file to a certain version, to create a patch file for ClearPath Enterprise Servers, or to scan a sequence range searching for modifications.

This example shows a shareware product called “WinMerge” showing the differences between two files. The two files are constructed using the delta files.



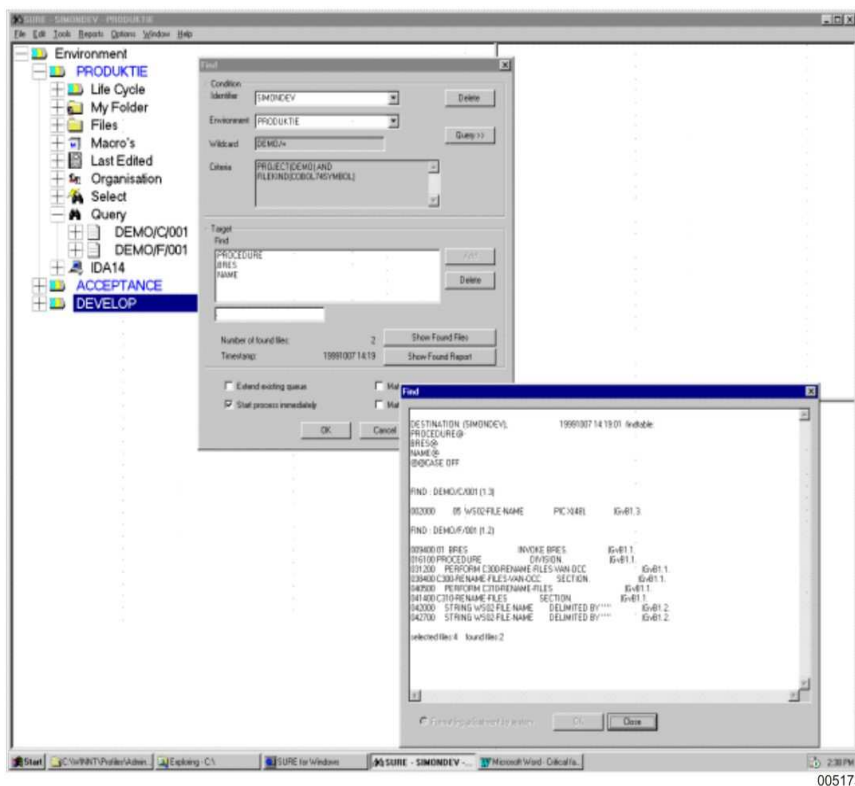
If a developer checks in a source file, SURE enters the mark-id field with version and programmer identification details. Therefore, every line will indicate in which version and by whom it was last changed.



## 6.2. (SURE Development Support) Query Facility

The underlying structure of the SURE software is a repository that holds all source files and related information. The source file information is stored by the following procedures:

- The check-in and check-out procedures and by the compilation process.
- The SURE syntax scanner, for example, information on copy files, declared database, and invoked data sets.
- The life cycle support mechanism in SURE by extracting the information from the actual source files. The information is stored explicitly in the repository and is available for use through a dynamic query language. Examples of these query languages are:
  - Show files with data-set (account) and changed (since 19950101).
  - Show files maintained-by (G\_Anderson).
  - Show files with (copy file (C/1) or copy file (C/2)) and object-pack (acc).



## 6.3. (SURE Development Support) Information

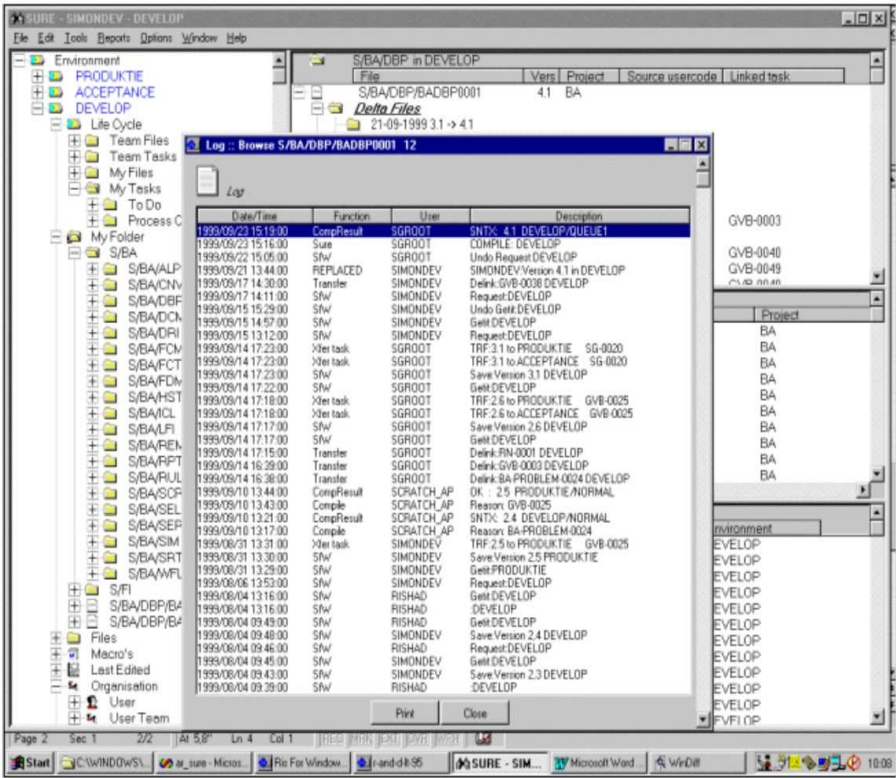
Programmer productivity might be improved by providing relevant and easy accessible information. In SURE, such information includes run-time statistics and an event log.

6.4. Run-Time Statistics

The SURE batch facility contains a program that reads the system SUMLOG file to extract information that is later loaded into the repository. This results in the storage of many run-time statistics for every program or job that is present in the system log. This information can then be shown on the user interface.

6.5. Event Log

All SURE events impacting a source file are kept in a log file and can be shown on the end user interface. The event log contains all the check-in, check-out, and transfer actions that were performed on a file. From the task administration facility, it obtains and shows all the connected files.



6.6. Controlled and Automated Management of Environments

For each environment, a batch run controls the compilation and transfer of objects to the defined destinations. This batch run is a fully automated mechanism that does not need manual intervention. No jobs must be modified if new symbolic code is added to the system.

## Section 7

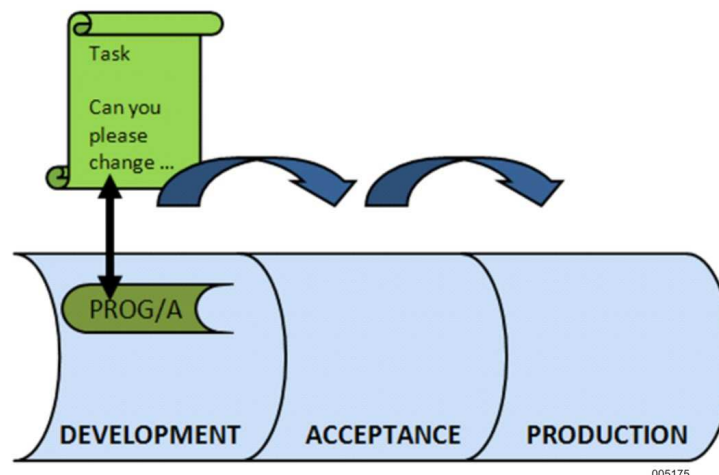
# Applicability of SURE

### 7.1. Development of Internal Software

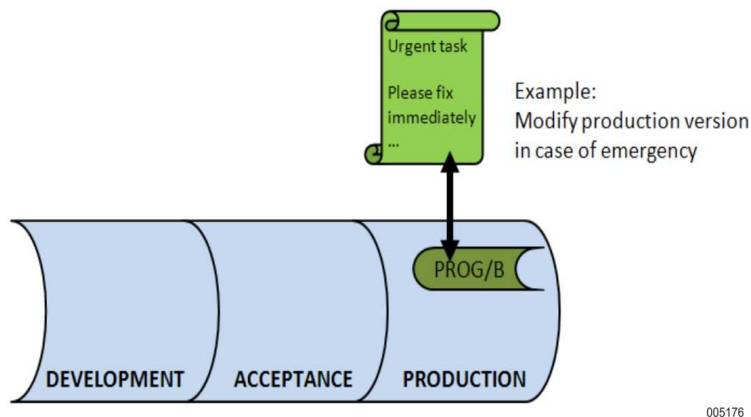
Many organizations develop their own information systems. The resulting software is unique and is used only by that organization. The basic flow of the development process within such an organization does not differ very much from that of a software manufacturer. In both cases, changes are made by developers and through an acceptance procedure the changes are taken into production.

There is, however, a difference in the support of old software releases. Although a release identification mechanism may be used for “in house” developed applications, old releases will not be supported and maintained. This is different from packages supplied by software manufacturers, where old releases are often supported for a certain period.

In SURE, the configuration for the development of internally written software consists of a sequence of environments, for example, Development, Acceptance, and Production. By default, sources are modified in the development environment. Tasks are required to modify source modules, and the changed sources are automatically linked to a task.



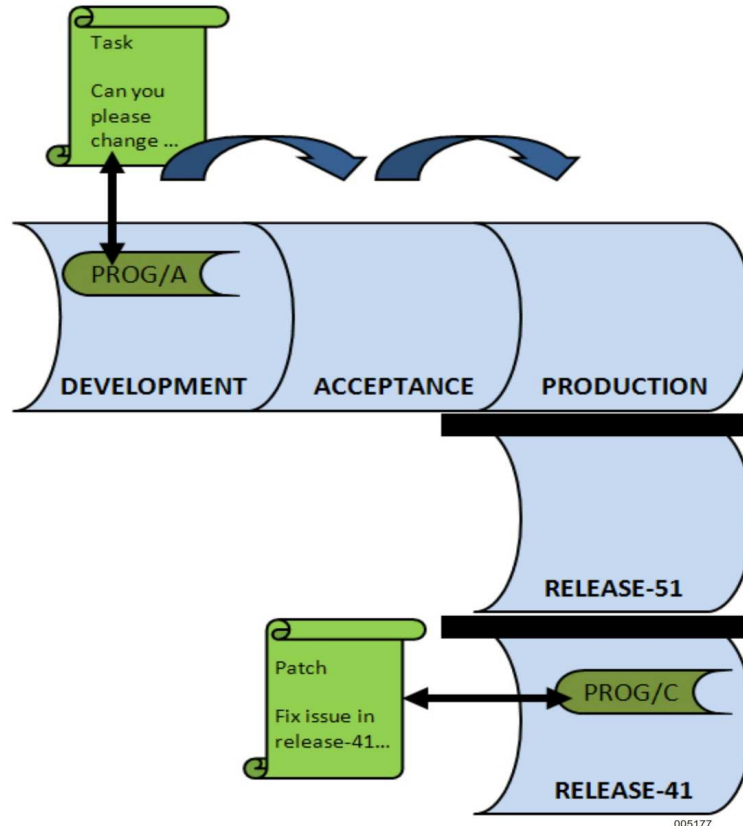
If you use quick fixes method, the system permits the modification of a production source. The task is categorized according to a user defined tasktype that indicates the ability to make a quick fix and the environment in which it can be executed.



## 7.2. Development of Packages

Commercial developers of software packages require formal release mechanisms to deliver the package to their customers. New releases are delivered often as a complete set of software for the new version. Support of the package often requires delivery of patches or updates to those customers using a particular release.

Since packages are sold to many different customers, different releases of a package might be running at various sites. SURE is designed to deal with the specific requirements for the delivery of packaged software.





SURE handles the development and delivery of packages through two major mechanisms:

- SURE delivery mechanism: At any time, a new environment can be created to prepare for the shipment of a new release.
- Patch file mechanism: Support of releases in the field is achieved with the quick fix mechanism.

The SURE delivery mechanism and the patch file mechanism provide new releases to customers and support for existing releases.



## Section 8

# Installation

Installation of the SURE software consists of two steps:

- The installation of the SURE PC software.
- The installation of the mainframe software.

### 8.1. Installation on the PC

#### 8.1.1. (Installation) License Key

The SURE PC software is protected through a license key mechanism. One copy of the program through the same workstation is initially allowed. With this first copy, the license key might be entered through the SURE explorer Options/License.

This section contains valuable information about the structure of the SURE client software. We strongly advise this chapter to be read by the technical support staff. If you have questions about this chapter, then contact Infra Design.

The actual installation procedure is described in the "PC install procedure."

#### 8.1.2. (Installation) PC Application Structure

The main components in a SURE PC application are:

- <installation directory>\Ris\Bin\AW\_OBJ.EXE  
The executable file for the client software. The program manager executes this program with the configuration file as parameter.
- <installation directory>\AW\_OBJ.INI  
A configuration file or INI file that contains redirections for directories. This configuration file is passed as a parameter through the program manager definition. If no parameter is provided in the program definition, a file called AW\_OBJ.INI is used from the Windows directory. A secondary INI file might be located in the same directory as where the executable AW\_OBJ.EXE is present. This global AW\_OBJ.INI file contains attributes that override the attributes in the private INI file.

- <installation directory>\AW\_OBJ.CFG

A second configuration file is used with the same name as the AW\_OBJ.INI file, but with CFG as extension. This file contains objects in environments that should be added to the toolbar. In the SURE installation this results in showing the logon screen after starting up. If this file is missing, no logon screen is shown.

- <installation directory>\Ris\Tool\\*.\*

The configuration module of SURE on the PC is a RisForWindows system using a tool repository. This tool repository contains definitions for configuration entities such as DLL, Repository, Environment, Security, and User. This configuration tool repository is located through the TOOL folder in the configuration file.

- <installation directory>\Ris\Data\\*.\*

The configuration data that describes the installed modules such as DLL's, Repositories, Environments, Securities, and Users are located in the configuration data repository. The configuration data repository contains static data, which is applicable to all installations. The configuration file contains tailored data each installation such as directory redirections.

- <installation directory>\Ris\Bin\\*.\*

The AW\_OBJ.EXE executable uses a restricted number of direct linked libraries such as cd\_res.dll, cd\_ris.dll, cw\_formi.dll, and cd\_prof.dll. These direct linked DLL's are located by the Windows system through the defined default path in the program definition. For this reason, the default path should be the path where the SURE executables are installed; by default the path in which the AW\_OBJ.EXE files are located. The installation sets up a program definition with the correct path supplied. If this path name is altered afterwards, the default Windows search mechanism is invoked using the PATH environment variable and the Windows or Windows\System directory.

- System DLL's

The SURE software requires some system-defined libraries and a WINSOCK interface. By default, these libraries are installed by the Windows software at installation or if the system is upgraded.

- Included applications

A SURE application consists of a tool repository with additionally application extension DLL's, a data repository and bitmap files. The included applications are located through the DIRECTORY chapter in the configuration file.

[DIRECTORY]

DLL           =<directory>

TOOL          =<directory>

DATA          =<directory>

BITMAP       =<directory>

CRW           =<directory>

For DLL files and bitmap files a second search level is invoked for the default directory, in other words if the file is not present in the directory defined by the repository redirection, the file is located in the directory defined by the directory redirection.

```
[<repository>]
DLL             =<directory>
TOOL            =<directory>
DATA            =<directory>
BITMAP          =<directory>
CRW             =<directory>
```

### 8.1.3. (Installation) PC Install Procedure

The SURE PC software is delivered through one CD-ROM:

SURE\_xxx (xxx is the release number of the software)

Prepare a PC with the following specifications:

Processor	80486 or better
Memory	at least 8 Mb, preferably 16 Mb
Hard disk	at least 6Mb free space
Software	Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP

A TCP/IP connection to your mainframe

Put the CD-ROM in the CD-drive. The installation program starts automatically.

If the installation program does not start automatically then:

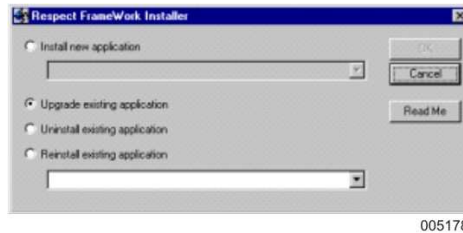
E:\SURE\_xxx (where E refers to the CD-ROM drive).

Double-click the INSTALL.EXE to start the installation procedure.

### 8.1.4. (Installation) PC Install Program

#### 8.1.4.1. The First Installation Dialog

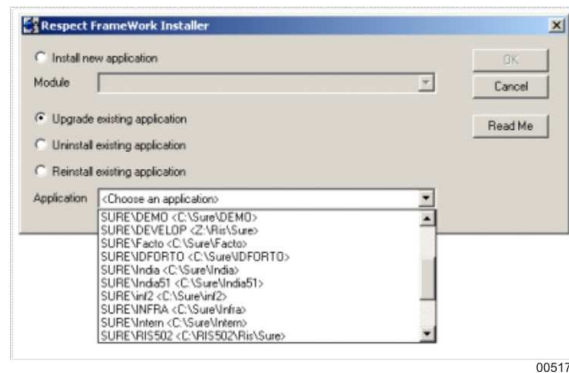
Define the installation action: new, upgrade, un-install, and re-install.



- A click on this button opens the README file. This file contains bootstrap information for the installation. Read this file before you proceed.
- Install New Application

If SURE must be installed in a new path, then this option is used. This installs all components of SURE including tool repositories, data repositories, executables, a configuration file, and program manager items. A new installation in the same directory as an existing installation behaves like an "Upgrade existing installation".

On the client, the installed paths are maintained in an INI file in the Windows directory. It is possible to install SURE multiple times on your PC, each time in a different path. The second drop-down box shows the paths on your PC where SURE is already installed. You can upgrade, un-install, or re-install the SURE application.



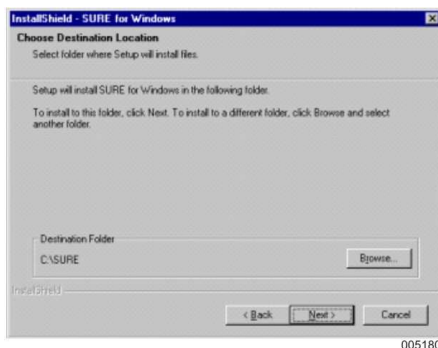
- Upgrade existing application  
You can upgrade a previously installed installation, which is named in the drop-down box.
- Un-install existing application  
You can un-install a previously installed installation that is named in the drop-down box. The uninstall action will remove all files that were previously installed in this path and not updated since then, and it removes the items in the registry.

- Re-install existing application

You can reinstall a previously installed installation that is named in the drop-down box. The application is uninstalled and then reinstalled. You can use this function if too many files are present in a given installation.

### 8.1.4.2. The Second Installation Dialog

#### Path



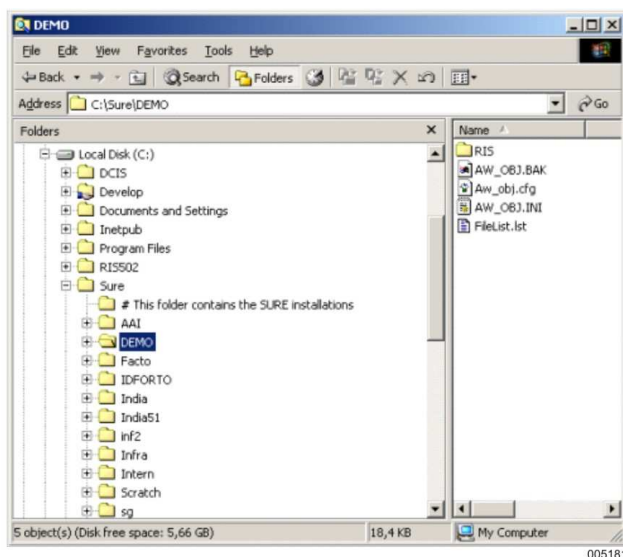
The default path is C:\SURE but you can choose any other path you like.

We recommend to use sub-directories so that it is easy to install the SURE client software a second time, but in another sub-directory.

#### Example

Consider the case that you have (amongst others) two SURE repositories on the mainframe: one for the AAI department and one for the DEMO department.

In this case, we recommend to install the SURE client software in two sub-directories: C:\Sure\AAI, which routes to the SURE-AAI repository, and C:\Sure\DEMO, which routes to the SURE-DEMO repository.

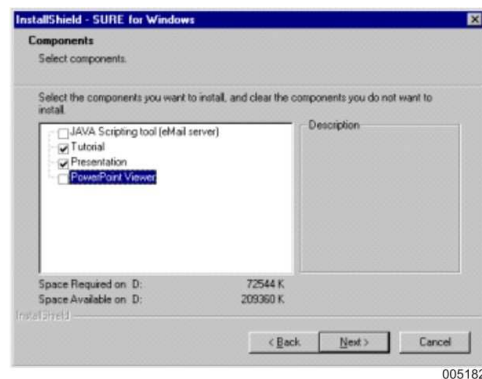


This example shows that SURE is installed multiple times using the sub-directories:

- C:\Sure\AAI
- C:\Sure\DEMO
- C:\Sure\Facto

### 8.1.4.3. The Third Installation Dialog

#### Installation components



- Java scripting tool (for email server)

The installation of this component is only applicable if you want to define this PC as SURE email server, and the mail program is NOT Outlook Express and a scripting tool was NOT yet installed on the PC. Only one PC has to be configured as SURE email server.

This option installs the Java Scripting Engine required for executing a Java Script (.js file). The SURE client software only uses this Java scripting facility if you install the email server installation option. The email server installation option configures a PC so that it can send the email request from SURE to the correct email address. If you are using Outlook Express, the mail is sent directly using the MAPI interface. If you are using Outlook or another mail client, the mail is send using a Java script called <Directory>\Ris\Script\Mail.js. This Java script uses the Outlook component model for sending this mail.

- Tutorial

The SURE tutorials are also installed.

The tutorials contain a systematic quick introduction to SURE. The tutorial leads you through the basic functions of SURE, and when you finish the tutorial, you have a better understanding of the features of SURE.

We recommend setting this option to install the SURE tutorials.



- Presentation

The SURE presentations are also installed.

Various PowerPoint presentations about SURE and SURE functions are available. These PowerPoint presentations can be accessed from the SURE explorer and can be used as a help file.

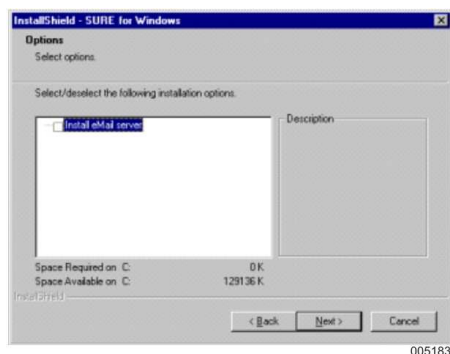
We recommend setting this option to install the SURE presentations.

- PowerPoint Viewer

The SURE presentations (as mentioned above) are created with PowerPoint. Set this option to install a PowerPoint viewer if PowerPoint itself is not available on your PC.

#### 8.1.4.4. The Fourth Installation Dialog

##### Options



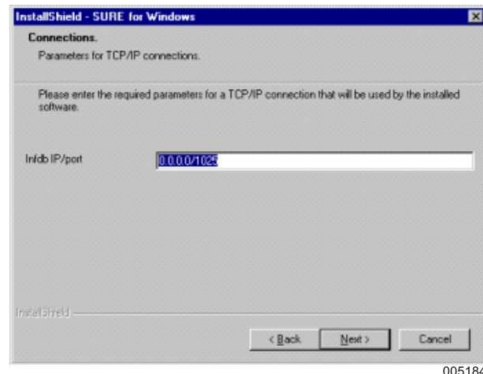
- Email Server

Set this option if you want to define this PC as SURE email server. Only one PC has to be configured as SURE email server.

SURE is integrated with an email system for outgoing mails from the SURE system. This facility requires that a PC is defined as SURE email server. This SURE-email-server is a standard email client in the mail system (that is outlook express). Emails are routed by SURE from the mainframe to the email server (using the IP number of that station and a self chosen port number >1024), and the server forwards them to the correct email address.

### 8.1.4.5. The Fifth Installation Dialog

#### Connections



005184

The IP address of the host and the TCP/IP port number for the communications has to be defined in this dialog in the format "nnn.nnn.nnn.nnn/pppp".

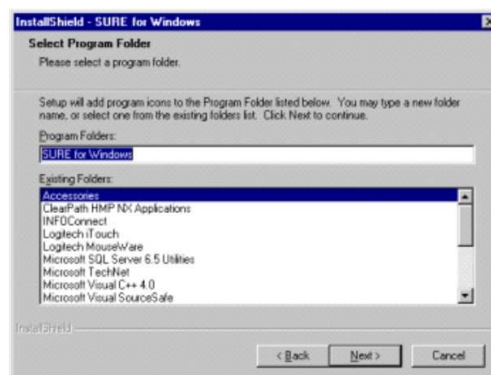
If you are using a LMHOSTS file then you can enter the name of the host followed by the TCP/IP port number in the format "<hostname>/pppp".

The default port number is 1025. Port numbers less than 1025 are not recommended because they are not supported anymore under MCP release 47.1.

This port number is also used during the mainframe installation procedure. Make sure that the same port number is used for the mainframe-installation and the PC-installation.

### 8.1.4.6. The Sixth Installation Dialog

#### Entry in the programs list



005185

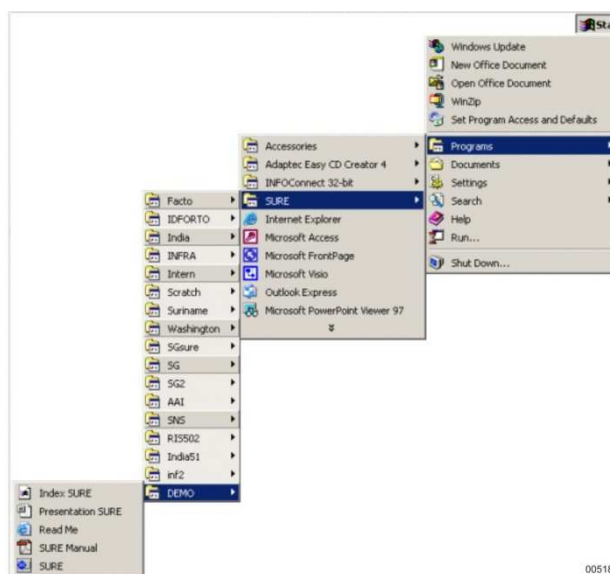
- This creates an entry in the programs list in the start menu. The default is “SURE for Windows”.

We recommend using sub-directories in the program folder so that it is easy to install the SURE client software a second time.

### Example

Consider the case that you have (amongst others) two SURE repositories on the mainframe, one for the AAI department and one for the DEMO department.

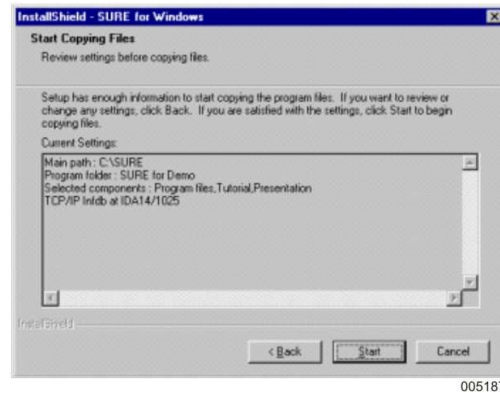
In this case, we recommend to install the SURE client software with two sub-folders: Sure\AAI, which routes to the SURE-AAI repository, and Sure\DEMO, which routes to the SURE-DEMO repository.



This example shows that SURE is installed multiple times using sub-folders:

- Sure\AAI
- Sure\DEMO
- Sure\Facto

After these dialogs, a confirmation screen is presented, with a summary of the installation that will take place. Not all choices presented here were user-enterable in this demonstration. If the data is correct, press <Start>. Pressing <Back> will let you go through the dialogs again.



The set-up program now installs the SURE software. During this installation, it spawns several programs that finish automatically.

The set-up program creates a program group "SURE for Demo," which is added to the programs start menu of your PC. This program group contains the following shortcut to SURE:

```
<directory>\RIS\BIN\AW_OBJ.EXE <directory>\AW_OBJ.INI
```

The installation of the client software is now complete.

### 8.1.5. (Installation) PC Configuration Management

The client software of SURE can be configured in various ways depending on the location of executables and repositories.

The following configurations are possible:

- All the files are located and executed from a single workstation.
- All the executables, tool, and data repositories are located on the shared server disk. The client does not contain a single file.

#### 8.1.5.1. (Installation) PC Standalone Configuration

The standalone configuration installs all files on a single computer. In case of a standalone configuration, all files are installed in a single directory (defined during the installation procedure). The installation procedure creates a program manager entry that uses a configuration file called AW\_OBJ.INI located in the installed directory.

Upgrading a standalone installation does not require any special actions: Just choose "Upgrade existing application" on the install-dialog.

A single PC may contain multiple SURE installations, each installed in a different directory.

### Example Directory StandAlone

```

Directory PATH listing for Volume Infra
Volume Serial Number is 0B59-16EE
C:.\
+---SURE                                     % installation root-directory
|   +--<sub-directory>                     % installation sub-directory
|       AW_OBJ.INI                         % configuration data
|       AW_OBJ.CFG                         % configuration data
|       +--RIS                             % RIS environment
|           +---DATA                       % data repositories
|               |   +---CONFIG             % configuration repository
|               |   +---INFDB              % ClearPath Enterprise Servers
connection repository
|               |   +---DATA               % repository used for logon
purposes
|           +---BIN                        % executables
|           +---HELP                      % helpfiles
|           +---SDK                       % software development kit
|           +---TOOL                      % tool repositories
|           +---BITMAP                    % bitmap files

```

### 8.1.5.2. (Installation) PC Server-Only Configuration

By default, the client configuration creates directories on the client computer that contain the data repositories and the configuration file. The files in this directory are updated by SURE. Creation of a server only configuration requires that the data repositories be set as read-only files on the server location. The only files at the client are partial configuration files with individual preferences and a shortcut.

### Example Directory Server Only

```

Directory PATH listing for Volume Infra
Volume Serial Number is 0B59-16EE
C:.\
+---SURE                                     % installed directory
|   +--<sub-directory>                     % installation sub-directory
|       +--RIS                             % RIS environment
|           +---DATA                       % data repositories
|               |   +---CONFIG             % configuration repository
|               |   +---INFDB              % ClearPath Enterprise Servers
Series connection repository
|               |   +---DATA               % repository used for logon
purposes
|           +---BIN                        % executables
|               |   AW_OBJ.INI             % common configuration data
|           +---HELP                      % helpfiles
|           +---SDK                       % software development kit
|           +---TOOL                      % tool repositories
|           +---BITMAP                    % bitmap files

```

### Example Directory (client in Server-Only configuration)

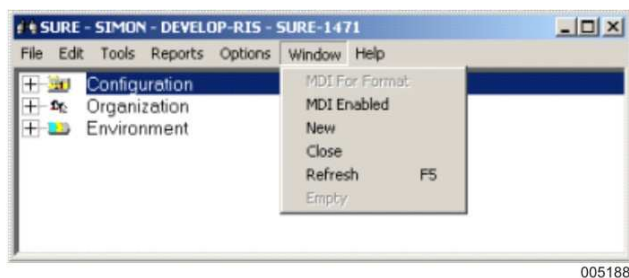
```
Directory PATH listing for Volume Infra
Volume Serial Number is 0B59-16EE
C:.\
+---SURE                                     % installed directory
|   +--<sub-directory>                       % installation sub-directory
|       AW_OBJ.INI                           % configuration data
|       AW_OBJ.CFG                           % configuration data
|       <shortcut to <server>:
|           C:\SURE\<sub-dir>\RIS\BIN\AW_OBJ.EXE  C:\SURE\<sub-
dir>\AW_OBJ.INI
```

Upgrading a server-only installation requires some special actions. Read the following text to understand the server-installation process. At the end the special upgrade actions are described.

Setting up a ServerOnly configuration requires the following steps:

### The Server

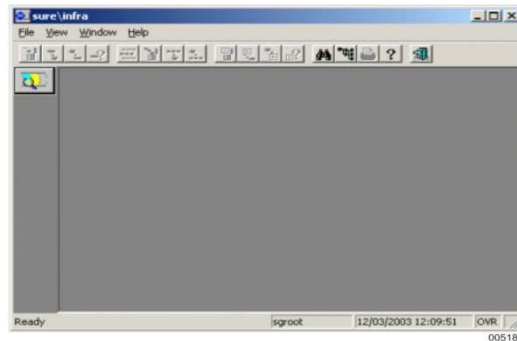
1. Choose a path on the server (called <ServerPath>) where SURE is going to be installed, and create a network connection to that path. The clients will use this connection to the server path. A good way to do this is with UNC-names, because that makes the installation independent of all kinds of drive-names on the clients.
2. Install SURE in the following server path:  
Run <cdrom>:\<directory>\INSTALL.EXE and choose the <ServerPath>
3. Start **SURE** on the server.
4. Logon (this creates <ServerPath>\RIS\DATA\INFDB\\*.\*)).
5. Click **Window** in the menu bar and set option **MDI Enabled**.



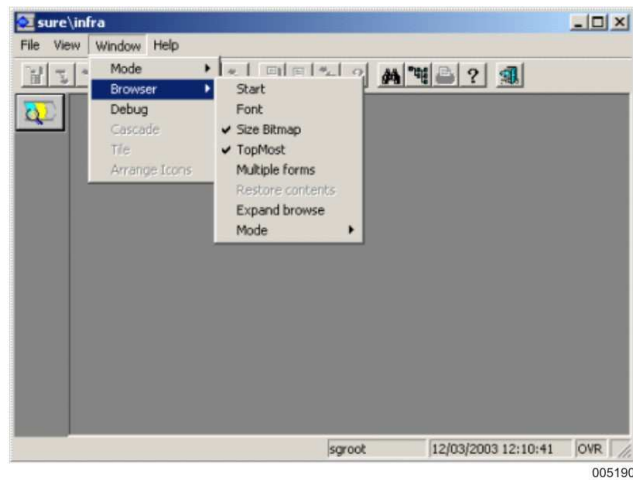
6. A new task button appears now in the Windows task bar.



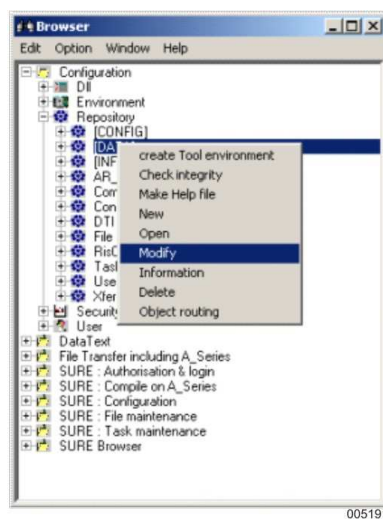
7. Click on this task button to switch to the SURE MDI window.



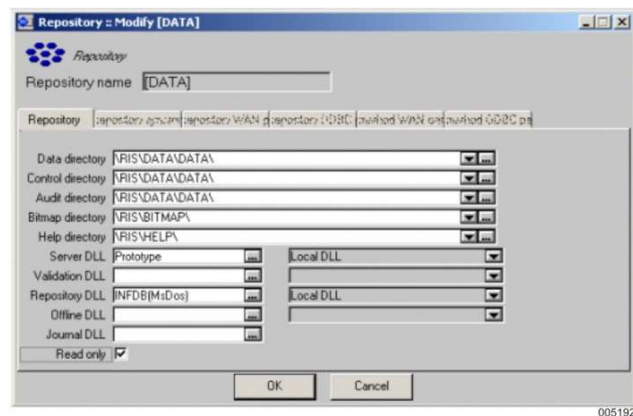
8. From Windows menu, click **Browser**.
9. Start to enter the SURE configuration browser.



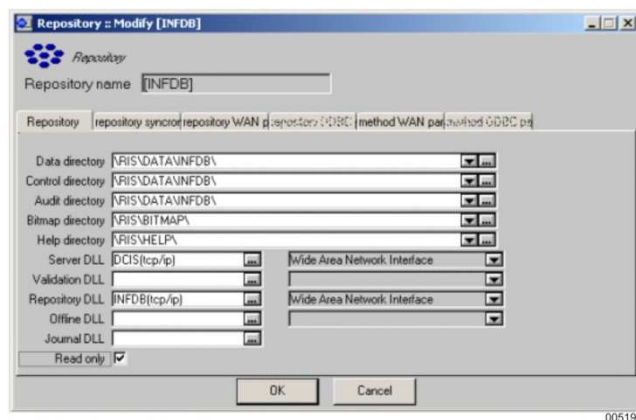
10. Open folder **Configuration** and thereafter **Repository**.



11. Right-click **DATA**, select **Modify**, set "Read Only" and press **OK**.



12. Right-click **INFDB**, select **Modify**, set "Read Only" and press **OK**.



13. Close **SURE**.
14. Create a file called <ServerPath>\RIS\BIN\AW\_OBJ.INI with minimum following static entries:

```
[GLOBAL]
VERIFY=FALSE
CONFIGURE=FALSE
[DATA]
READONLY=TRUE
[DIRECTORY]
TOOL=<ServerPath>
DATA=<ServerPath>
BITMAP=<ServerPath>
DLL=<ServerPath>\RIS\BIN\
SDK=<ServerPath>
[INFDB]
IDENTIFICATION=<IP Address/Port>
LFI=99
TERMINAL=0
SOURCE=04
```



```

WINDOW=4
WINDOWACK=2
MAXBLOCK=3600
[ <IP Address/Port> ]
BUFFER=14500
SYNCTIME=FALSE
[ AR_SURE ]
BITMAP=<ServerPath>\RIS\BITMAP
DLL=<ServerPath>\RIS\BIN\
TOOL=<ServerPath>\RIS\TOOL\AR_SURE

```

The original AW\_OBJ.INI file may be used as a template to create the <server>\Aw\_obj.ini file.

**Note:** The <server>\AW\_OBJ.INI file contains entries that are fixed for an installation. The clients cannot overrule these settings. A site is free to keep more settings in the <server>\Aw\_obj.ini file than the above mentioned, but these settings should not hold screen positions, font settings or other options that should be free to be altered by the user.

Upgrading a server-only installation requires some special actions: the new [DATA] repository and the new [INFDB] repository must be defined again as read-only. The easiest way to upgrade a server-only installation is as follows:

- Un-install the current SURE-client software on the server.
- Check that the following directories are removed:
  - <installation directory>\Ris\Tool\\*.\*
  - <installation directory>\Ris\Data\\*.\*
- Install the new Client software again on the server in the same <installation directory>.
- Perform all actions described earlier in this paragraph to make the new [DATA] and [INFDB] repositories read-only.

### The Client

The client that communicates with a ServerOnly configuration can run using the following files:

- AW\_OBJ.INI or an extract for his personal settings.
- AW\_OBJ.CFG that will start the logon screen.

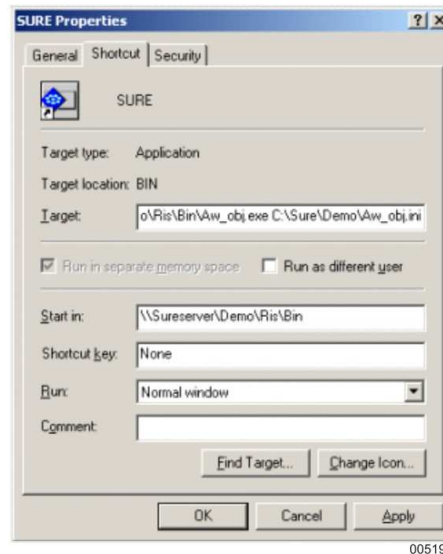
Shortcut attributes:

```

Target    = <ServerPath>\Ris\Bin\Aw_obj.exe <ClientPath>\Aw_obj.ini
Start In  = <ServerPath>\Ris\Bin

```

### Example Shortcut



Target = \\Sureserver\Demo\Ris\Bin\Aw\_obj.exe C:\Sure\Demo\Aw\_obj.ini  
Start in = \\Sureserver\Demo\Ris\Bin

### 8.1.6. (Installation) Multiple Installations on One PC

The program definition links:

- The SURE executable AW\_OBJ.EXE
- A configuration file
- The working directory

The above mentioned entities configure a complete SURE application on a PC. No other files in other directories (including Windows or shares registry entries) are used. For this reason multiple different releases or configurations of SURE can be run from a single computer.

The only requirement is:

- Each new installation requires a unique path (working directory).
- The TITLE attribute in the GLOBAL section of the configuration file is different for every different configuration. This title must be entered at installation time on one of the install dialogs.

When multiple SURE applications run simultaneously on the same PC and a compilation of a mainframe program is started from an editor, then this compilation cannot determine to which SURE application it belongs (because it is started as an independent process).

Some extra requirements solve this problem:

A compilation is controlled by utility AS\_COMP. This utility compiles against a specific configuration by using the SERVICENAME option in the GLOBAL section of configuration file AW\_OBJ.INI. Setting this option requires the AS\_COMP utility to start with the parameter /S<ServiceName>.

### Example

Consider the case that SURE is installed two times on your PC: the first for a repository that contains application system SYS1, and the second for a repository with application system SYS2.

SYS1 requirements:

- SURE for SYS1 is installed on your PC in path C:\Sure\Sys1\\*.\*
- Configuration file C:\Sure\Sys1\AW\_OBJ.INI (for system SYS1) contains in section [GLOBAL] the following option: SERVICENAME=SYS1.

[GLOBAL]

SERVICENAME=SYS1

- Compilation utility AS\_COMP started from a local editor requires now the extra parameter SSYS1.

SYS2 requirements:

- SURE for SYS2 is installed on your PC in path C:\Sure\Sys2\\*.\*
- Configuration file C:\Sure\Sys2\AW\_OBJ.INI (for system SYS2) contains in section [GLOBAL] the following option: SERVICENAME=SYS2.

[GLOBAL]

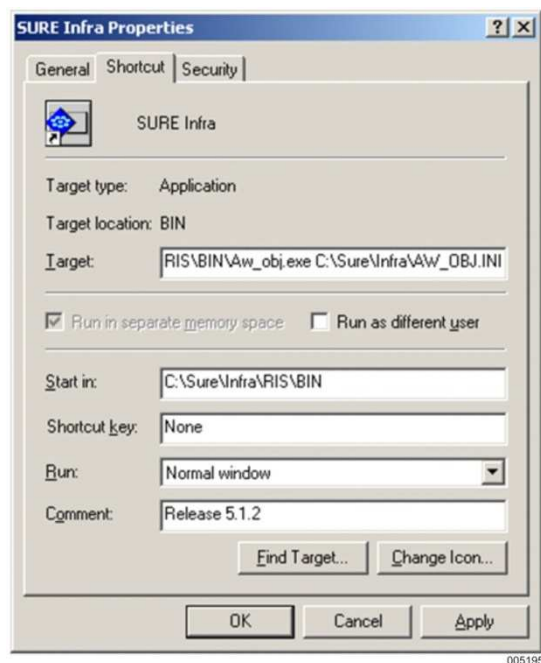
SERVICENAME=SYS2

- Compilation utility AS\_COMP started from a local editor requires now the extra parameter SSYS2.

The reporting facility requires OLE automation services. Every specific installation should use a unique OLE service identifier. This OLE identifier can be defined in the GLOBAL section of the configuration INI file using OLESERVER=<1::9> where 1 is the default.

### 8.1.7. (Installation) The AW\_OBJ.INI File

The SURE Explorer uses a configuration or INI file to store configuration variables. This configuration file is provided as the parameter in the program or shortcut definition. The configuration file is usually defined including a full path name. If this file is not defined, a default name is used called AW\_OBJ.INI, which must be located in the Windows directory.



The configuration file contains many chapters, each providing different variables. Some of these variables must be defined manually; others are set automatically by the software (these are system variables).

This documentation describes the variables used by the SURE Explorer. Variables that are not mentioned in this document are system variables, and should not be changed manually (unless explicitly described in the documentation).

The default values of Boolean variables are in bold.

[GLOBAL]

Servicename = <id>

This is only relevant if SURE is installed multiple times on the same PC. The servicename is used as a key for the compilation utility. Refer to "8.1.6 (Installation) Multiple Installations on one PC."

Sccprovider = <**false**,true>

If sccprovider is true, then SURE will add a sccprovider-key in the registry. This enables the SCC interface of SURE. If sccprovider is false, then the registry-key is removed.

Trace = <tracefile>

The tracefile can be used for debugging purposes. The tracefile is refreshed each morning.

## [DATA]

Readonly = <**no**,yes>

The READONLY attribute defines the local data repository as read-only. This is required if the local data repository is located on a shared network drive and multiple users access this repository as their configuration data. In this case, each user has the same configuration (defined visible objects).

## [EXE]

WINZIP = <name>

The location of Winzip.

## [INFDB]

Identification = <host/port>

The hostname (or the IP address of the host) followed by the port number. This identification points to another chapter in this INI file with details about the id.

For example. If the identification = HOSTA/1234 then there must be a chapter in the INI file called [HOSTA/1234] with details about that connection.

## [host/port]

Buffer=14500

Synctime=false

Mode=unattended

This chapter creates a connection-id for <host/port>. If the identification in chapter [INFDB] routes to this connection-id, then this port is opened.

## [SUREHELP]

<helpfile> = <id>

This chapter can be used to customize the Help menu in the SURE Explorer. This example puts <id> as an extra entry in the Help menu. A click on <id> opens the corresponding <helpfile>.

## [STATUS]

Maxaa=<nr>, default 30

Maxw=<nr>, default 30

Maxs=<nr>, default 30

Maxc=<nr>, default 5

Maxmsg=<nr>, default 10

This chapter can be used to customize the output of the status of the mix for a specific usercode. (Tools → CP/NX environment → Status).

MAXAA = the max. number of active entries (AA USER ..)

MAXW = the max. number of waiting entries (W USER ...)

### [SURE]

Copy-prefix = <prefix>

Files that are copied from SURE to disk through the COPY function in the explorer, will get the <prefix>.

Attachment = unzipped

Attachments for files and tasks are not zipped before they are stored in the repository. This option is required if you install a site-specific email library.

Verifylogoff = <**true**,false>

If verifylogoff is true, then an extra verification message is given when you close the SURE explorer.

### [POWERBUILDER]

Silent = \*.PBG:<default project>:<default file type>

This enforces that all PowerBuilder-PBG-files are loaded and maintained automatically, and that they all get the given project and file type. If no <default project> is defined, then the project of the current task is used. If no <default file type> is defined, then the file-extension is used. It is possible to extend the SILENT option for other file extensions.

### [CUSTOMLOGON]

Hidepassword = <**false**,true>

Hideaccesscode = <**false**,true>

Hideaccesscodepwd=<**false**,true>

Hidechargecode=<**false**,true>

These variables make it possible to hide the fields Password, Accesscode, Accesscode-pwd, or Chargecode on the logon screen.

Var1 = <name>

Override the default name of the first logon field.

Var2=<name>[,true]

Var3=<name>[,true]

Var4=<name>[,true]

Var5=<name>[,true]

Override the default names of the other four fields in custom logon. If a key is not used then the corresponding field is hidden. Option "TRUE" indicates that the field is a password. If one of these options is used, then VAR1 must be defined too.

Saveallowed=<**true**,false>

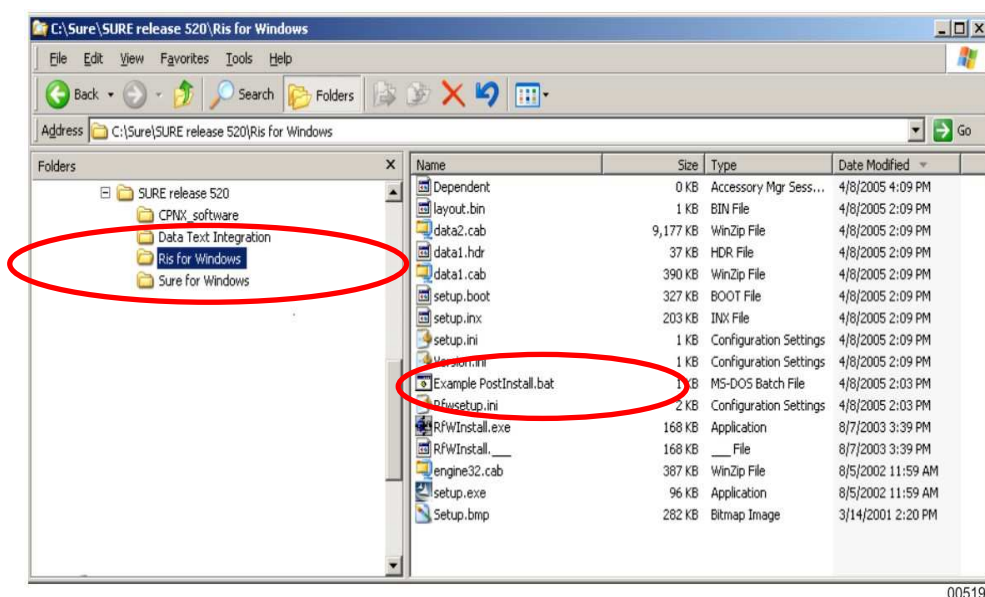
Do not allow saving the logon credentials.

### 8.1.8. Customized Post Installation Actions

It is possible to define custom post installation actions.

The SUREforWindows client installation process searches for a file called "PostInstall.bat" in the installation sub-directory "Ris for Windows." If the post-installation bat file is found, it will be started after the regular installation is done.

#### Example



The following example post installation bat file is already placed in the "RIS for Windows" sub-directory:

..\SURE release 520\Ris for Windows\Example PostInstall.bat

```
rem Example custom post installation batch. rem Rename this file to
rem "PostInstall.bat" and keep it situated
rem in the main installation directory.
rem
rem This batch file is called with the following argument:
rem %1 = target directory of the installation (the folder where
rem      RfW has been installed)
rem
rem Copy redistributable DLLs from RIS\BIN to Windows\System
rem directory, to accommodate SCC interface.
if not exist %WINDIR%\System32\Mfc71.dll copy %1\Ris\Bin\Mfc71.dll
%WINDIR%\System32
if not exist %WINDIR%\System32\Msvcp71.dll copy %1\Ris\Bin\Msvcp71.dll
%WINDIR%\System32
if not exist %WINDIR%\System32\Msvcr71.dll copy %1\Ris\Bin\Msvcr71.dll
%WINDIR%\System32
rem
rem Run client application that modifies the INI file
C:\MyTools\UpdIni.exe %1\AW_OBJ.INI
```

You can change the contents of this example file to your own needs.

Change the name of this example file to "PostInstall.bat" to enable it.

The bat file will be started with the "target installation directory" as argument %1.

The example bat file shows the following examples:

- How to use argument %1 (target installation directory) in the bat file.
- How to copy the required redistributable DLLs from RIS\BIN to Windows\System directory, to accommodate SCC interface.
- How to run a client application that modifies the INI file.

### 8.1.9. SURE Triggers New Installation According to Versions

Updating the SURE client software may require much effort when having a large network. For this reason, SURE supports different styles of installation:

- Local client installation on each applicable workstation.
- Server based installation with personalized options.
- Full server installation.

#### Server installations (options b and c)

In the case of a server installation, the SURE PC-software is placed on a central server that is accessed by the workstations of the developers. Because of this central installation, a software update is reasonably easy to install for the first time. Upgrading a server installation to a newer SURE release is cumbersome.

The trade off for a server installation is that the network throughput must be sufficient (100KBS). It does not support wide area networks with a low bandwidth.

#### Local client installations (option a)

In the case of a local client installation, the SURE-PC-software is placed on the workstation of a developer. Installing the SURE PC-software as a local client is very easy for initial and upgrade installations.

The trade off is that it is difficult to synchronize the upgrade to a newer SURE release when the SURE-PC-software is installed locally at many workstations.

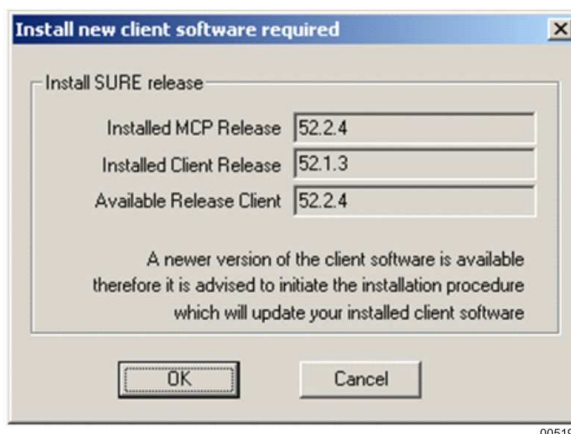
#### Solution

SURE can (optionally) check the version of the PC-software at logon time and may initiate a local client installation if that is required. This combines the benefits of an easy local client installation with the benefits of a synchronized server installation.

#### Example

The following dialog is shown after logon when the versions of the installed client release and the available release for the client do not match.





- In this case, the version number of the SURE software on the MCP is 52.2.4.
- The version number of the “available” SURE PC-software is also 52.2.4.
- The version number of the client software that is now installed on this workstation is 52.1.3.

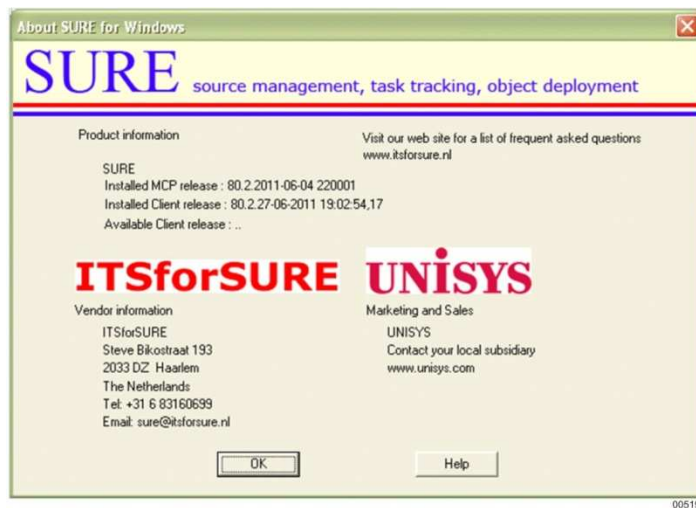
Obviously, a new version of the mainframe software was installed and the corresponding new version of the client software was “made available.”

**Note:** *The version of the mainframe software is not relevant to display the above screen.*

Click **OK** to start the installation, which will install the available client software.

Click **Cancel** to skip the installation, however the same message will then be issued after the next logon. A possible reason to skip the automatic installation is for example when the developer works at home through a slow connection, and he does not want to install the SURE PC-software through that slow connection. A solution is to install the PC-software later from a release CD-ROM. It is possible to work with a SURE-client that has another software version number than the ClearPath server software, but some actions may issue an error message.

The about dialog now contains the various versions according to the next example.



### MCP version information

- Major release version      In this example: 80
- Minor release version      In this example: 2
- Build timestamp            In this example: June 4, 2011 at 22:00:01

### Client Version Information

- Major release version      In this example: 80
- Minor release version      In this example: 2
- Build timestamp            In this example: June 27, 2011 at 19:02:54

The Major and Minor version numbers of MCP and client must be equal. The build timestamp is used for more detailed verification.

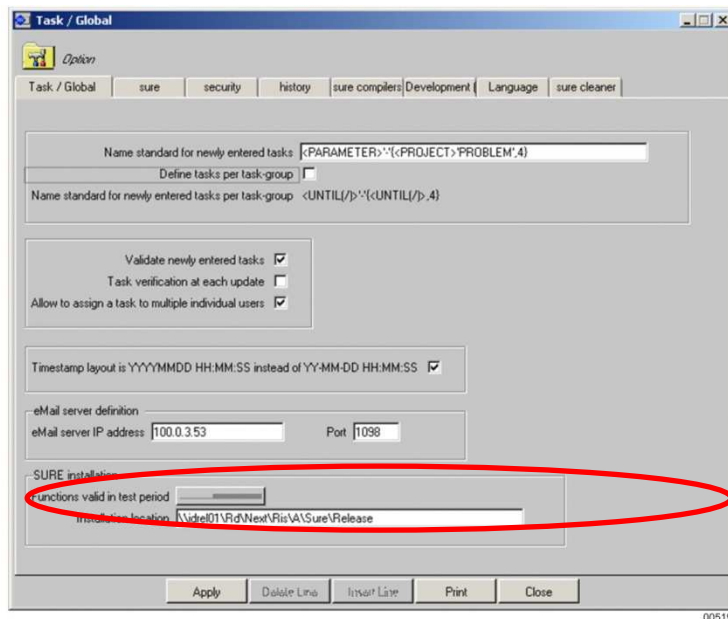
### Technical details and considerations

The following procedure is required to trigger the automatic client installation.

The delivered SURE software package contains the SURE ClearPath server modules and the SURE client modules. The two modules are compatible and have the same version number.

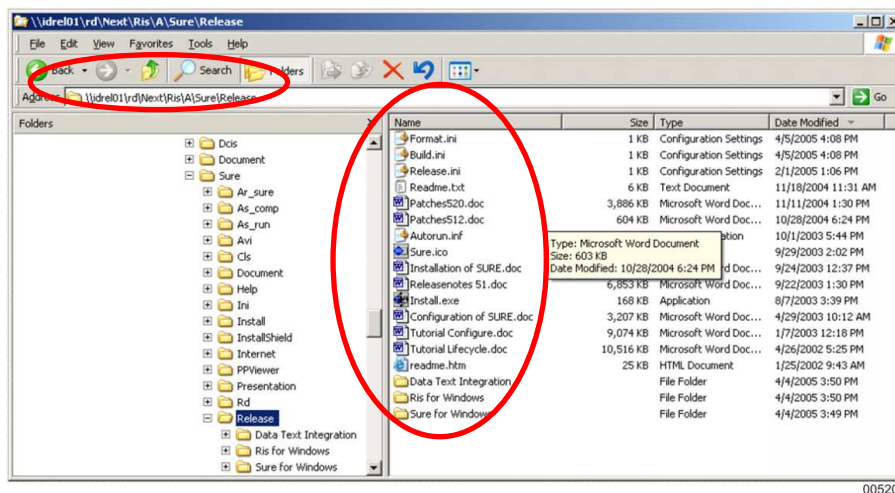
The entire installation consists of three steps:

1. Copy the new version of the SURE ClearPath software to the correct location on the mainframe.
2. Define the location of the available client software in SURE:
  - a. Click **Global Options**.
  - b. Select the field **Installation Location**.



The location of the available client software is here:  
 “\\vidrel01\rd\Next\Ris\A\Sure\Release”

- c. Copy the new version of the SURE-PC-software (a copy of the CD-ROM image) to that location.



The SURE client software is available in directory  
“\\idrel01\Rd\Next\Ris\A\Sure\Release”

This directory contains three files that control the version number of the release:

- Release.ini
- Format.ini
- Build.ini

The number in file Release.ini defines the first part of the software version number (80 in the first example).

The number in file Format.ini defines the middle part of the software version number (two in the first example). This number is raised by one if the layout of the interface between PC and mainframe is changed.

The number in file Build.ini defines the last part of the software version number. It is the timestamp when the PC-software was built. This timestamp does not play a role in the version-check of the automatic installation.

If any of the three files (step c) is missing, or if the installed location (step b) is not defined, then the automatic installation dialog is skipped.

### **RFW.INI**

It is possible to install multiple instances of SUREforWindows on one PC. Information about each SUREforWindows installation is kept in file RFW.INI in the Windows directory. The following example RFW.INI file contains information about two installations: an installation for the ‘Infra’ repository and an installation for a “Demo” repository.

```
[GLOBAL]
APPLICATION1=SURE for Windows\Infra
APPLICATION2=SURE for Windows\Demo
[SURE for Windows\Infra]
INSTALLED1=RIS for Windows,6.3.0
INSTALLED2=Data Text Integration,1.1.1
SETUP=SURE for Windows
DIRECTORY=C:\Sure\Infra
INSTALLED3=SURE for Windows,5.2.0
[SURE for Windows\Demo]
INSTALLED1=RIS for Windows,6.3.0
INSTALLED2=Data Text Integration,1.1.1
SETUP=SURE for Windows
DIRECTORY=C:\Sure\Demo
INSTALLED3=SURE for Windows,5.2.0
```

When a SUREforWindows client is installed for the first time on a PC, the installation process asks for a program folder name. This program folder name is used for three purposes:

- Create a program folder in the Windows startup menu.
- This name becomes the title of the SUREforWindows client (in the title bar).
- This name is placed in the AW\_OBJ.INI file ([global] title).
- This name is used as a key for the RFW.INI file.

The automatic installation searches for an application with the program folder name in the RFW.INI file. If the program folder name is not found in the RFW.INI file, then the automatic installation will be aborted with an error message. If the program folder name is found, then the software will be installed in the directory that is defined for the program folder name.

### **Example**

A user starts "SURE for Windows\Demo" and logs on. An automatic installation is required and the user clicks on the OK button to start it. The automatic installation is started with parameter "SURE for Windows\Demo." The automatic installation process opens the RFW.INI file and searches for application "SURE for Windows\Demo." If that application is found then the software is installed in the directory that is defined in paragraph [SURE for Windows\Demo]. This is directory C:\Sure\Demo.

## **8.2. Installation on the Mainframe**

Installation of the SURE software is supported by an online program that allows defining the installation parameters. This program generates an installation job that will do the actual work.

In theory, the installation procedure automates the complete process. In standard installations on ClearPath Enterprise Servers, this is true; however, there may be circumstances where dethoughted compiler names or special security settings jeopardize an automatic installation.

This chapter contains valuable information about the structure of the SURE mainframe software. We strongly advise this chapter to be read by the technical support staff, especially the database administrator. If you have questions about this chapter, then contact Infra Design.

The actual installation procedure is described in "Mainframe Install Procedure."

### 8.2.1. (Installation) Mainframe Application Structure

The SURE software consists of the following components:

- |    |   |                                      |
|----|---|--------------------------------------|
| 1. | A DMSII database.   | INFDB                                |
| 2. | An initialization file.   | RESPECT/TITLES                       |
| 3. | End user interface programs for SURE on the PC's (the SURE Explorer interface). | OBJECT/RIS/API/=                     |
| 4. | An end user interface program for ET type of terminals.                         | OBJECT/RIS/MENU                      |
| 5. | A library.  | OBJECT/RESPECT/LIBRARY.              |
| 6. | Approximately 100 other supporting objects.                                     | OBJECT/RESPECT/= and<br>OBJECT/RIS/= |
| 7. | A CD-ROM containing the SURE PC software.                                       |                                      |

### 8.2.2. (Installation) Repository INFDB

INFDB is a DMSII database and contains 11 datasets. Database INFDB is also called the "repository" or the "SURE repository." Sources and program attributes are loaded in this repository.

The database is initialized when the software is installed for the first time.

It is possible to create multiple INFDB repositories on your system. In that case, the installation procedure "create a new repository" must be executed multiple times.

If multiple repositories are present, then each of them must be installed under a separate usercode. The user is free to choose the directory where database INFDB is installed. We advise to install the database under a usercode and not in the global directory "\*". This makes it easier to install an extra INFDB database later on.

All SURE programs are designed in such a way that they can work for multiple repositories at the same time. This is only possible if the SURE software is compatible with all those databases. Therefore, the database-timestamp and description-update-level of each repository must be equal to those values stored in the SURE objects (otherwise, the SURE object will fail with a DMOPEN error).

The installation procedure ensures that each newly created INFDB contains the correct database-timestamp and description-update-level.

The database will go out of the mix when it is not in use by any SURE program.

The following steps are required to close the database (Refer to "8.2.5 (Installation) Object Files"):

1. Disable the SURE Explorer interface in COMS.
2. THAW the permanently frozen libraries RESPECT/LIBRARY and RESPECT/LIBRARY/SURE.

### 8.2.3. (Installation) SURE Objects and Database Compatibility

The Unisys DMSII software has a strong relationship with the objects that use a DMSII database. At compilation time, the update timestamps of database structures are compiled into the object. This mechanism is very secure, but rather static when such object files need to be delivered to a customer.

The SURE objects work with a DMSII database (INFDB).

If a customer installs new SURE objects, then these new objects must be capable to communicate with any existing INFDB of the customer. The objects may not get a DMOpen error. The SURE sources are NOT delivered to the customer, so it is NOT possible to recompile the SURE objects at the customer's site.

A SURE object is compiled at a central place, and then delivered to many customers.

The object must be capable to communicate with all INFDB's, installed at any customer.

This is only possible if all these INFDBs are "direct children" of the original INFDB, and if the SURE objects are compiled against the description file of the original INFDB.

We call this original INFDB "the mother of all INFDBs," and its description file "the mother of all description files."

The first mother description file was created during DMSII release 39. Since then, we always saved this mother description file carefully and upgraded it for each DMSII release. We did not make any other database change since then.

The result is that we have a mother description file for each DMSII release, and we place all these mother description files on the release tape or CD-ROM. These description files are called, DESCRIPTION/INFDB/INFRA/DMSxx where xx is the DMSII release.

Although the layout of the INFDB never changes, a customer may still have to upgrade his INFDB for one of the following reasons:

- He upgraded to a newer version of the DMSII software.
- He got limit errors in the INFDB and had to define bigger populations or area sizes in his DASDL/INFDB source.
- He moved the INFDB to another location (usercode/pack).
- All these actions are possible without recompiling the user software.

If a customer receives a new SURE release, then the new SURE objects may not be compatible with his existing INFDB.

- The customer has installed an INFDB that is a child of mother description file 461.
- The SURE objects that he runs are compiled with mother description file 461.
- The customer INFDB is upgraded to DMSII release 471 shortly after the installation.
- At this moment the SURE software will still run, because the objects are compiled against mother description file 461 and the customer database is a (grand) child of that mother description file.
- Mother-description 461 was delivered to the customer, upgraded at the first installation with pack names, and later on upgraded to 471, user software still compatible.
- The customer receives a new SURE release and the SURE objects in this new SURE release are now compiled with mother description file 471.
- These new SURE objects are not compatible because the customer description file 471 is not a (grand) child of mother description file 471 (used for the compilation of the new SURE objects), but of mother description file 461.
- The following picture illustrates the compatibility between the SURE objects and the INFDB.



005201



### 8.2.3.1. (Installation) Compatibility at a NEW Installation

During a NEW installation of the SURE software the following actions are done:

1. Check the release level of the DMSII software that the customer is currently running.
2. The mother description file of that release level is placed on disk.
3. A customer-dasdl is generated and this customer-dasdl is exactly the same as the mother-dasdl, except for the packnames and usercodes. Option UPDATE is set in the customer-dasdl.
4. The customer-dasdl is compiled against the mother description file.

The result of this dasdl compilation is a customer description file that is a “child” of the mother description file.

The only differences in the two dasdls are usercode/family specifications. This results in an UPDATE dasdl compilation. It does not require any reorganization. A control file specification is used in the dasdl. All this makes it not necessary to recompile the SURE objects.

**Note:** A SURE object can always find the location of the INFDB control file, because that location is mentioned in the file RESPECT/TITLES. Each SURE object reads the INFDB location from respect/titles and file-equates the database with that database location.

The usercode/family specifications in the dasdl are necessary for the situation that the database is not active. In that case, the first run of a program will also start the database, but this first program can be started from any usercode. If the first program is started from a usercode NOT equal to the usercode where the tailored database software is placed, and the tailored software is declared in the dasdl WITHOUT a usercode/pack then that program will get a DMOpen error.

At the end of a NEW installation, we have an INFDB that is a child of the mother INFDB, and a description file that is a child of the mother description file. The delivered SURE objects will run smoothly without DMOpen errors.

### 8.2.3.2. (Installation) Compatibility at an UPGRADE Installation

During an UPGRADE installation, we check if the new SURE objects are compatible with the INFDB of the customer.

If the new SURE objects are not compatible with the INFDB then the following actions are done:

1. Check the release level of the DMSII software that the customer is currently running.
2. The mother description file of that release level is placed on disk.
3. Verify that the correct DASDL/INFDB of the customer is on disk.

4. Compile the customer DASDL/INFDB against the mother DESCRIPTION/INFDB.

The compilation can have two results:

- No reorganization of INFDB necessary (if the customer dasdl is still equal to the mother dasdl, except for usercodes and pack names).
- Reorganization of INFDB necessary (if the customer also changed areasizes or blocksizes).

This reorganization must be skipped, because the INFDB was already reorganized.

The final result of the dasdl compilation is a description file that is again a child of the mother description file (that was used to compile the objects). The new description file also contains the usercodes, pack names and other physical attributes of the customer (mentioned in the customers DASDL/INFDB).

5. Create a new control file from the newly created customer description file:
  - RUN SYSTEM/DMCONTROL("DB=(uscd)INFDB ON pack RECOVER INITIALIZE")
6. From now on, the new SURE objects are compatible with the existing INFDB of the customer.

### 8.2.3.3. (Installation) Compatibility Direction

SURE objects can only work at the customer site if the current DMSII software version of the customer is equal or higher than the DMSII-version of the mother description file that was used to compile the SURE objects.

#### Example

- The SURE objects are compiled against mother description file 471.
- Customers who work with DMSII-software version 46x or lower cannot use these objects.
- Customers who work with DMSII-software version 471 or higher can use the objects.
- The SURE 5.1.2 objects are compiled with a mother description file of DMSII version 47.1.

### 8.2.4. (Installation) RESPECT/TITLES

File RESPECT/TITLES is the repository-control-file. Each SURE program has the same initialization routine: when a SURE program is started, the first action is to search for file RESPECT/TITLES and to read this file.

File RESPECT/TITLES is a sequence data file created by the installation program. This file may be modified manually and distributed over various packs according to the requirements.

Each record in the file contains a variable used by the SURE software.

### Example

Example of file RESPECT/TITLES (SEQDATA)

```
00000100 INFDB-LOCATION  = (SCRATCH)INFDB ON IDRD1
00000200 OBJECT-LOCATION = *OBJECT/ ON IDRD
```

- Key INFDB-LOCATION defines the location of INFDB/CONTROL, CONTROL, the database control file.
- Key OBJECT-LOCATION defines the location of the SURE software.

In this example, the INFDB repository is located in directory (SCRATCH) on IDRD1.

The software OBJECT/= is located in directory "\*" on IDRD.

It is possible to install multiple SURE-repositories or multiple sets of SURE-objects on a single mainframe. For example: when each department has a separate repository.

The purpose of RESPECT/TITLES is to determine the location of a specific repository and its corresponding SURE objects.

Each SURE program has the same initialization routine.

When a SURE program is started, the first action is to search for file RESPECT/TITLES and to read this file.

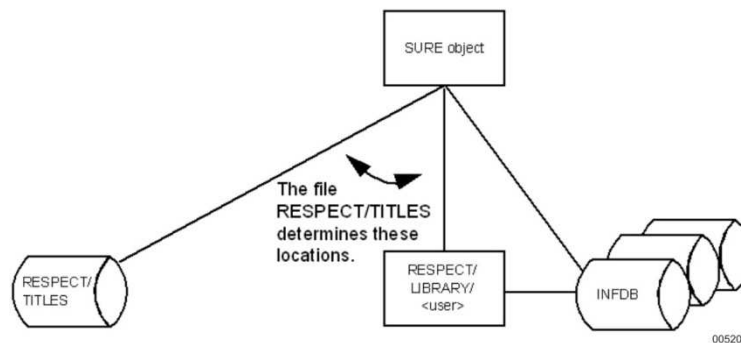
The following initialization keys can be used in the RESPECT/TITLES file.

Initialization key	Mandatory/ Optional	Explanation
INFDB-LOCATION=(<uscd>)INFDB ON <pack>	Ma	The place of the INFDB control file
OBJECT-LOCATION= <directory> ON <pack>	Op	Default value = "OBJECT/".The directory where the RESPECT and RIS objects are placed on disk.
WFL-LOCATION= <directory> ON <pack>	Op	Default value = <empty>. SURE batch jobs can be generated with a prefix. For example "REL51".
SCREEN-TIMELIMIT= <nr of seconds>	Op	Each RIS/MENU screen has its own timeout value. RIS/MENU goes to end of task if no input was done on a screen for that amount of seconds. This option overrules the default timeout value of all the screens in RIS/MENU.

Initialization key	Mandatory/ Optional	Explanation
SITE-FUNCTION = OBJECTNAMES	Op	This option improves the initialization procedure of the SURE-software if site function "Objectnames" is used.
SITE-FUNCTION = OBJECTNAMES-ALWAYS	Op	This option improves the initialization procedure of the SURE-software if site function "Objectnames-always" is used.
SITE-FUNCTION = PHYSICAL ATTRIBUTES	Op	This option improves the initialization procedure of the SURE-software if site function "Physical attributes" is used.
SITE-FUNCTION = SECURITY	Op	This option improves the initialization procedure of the SURE-software if site function "Security" is used.
SITE-FUNCTION = EVENT	Op	This option improves the initialization procedure of the SURE-software if site function "Event" is used.
SITE-FUNCTION = EMAIL	Op	This option improves the initialization procedure of the SURE-software if site function "Email" is used.

The record in the RESPECT/TITLES file that starts with key INFDB-LOCATION contains the location of the INFDB repository. The program will open that database.

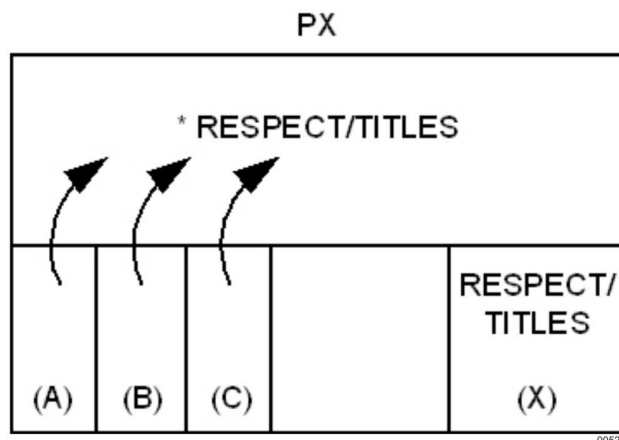
The record in the RESPECT/TITLES file that starts with key OBJECT-LOCATION contains the location of the SURE objects. If the program has to access a SURE library or if the program has to start another SURE program, then this object-location is used to start or access the correct SURE object.



The visibility of the file RESPECT/TITLES together with the ClearPath server file location mechanism is used to connect groups of users to the same repository. Installation of RESPECT/TITLES on pack PX will set the repository for all the users that

will have PX as their primary family. If a user has a RESPECT/TITLES under his user code, then the location in that repository-control-file will be used.

### Example



All users on pack PX are routed to the same repository except for user (X). He has an own file RESPECT/TITLES.

A SURE program starts another SURE program with the same RESPECT/TITLES. This ensures that they started SURE program will access the same repository as the calling SURE program.

### SITE-FUNCTIONS

If a SURE site library is defined (on Global Options dialog click the tab "Sure" and then click the field "Site library)" then SURE checks and activates the defined functions in that site library at start-up time. This is done by RESPECT/LIBRARY. RESPECT/LIBRARY starts the site library for each possible entry point to check if that entry point is defined. This may give some overhead.

This default check for ALL entry points is skipped if the applicable site functions are defined in the RESPECT/TITLES file through the SITE-FUNCTION method. If one or more site-functions are declared in the RESPECT/TITLES then only those site-functions are checked and activated by RESPECT/LIBRARY. Notice that this declaration in RESPECT/TITLES is optional.

## 8.2.5. (Installation) Object Files

Each SURE program has the same initialization routine.

When a SURE program is started, the first action is to search for file RESPECT/TITLES and to read this file.

The record in RESPECT/TITLES that starts with key INFDB-LOCATION contains the location of the INFDB repository. The program will open that database.

The record in RESPECT/TITLES that starts with key OBJECT-LOCATION contains the location of the SURE objects. If the program has to access a SURE library or if the program has to start another SURE program, then this object-location is used to start or access the correct SURE object.

A SURE program starts another SURE program with the same RESPECT/TITLES. This ensures that they started SURE program will access the same repository as the calling SURE program.

### 8.2.5.1. (Installation) SURE Explorer Interface

The programs in directory OBJECT/RIS/API/= form together the SURE Explorer interface on the mainframe.

It is possible to install multiple SURE repositories on a single mainframe, each with its own SURE Explorer interface. Each SURE Explorer interface for any INFDB repository requires an separate set of RIS/API/= objects in a separate usercode directory. We call this directory the SURE-Explorer-datacom-directory.

Usually there is only one SURE Explorer interface each repository.

Each SURE Explorer interface must be connected to a unique TCP/IP port number. This TCP/IP port is used for the communication with the PC-environment.

Each SURE-Explorer-datacom-environment requires the following:

- A set with objects OBJECT/RIS/API/=
- A file RESPECT/TITLES
- A file RISAPPLICATION/TITLES/WINDOWS
- A "SURE Explorer interface" program-definition in COMS/UTILITY

File RESPECT/TITLES is used to link the SURE Explorer interface to the correct repository, and to locate the other SURE objects.

File RISAPPLICATION/TITLES/WINDOWS is used to determine the TCP/IP port number and the work-pack. This file is created automatically during the installation procedure (install a new repository).

The contents of this file are as follows:

```
00000001WINDOWS@
00000002OBJECT/RIS/API/LIB@
00000003OBJECT/STATISTICS/TIMING@
00000004OBJECT/RIS/API/TPP@
00000005OBJECT/RIS/API/FTP@
00000006@
00000007@
00000008@
00000009@
00000010@
```

```
00000011@
00000012@
00000013<debugfile pack>@
00000014<work pack>@
00000015@
00000016<filetransfer-pack>@
00000017<tcpip port-nr>@
```

- The values on lines 1 through 12 and 15 are fixed.
- The pack names on lines 13, 14 and 16 are optional. When they are not entered, then the default family of the SURE-Explorer-datacom-directory is used instead.
- The <TCP/IP port-nr> on line 17 is mandatory, and the default value is 1025.

The SURE Explorer interface must be defined as a COMS-program in COMS through COMS/Utility.

The SURE Explorer interface can only be started by "enable <COMS-program>" and the only way to terminate the SURE Explorer interface is by "disable <COMS-program>":

```
<mixnr COMS>SM ENABLE PROGRAM <SURE-Explorer-interface>
```

```
<mixnr COMS>SM DISABLE PROGRAM <SURE-Explorer-interface>
```

### 8.2.5.2. (Installation) Terminal Emulation Interface

Program RIS/MENU handles the terminal emulation interface for SURE. This program should be placed on disk in a directory that is visible for all programmers.

OBJECT/RIS/MENU is designed in such a way that it can work with multiple repositories at the same time. It is not necessary to install this program again for each repository.

The program can be started from CANDE:

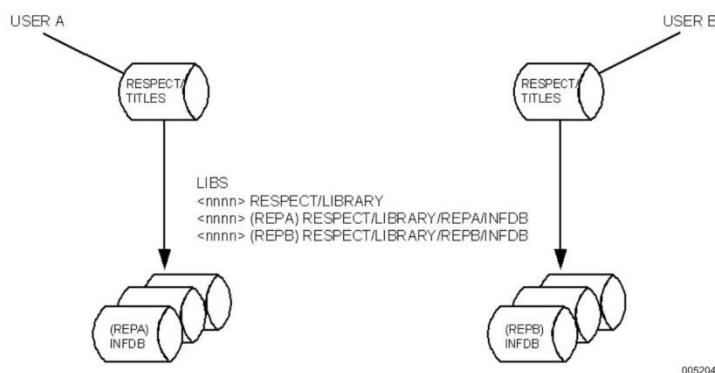
```
RUN RIS/MENU
```

or through COMS (if a COMS-program and a corresponding COMS-window (remote-file) are defined):

```
? ON <SURE window>
```

### 8.2.5.3. (Installation) RESPECT/LIBRARY

Libraries RESPECT/LIBRARY and RESPECT/LIBRARY/SURE are shared libraries using a dynamic selection procedure. This construct will create one additional library instance each repository. Therefore the LIBS entries will show one library RESPECT/LIBRARY and one additional for every repository called (<usercode>)RESPECT/LIBRARY/<usercode>/ INFDB. These libraries are frozen permanent, for this reason they need to be thawed if the database needs to be terminated.



This example shows two users, each using another INFDB repository. The RESPECT/TITLES of user A links that user to repository (REPA) INFDB, and the RESPECT/TITLES of user B links that user to repository (REPB) INFDB, RESPECT/LIBRARY can be installed in the same directory as the SURE batch programs.

Each program does a call on RESPECT/LIBRARY. Since this is a permanent frozen library, it has to be thawed manually to terminate it.

### 8.2.5.4. (Installation) Batch Programs

The SURE software can address multiple repositories on the same machine simultaneously. Except for the SURE Explorer interface, it is not necessary to install the SURE software multiple times; one set with SURE objects can be used to address multiple repositories.

The programmers do not require any direct access to the RESPECT batch programs. Some batch programs can be started through online functions, and these online functions know where the corresponding batch program is installed. Other batch programs are only necessary in the daily evening batch. The change control staff has to add the correct run-usercodes and run-families in the jobs that belong to the daily evening batch of the RESPECT package.

The installation procedure will place all objects in one directory, and the user can choose that directory at installation time. The user is free to choose any directory for the object files. It is not mandatory to install the objects global under a "\*" directory. If the objects are not installed under a "\*" directory then each time a program is started the correct usercode has to be placed in front of the object name.

If a batch program is started through an online function, then file RESPECT/TITLES is used to obtain the correct directory where the batch program is placed.



### 8.2.6. (Installation) Mainframe Install Procedure

The SURE mainframe software can be delivered as follows:

- Through CD-ROM INFRASYSTEM-yymm (yyymm is the year and month of the release).
- Through a wrapped and zipped container file on a CD-ROM, by email or from the internet.

**Note:** Before the installation is started one has to decide where the software has to be installed.

The following locations need to be addressed:

- The location of the INFDB repository (database)
- The location of the SURE objects
- The location of the SURE Explorer interface
- The location of the "repository location file" is RESPECT/TITLES

SURE can be installed in any directory on your system. You can choose your own usercode and family where SURE must be installed.

**Note:** The user must be logged on through CANDE with the usercode/family where you want to start the installation.

This usercode/family must meet the following conditions:

- Your usercode must be equal to the usercode where the INFDB repository must be installed.
- Your family or your alternate family must be equal to the family where you want to install the file RESPECT/TITLES.
- RESPECT/TITLES must be installed under your usercode or under global directory "\*" .

#### Example

If you are logged on with usercode SUREDEMO and your current family specs are FAMILY DISK = DEMOPK OTHERWISE DISK then the following configurations are possible:

INFDB location	SUREDEMO ON any-family	OR * ON any-family
RESPECT/TITLES	(SUREDEMO) ON DEMOPK	OR * ON DEMOPK
		OR
	(SUREDEMO) ON DISK	OR * ON DISK

**Note:** The installation usercode must be a PRIVILEGE usercode (PU); otherwise, the installation will fail.

The SURE demonstration software consists of one CD-ROM: INFRASYSTEM-yymm (yyymm is the year and month of the release). The label on the CD-ROM gives the correct yyymm.

Special releases can be delivered in a zipped container file on a CD-ROM or through e-mail.

It is possible to install the software directly from the CD-ROM to your machine, but it is also possible to copy the software first from the CD-ROM to a diskpack on your machine, and to install the software from that disk pack.

If the software is delivered through a wrapped container file, then you must choose installation option C (below); otherwise, you can choose between installation options A and B.

Option A. If you want to install the software directly from the CD-ROM:

- A-1 Copy the initial installation DO-file from the tape or CD-ROM:  
COPY (INFRA)RESPECT/SETUP AS RESPECT/SETUP FROM  
INFRASYSTEM-yymm
- A-2 Process this DO-file:  
DO RESPECT/SETUP

Option B. If you want to copy the software from the CD-ROM to a disk pack and install from that pack:

- B-1 Copy the files from the CD-ROM to the pack:  
COPY = FROM INFRASYSTEM-yymm TO <packname>(PACK)
- B-2 The software is placed on disk under usercode (INFRA) on <packname>.  
Process the installation DO-file:  
DO (INFRA)RESPECT/SETUP("<packname>(PACK)")

Option C. If you want to copy the software from a WRAPPED container file to a diskpack and install from that pack:

- C-1 The name of the container file is "SureNNNcont" or "SureNNNcont.zip" (NNN is the release number). The container file can be attached to an email, it can be placed on a CD-ROM, or it can be downloaded from our Web site. This container file contains the object files and description files for ClearPath Enterprise Servers plus some extra files that are required during installation. It is possible that the container file is zipped.
- Copy the container file from your PC to the mainframe (through ftp) as follows:
- If the container file has extension "ZIP" then it is zipped. Unzip the container file.
  - Open program "MS-Dos prompt" on your PC.

- Go to the directory on the PC where the container file placed.
- Enter command (in lowercase): ftp <your ClearPath Enterprise Servers hostname>.
- Log on (in lowercase): first your usercode and then your password.
- Enter command (in lowercase): type binary.
- Enter command (in lowercase): put <container filename>.

It will take some time to copy the container file to the mainframe. The file is copied to the usercode that you entered during your ftp session, on the default primary family of that usercode.

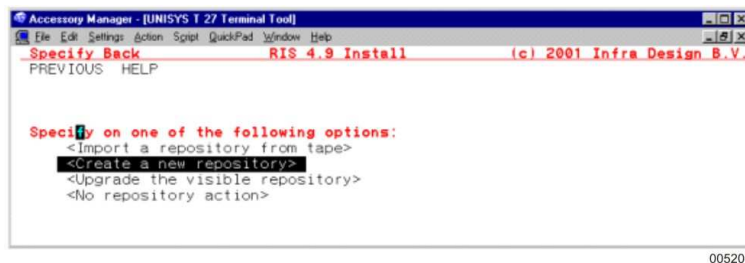
- C-2 Arrived on the mainframe, the container file must be unwrapped as follows:
- Log on to CANDE with the usercode where the file is placed.
  - WFL UNWRAP \*= OUTOF <container name> TO <yourpack>(PACK, RESTRICTED=FALSE).
- C-3 The software is available without restrictions under usercode (INFRA) on <packname>. Process the installation DO-file:  
DO (INFRA)RESPECT/SETUP("<packname>(PACK)")

Additional files are copied from the tape or CD-ROM and the installation program RIS/INSTALL is started. This program presents the installation screen.

## 8.2.7. (Installation) Mainframe Install Program

### 8.2.7.1. The First Installation Screen

Define the installation action: new, upgrade, import.



- <Import a repository from tape>

This option will install a repository from tape INFDBTAPE-yyymm.

It is not possible to choose this option if repository-control-file RESPECT/TITLES is resident under your usercode.

- <Create a new repository>

This option creates a new empty repository.

It is not possible to choose this option if repository-control-file RESPECT/TITLES is resident under your usercode.

- <Upgrade the visible repository>

This option will load screens and messages in the visible repository and run optional conversion programs between SURE releases.

This option is only possible if repository-control-file RESPECT/TITLES is visible from your current usercode. The repository that is mentioned in the RESPECT/TITLES will be upgraded.

This option also checks whether the SURE objects can run against the repository. If this is not the case, the installation job will start DASDL compilations to update the repository. These DASDL updates will synchronize the database timestamps with the timestamps present in the SURE object files. This option cause the job to OVERRIDE the database timestamp in order to synchronize with the description file used for the SURE compilations. An off-line dump is required before this procedure is executed.

- <No repository action>

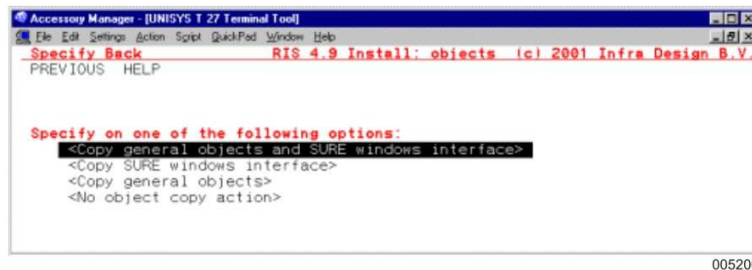
Setting this option is used to copy the SURE software. For existing installations, this may be used to reload files.

**Note:** SURE release consists of objects and screens. These screens are loaded in the repository. For this reason, using this option is not sufficient to re-install a release.

Choose "Create a new repository" if SURE was not installed earlier on your host.

### 8.2.7.2. The Second Installation Screen

Define the software that must be installed.



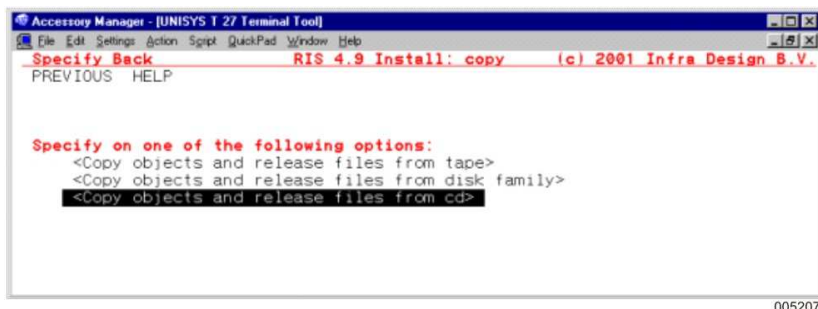
The SURE mainframe objects can be sub-divided into two groups:

- Objects in directory OBJECT/RIS/API/= form together the SURE Explorer interface on the mainframe. Each different repository requires a separate SURE Explorer interface.
- Objects in directories OBJECT/RIS/= and OBJECT/RESPECT/= (but not in OBJECT/RIS/API/=) form together the group of general objects on the mainframe. Multiple repositories can work with the same set of general objects.

Choose "copy general objects + SURE Explorer interface" if SURE was not installed earlier on your mainframe.

### 8.2.7.3. The Third Installation Screen

Define the medium from where you want to install.



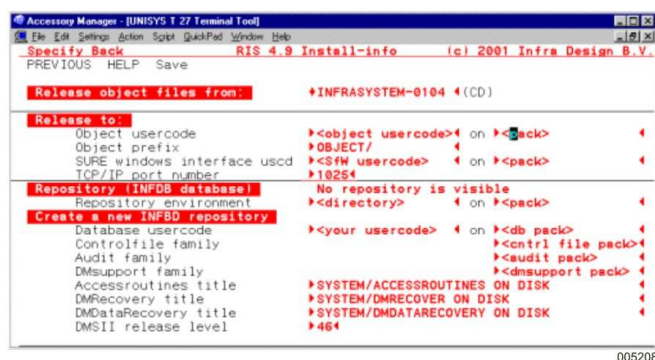
Objects files are copied if option "No object files are copied" is NOT chosen.

Release files are copied if option "No repository action" is NOT chosen.

Choose tape, disk, or CD.

### 8.2.7.4. The Fourth and Last Installation Screen

Define the usercodes and families where you want to place the software.



- Object usercode/pack

This is the location of the general SURE object files. This object-location will be stored in repository control file RESPECT/TITLES.

If you choose object-usercode = "\*", then the SURE objects will be copied to directory "\*" and they will be visible for all usercodes who are working on this family through the normal MCP visibility rules.

If you choose a specific object-usercode, for example usercode SURE, then the SURE objects will be copied to that directory. If a SURE program has to be started from another usercode, then the name of the object has to be prefixed manually with usercode SURE (example, RUN (SURE) RIS/MENU).

- Object-prefix

This is the object-prefix of the general SURE objects. By default, each general SURE object has prefix "OBJECT/". This default prefix can be replaced by a user-defined prefix, such as "SURE/501/".

- SURE Explorer interface usercode/pack

This is the location of the "SURE Explorer interface" object files (OBJECT/RIS/API/).

It is not necessary to install these objects in a global directory, because these objects are always started through COMS program definition.

- TCP/IP port number

If a new repository is installed, then a TCP/IP port number must be entered to identify a port for the SURE Explorer interface. The port may not be in use by any other application. The default value is 1025. You are free to choose any other port number if you do not agree with the default value. Notice that the port number also has to be defined at the SURE installation on the PC.

This port number is also used during the PC installation procedure. Make sure that the same port number is used for the mainframe-installation and the PC-installation.

- Repository environment usercode/pack

The location of repository control file location of repository control files RESPECT/TITLES. The installation job will create a file RESPECT/TITLES in this environment. All usercodes that have visibility to this file will route to the defined repository.

The repository environment is defined as "all usercodes that have access to repository-control-file RESPECT/TITLES through the normal MCP visibility rules."

**Note:** File RESPECT/TITLES must be visible for all usercodes that are going to access SURE. If only one usercode is going to access SURE, then this specific usercode can be entered in field "repository environment usercode"; otherwise, an "\*" must be entered which refers to the global directory "\*".

- Database INFDB usercode/pack

This is the location of database INFDB data files. If database INFDB is installed multiple times on the same host, then each INFDB must have a unique usercode. Therefore it is advised to put the database files under a usercode and not in the global directory "\*".

The usercodes of the INFDB control file, the audit-files and the dmsupport library are all equal to this database usercode.

- Controlfile family

This is the location of the INFDB-control file. (The controlfile usercode is equal to the database usercode.)

- DMsupport family

This is the location of the INFDB dmsupport lib. (The dmsupport usercode equals the database usercode.)

- Audit family  
This is the location of the INFDB audit files. (The audit file usercode equals the database usercode.)
- Accessroutines title  
The default title is SYSTEM/ACCESSROUTINES ON DISK. You can override this with your own title for accessroutines.
- DMRecovery title  
The default title is SYSTEM/DMRECOVERY ON DISK. You can override this with your own title for dmrecovery.
- DMDataRecovery title  
The default title is SYSTEM/DMDATARECOVERY ON DISK. You can override this with your own title for dmdatarecovery.
- DMSII release level  
This is the current release level of the DMSII software on the machine. Note that this may differ from the MCP release level.
- SAVE  
Specify on button "SAVE" (on the second line of the screen) if you are satisfied about the entered options. The final installation job, called WFL/RIS/INSTALL, will now be generated. In special circumstances, this job may be modified with site-specific procedures.

Specify on button "SAVE" (on the second line of the screen) if you are satisfied about the entered options.

The final installation job, called WFL/RIS/INSTALL, will now be generated.

Start the installation job, START WFL/RIS/INSTALL

This job will take about 10 minutes to complete (depending on the power of your mainframe).

The installation job performs the following actions:

- Copy all objects and database files from the tape to the correct locations.
- Compile and install database INFDB.
- Authorize the installation usercode for all SURE commands and functions.
- Give a message when the installation is completed.

Job WFL/RIS/INSTALL performs all the actions to install the software on the mainframe, except for the last step, the COMS definitions of the SURE interface.

The following files are installed:

- (<repository-environment-usercode>)RESPECT/TITLES on <your family>
- (<object-usercode>)OBJECT/RIS/= on <your family>
- (<object-usercode>)OBJECT/RESPECT/= on <your family>
- (<infdb-usercode>)DASDL/INFDB
- (<infdb-usercode>)INFDB/=
- (<infdb-usercode>)DMSUPPORT/INFDB
- (<infdb-usercode>)DESCRIPTION/INFDB
- (<infdb-usercode>)RECONSTRUCT/INFDB
- (<SURE-Explorer-interface-usercode>)OBJECT/RIS/API/=
- (<SURE-Explorer-interface-usercode>)RISAPPLICATION/TITLES/WINDOWS

Notice again the importance of repository-control-file RESPECT/TITLES:

RESPECT/TITLES is a seqdata file and you can modify or list the file through CANDE. The record in RESPECT/TITLES that starts with key INFDB-LOCATION gives the location of the database control file. The record in RESPECT/TITLES that starts with key OBJECT-LOCATION gives the location of the SURE-objects. When a program is started, the first thing it does is opening the visible respect/titles file to obtain the SURE-database location, and to obtain the SURE-object-location. The second step is to link to library RESPECT/LIBRARY in the SURE-object-location. The third step is to open the SURE-database. The SURE-object-location is also used when a SURE-object starts another SURE-object. This procedure makes it possible that SURE can be installed in a very flexible way in whatever any database-location and SURE object-location.

Wait until the installation job WFL/RIS/INSTALL is finished before the SURE Explorer interface can be defined in COMS.

Refer to "8.2.8 Definition of the SURE Explorer Interface in COMS" for more information.

### 8.2.7.5. Installation of a First Time User

The options required for a first-time installation are:

- <Create a new repository>
- <Copy general objects and SURE Explorer interface>

### 8.2.7.6. Upgrade Repository and Install All Objects

This option is used to install a new SURE release and to update one repository with correct screen messages and optional conversions. If multiple repositories are running, each of these repositories must be updated through "Upgrade a repository and install a SURE Explorer interface" (next paragraph).



The options required for this installation:

- <Upgrade the visible repository>
- <Copy general objects and SURE Explorer interface>

#### **8.2.7.7. Upgrade Repository and Install SURE Explorer Interface**

If multiple repositories are present, the first repository may have been upgraded when all objects were installed. For the upgrades of the other repositories, it is only necessary to install the SURE Explorer interface.

The options required for this installation:

- <Upgrade the visible repository>
- <Copy SURE Explorer interface>

A separate SURE Explorer interface is required for each repository. It is not necessary to re-install the general objects, because these objects can address multiple repositories on the same machine simultaneously.

#### **8.2.7.8. Installation of the SURE Objects**

This option is used to load the SURE object files without any repository update action.

The options required for this installation:

- <No repository action>
- <Copy general objects> OR <Copy general objects and SURE Explorer interface>

If the SURE Explorer interface is also installed then file RISAPPLICATION/TITLES/WINDOWS is created too.

#### **8.2.7.9. Installation of the SURE Explorer Interface**

This option is used to install the SURE Explorer interface without any further repository update actions.

The options required for this installation:

- <No repository action>
- <Copy SURE Explorer interface> OR <Copy general and SURE Explorer interface>

File RISAPPLICATION/TITLES/WINDOWS is also created.

### 8.2.7.10. Creation of an Extra Repository

This option can be used when SURE is already active on the mainframe, but for some reason an extra repository has to be created:

The options required for this installation:

- <Create a new repository>
- <Copy SURE Explorer interface>

File RISAPPLICATION/TITLES/WINDOWS is also created.

An extra SURE Explorer interface is required for this new repository. It is not necessary to re-install the general objects, because these objects can address multiple repositories on the same machine simultaneously.

### 8.2.7.11. Creation of a New Repository from Tape

This option is used to install a repository from tape INFDBTAPE-yymm. This option requires that the SURE software is already installed.

The options required for this installation:

- <Import a repository from tape>
- <No objects files are copied>

## 8.2.8. Definition of the SURE Explorer Interface in COMS

The following COMS program definition is required.

```
Accessory Manager - [UNISYS T 27 Terminal Tool]
File Edit Settings Action Script QuickPad Window Help
P - PROGRAM ACTIVITY
Action: INO
Create Modify Inquire Delete GO Home (Press SPCFY for help)
Search First Last Next Previous Dump
COMS

Program Name . . . . . INFRA_API
Task Attributes
  TITLE . . . . . OBJECT/RIS/API/DCIS ON IDRD
  USERCODE . . . . . INFRA_API
  ACCESSCODE . . . . . .
  CHARGECODE . . . . . .
  ( Additional Task Attributes Can Be Specified On The PAT Screen )
Valid Security-Category List . . . . . ALL
Database Name . . . . . NONE
Input-Queue Memory Size (words) . . . . . 1000
Minimum Copies . . . . . 1
Maximum Copies . . . . . 1
Remote-File Interface (Y/N) . . . . . N
Have COMS Perform TXOPEN (Y/N) . . . . . N
XATMI Blocking Timeout . . . . . 0
Installation Data Name . . . . . NONE

Init Time Limit . . . . .
Term Time Limit . . . . .
Queue Depth . . . . .
Remote Users . . . . .

005209
```

All programs that deal with the SURE Explorer interface start with prefix OBJECT/RIS/API/=.

It is important that all these programs are installed in the same directory.

Task attribute TITLE must be equal to the FULL name of program RIS/API/DCIS.

**Note:** The program name in the TITLE contains a usercode and a family, both equal to the usercode and family of the SURE Explorer interface on the SURE-installation screen. If no usercode is specified, then the usercode in field "USERCOD" or the global usercode is used (according to the file visibility rules on the mainframe).

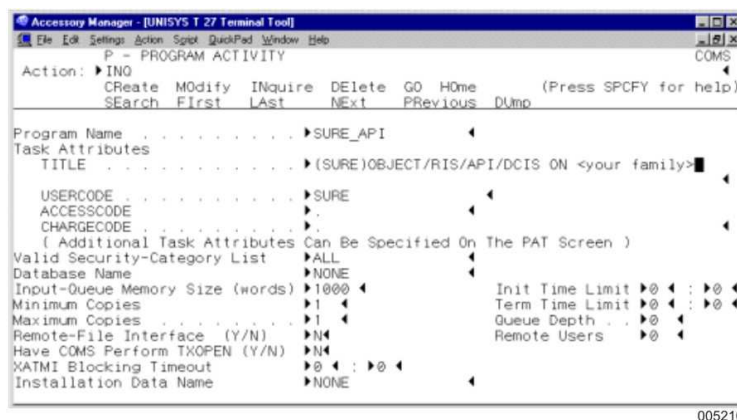
The example screen specifies a program title OBJECT/RIS/API/DCIS ON IDRD. The program will always run under usercode INFRA\_API because that usercode is entered. In this case, there are two possible directories where the SURE Explorer interface can be installed:

- (INFRA\_API)OBJECT/RIS/API/= ON IDRD
- \*OBJECT/RIS/API/= ON IDRD

The usercode defined for the COMS program (INFRA\_API in the example screen) should contain two files that are required to locate the repository and the TCP/IP port number, they are:

- RESPECT/TITLES
- RISAPPLICATION/TITLES/WINDOWS

### Example



In this second example there is only one possible directory where the OBJECT/RIS/API/= software can be installed:

(SURE)OBJECT/RIS/API/= ON <family>

### Example

The SURE Explorer interface must be started through a COMS-enable:

<mixnr COMS>SM ENABLE PROGRAM SURE\_API

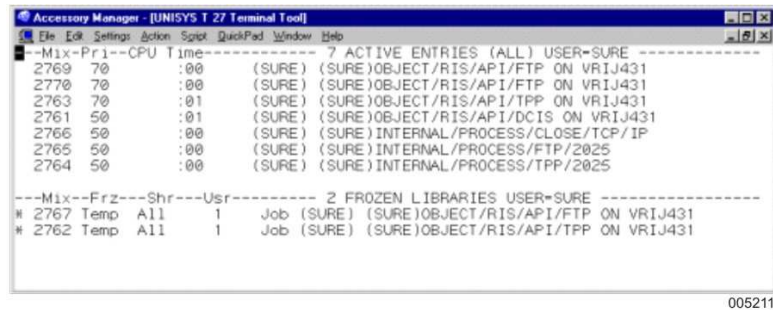
## Installation

---

The SURE Explorer interface can only be closed by a COMS-disable:

```
<mixnr COMS>SM DISABLE PROGRAM SURE_API
```

After you enabled the COMS program then at least the following objects must be active in the mix.



The screenshot shows a terminal window titled "Accessory Manager - (UNISYS T 27 Terminal Tool)". The window displays a list of active entries and frozen libraries. The active entries section shows 7 entries, all with a status of "USER-SURE". The frozen libraries section shows 2 entries, both with a status of "USER-SURE".

```
Accessory Manager - (UNISYS T 27 Terminal Tool)
File Edit Settings Action Script QuickPad Window Help
---Mix-Pri--CPU Time----- 7 ACTIVE ENTRIES (ALL) USER-SURE -----
2769 70 :00 (SURE) (SURE)OBJECT/RIS/API/FTP ON VRIJ431
2770 70 :00 (SURE) (SURE)OBJECT/RIS/API/FTP ON VRIJ431
2763 70 :01 (SURE) (SURE)OBJECT/RIS/API/TPP ON VRIJ431
2761 50 :01 (SURE) (SURE)OBJECT/RIS/API/DCIS ON VRIJ431
2766 50 :00 (SURE) (SURE)INTERNAL/PROCESS/CLOSE/TCP/IP
2765 50 :00 (SURE) (SURE)INTERNAL/PROCESS/FTP/2025
2764 50 :00 (SURE) (SURE)INTERNAL/PROCESS/TPP/2025
---Mix--Frz---Shr---Usr----- 2 FROZEN LIBRARIES USER-SURE -----
# 2767 Temp All 1 Job (SURE) (SURE)OBJECT/RIS/API/FTP ON VRIJ431
# 2762 Temp All 1 Job (SURE) (SURE)OBJECT/RIS/API/TPP ON VRIJ431
```

005211

The installation of the mainframe software is now complete.

## Section 9

# Configuration and Sizing

In the following pages, we shall introduce you to the main features of SURE through an explanation of the SURE configuration options. This document describes the theory and procedures behind the SURE system.

Before you start working your way through this document, we strongly advise you to familiarize yourself first with the purpose and terminology of SURE. You will find these explanations in the next section of this tutorial. It may also be helpful to read the online help files and release notes at <http://www.itsforsure.nl/downloads>.

### 9.1. (Configuration) Terminology

#### **Software Configuration Management (SCM)**

SCM is also known as Application Life-cycle Management and is best described in the following bullets:

- Storage of all source files and source related information in a central repository.
- Streamlined and secure separate environments in which all distinct activities of the application life-cycle take place.
- Application of rules and procedures, supported by automated tools through which these rules and procedures are implemented and controlled, and through which the impact of changes on the application life-cycle can be monitored.

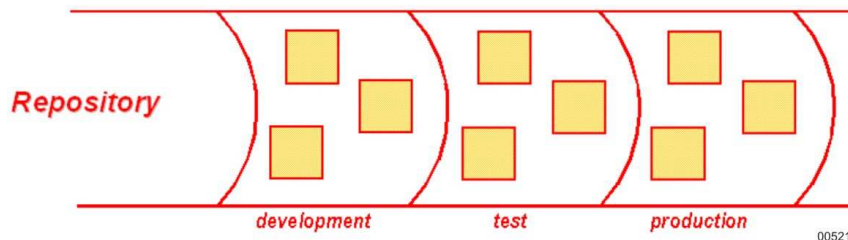
In other words, SCM is an integrated set of tools and procedures that manage or control the software development and the deployment process.

#### **Repository**

The repository is a DMSII database on Unisys ClearPath Enterprise Servers, into which all the source files, delta files containing changes to these source files and other source related information of the various applications an organization has, are loaded and safely stored. The DMSII database provides the advantages of keeping all this information in a single data structure and compresses this information to create a highly efficient storage. The fact that a database is used for source storage also has the advantage of applying proper dump procedures.

### Environment

The repository is divided into separate sections, called environments, which match the various phases in the life-cycle support for a particular organization. Typical examples, which are often recognized as separate phases within an organization are "development," "test," and "production."



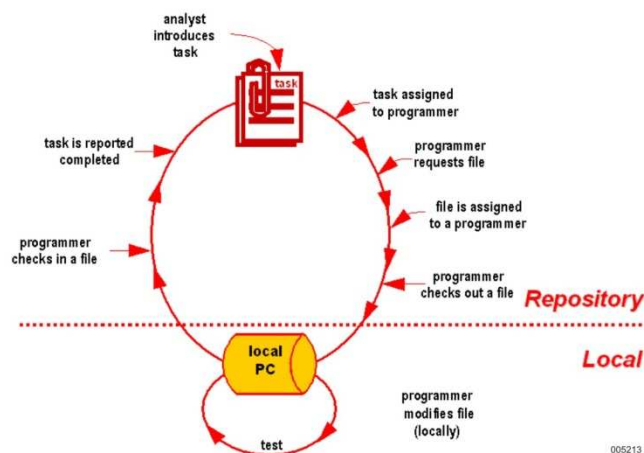
The above given structure may be adjusted to reflect an organization's own way of working.

### Files

For each application system within an organization, all source files are stored in these distinct environments. SURE allows storage of source files for ClearPath Enterprise Servers (regardless of the file-kind), as well as storage of any kind of PC file. Once stored into the repository, files may be retrieved from the repository through a check-out procedure and after editing, compiling and programmer-testing, stored back again through the check-in procedure.

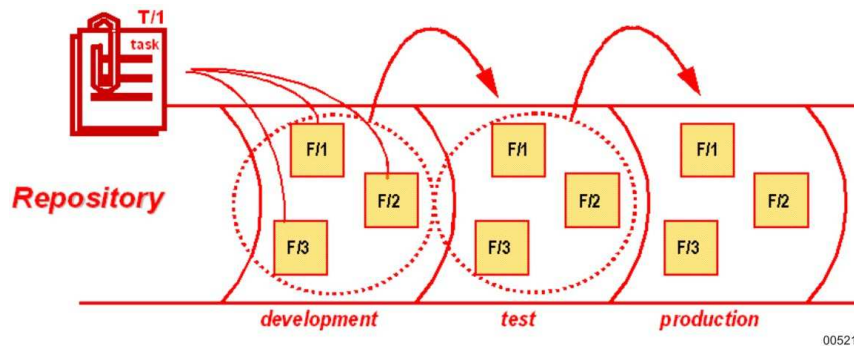
### Task

A task is a logical unit of work, and describes the reason of making changes to existing source files or creating new files. The task may apply to a required fix of an error, or to the development of a new feature. SURE links the task automatically to the source files that are modified because of that task. A developer can only check out a source if the task is assigned to him or her. Management assigns this task to one or more developers. Every time a developer starts his or her work, he or she first "subscribes" to the task that is assigned to him. From then on, all the work the developer does in the context of the task will be linked to that task.



## Transfer of Tasks

The task is the focus point for SURE to keep track of the progress of work. This means that SURE does not allow the transfer (or promotion) of a single source file to another environment. This is always done through the task mechanism. Since a task may involve changes to multiple source files, the transfer of the task ensures that all the changed source files are transferred as one unit of work, for example, from development to test and, after acceptance testing from test to production.



## Quick Fixes

When an error occurs in the production version of a program, the fix for it must be applied without disturbing the work that is already taking place in the development or test environments. SURE offers for this the quick fix feature: the error is quick fixed in the current production version of the source, and then applied to the newer versions of the application that are currently prepared in development and in test. The quick fix facility avoids the re-occurrence of the problem in the future.

If the source where the error must be fixed is not in maintenance in the development or test environments, then the regular development (check-in/modify/check-out/transfer) procedure can be used.

## Impact Analysis

A key-feature of SURE is the facility of impact analysis. With impact analysis it is possible to determine whether a particular modification in the changed source impacts other source files. A good example of this may be a modification in the source of a single copy file that is used in a number of programs. SURE knows where this copy file is used, and SURE will automatically compile all relevant program sources that make use of the changed copy file. The facility of impact analysis of SURE is continuously present throughout all phases of the development and deployment process, and ensures the integrity of the application systems.

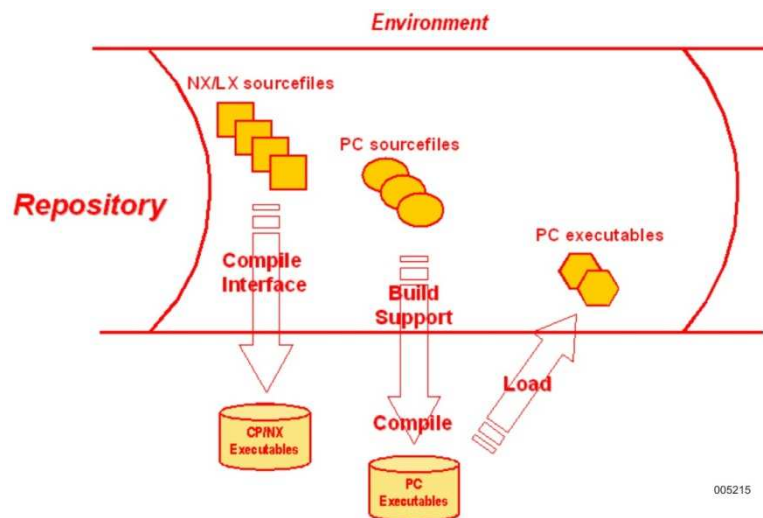
## Windows Workbench for Development for ClearPath Libra Servers

SURE comes with a fully featured development workbench on the Windows platform that works seamlessly with the server-part of SURE running on the ClearPath server. The SURE workbench can be integrated with editors, compilers, and any other PC tool from other vendors. This allows developers to download the source files that need to be maintained from the server to a local PC, do all the modifications locally, using their own favorite editor, do even a pre-compile with a local compiler on the PC and upload a syntax-free source file back to the server where it is stored in the repository again.

### Build Support

SURE supports the automatic compilation of source files of applications for ClearPath servers, such as the compilers for widely used programming languages as COBOL, ALGOL, DASDL and WFL, but also less commonly used compilers such as Pascal, Fortran77 and C++.

SURE provides build support for ClearPath servers as well as for the Windows platform. On the Unisys platform, this is achieved in a direct way, by compiling the sources directly for proper development phases for which a separate environment is kept. Through the task mechanism of SURE supported by its impact analysis facility, the compile process is managed automatically. The resulting object files are kept outside of the repository; however, relevant information of the object files (example, versioning) is stored.

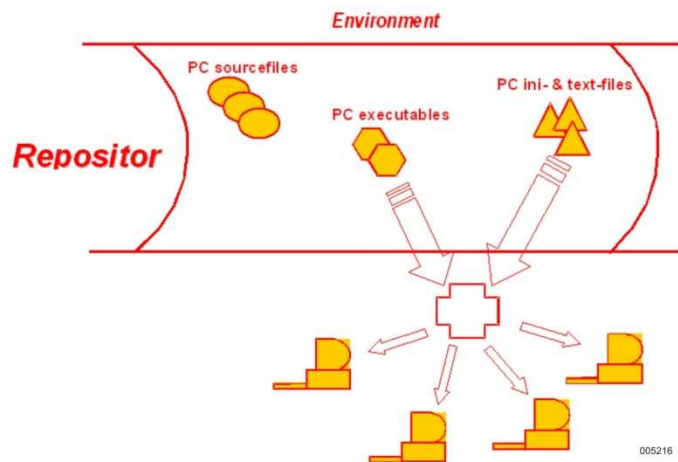


For the PC source files SURE supports the building of the necessary files for the compilation process that takes place on the Windows platform. After compilation, the resulting executables are loaded into the SURE repository as well.



### Distribution Support

The distribution support for ClearPath servers allows copying object files to the defined server machines using BNA or tape transfer. The distribution support in the Windows environment consists of preparing a directory that needs to be distributed. A distribution tool like Microsoft SMS or a tool like RoboCopy performs the actual distribution.



## 9.2. (Configuration) Quick Overview to SURE

Before we start to have a closer look at SURE, we shall first give a short overview in broad terms of SURE and its architecture.

SURE provides a comprehensive tool-set for SCM that makes it possible for an organization to manage and control all phases of the development and deployment process of applications.

SURE manages the repository on Unisys ClearPath servers, in which all source files of the organization's applications are stored.

SURE keeps a constant eye on the changes made to the contents of the repository and takes automatic actions accordingly, such as (multi-)compiles of the changed source files, transfer of new versions of an application from development or test to production and so on.

SURE keeps a strict separation between work that is done during the actual development or maintenance processes and work, which is required during production, such as error fixing. To this extent, SURE maintains the environments, in which distinct versions of entire applications are stored.

All activities during development and maintenance are carried out under the umbrella of the task mechanism in SURE. This adds significant value to the development and maintenance process in terms of completeness of work and integrity of the applications.

SURE consists of two parts, one part running on the ClearPath Enterprise Servers that effectively manages all aspects of the repository, does the impact analysis and provides compiler support. The other part of SURE, being the client-part, is a fully featured development workbench on the Windows platform that works seamlessly with the server-part.

SURE provides an organization with the means to control all of its applications, whether these exist on the Unisys platform or on the Windows platform, in an integrated manner, thus ensuring the quality and integrity required to support today's e-business requirements.

SURE can work in three modes:

- Source-maintenance  
The stable version of each source is stored in the SURE repository. A developer uses function "check-out" if he wants to modify a source and function "check-in" if the modifications are done. SURE keeps a physical history of each source.
- Source-maintenance + tasks and environments  
All functions of "source-maintenance" plus an integrated task tracking mechanism. A task describes the reason why a source has to be modified, and this task is linked to the source when that source is checked out. The (source-modifications because of a) task can be tested in different environments (TEST, ACCEPTANCE, and so on). SURE keeps also a logical history of each source.
- Source-maintenance + tasks and environments + application deployment  
All functions of "source-maintenance + tasks and environments" plus an integrated application deployment mechanism. Files that are checked in to SURE are automatically compiled (during an evening batch). If a copy file is changed, then the calling programs are compiled. The location of the compiled object can be defined in SURE. SURE copies the compiled objects to the correct object-locations. SURE guards the integrity of the total object environment.

### 9.3. (Configuration) SURE Configuration

SURE is a source management system, as well as a system that provides software build, configuration and distribution support. Because of the support for these integrated software life-cycle functions, the SURE system allows for simple configurations with just check-in and check-out support up to complex configurations, which include support for the creation of executables and distribution of software. This document guides you through the configuration options and it explains the various supported procedures.

The SURE system supports Unisys ClearPath server files, Windows-32-bit files and UNIX files, if they are shared from a Windows system. This document applies to all of these mentioned types of files unless it is explicitly stated otherwise.

The SURE system contains of two different parts, a server part implemented on ClearPath server files, which is used as a storage server. This server part consists of a DMSII database with a software layer around it, which let the server part behave as a repository. This DMSII database will contain all of your definitions and sources so that

a simple database dumps procedure can be used for archive purposes. The second part of the SURE system consists of a 32-bit Windows client software package, which provides a user interface similar to Windows Explorer.

Server and client part are connected through native TCP/IP, which allows you to run over a LAN or over the Internet.

The SURE system allows for a dynamic configuration. This means that the SURE system may be tailored to the specific procedures within an organization. Because the SURE system is flexible in its configuration, it requires that the system must be configured.

Configuring the system is not a difficult task if you are familiar with the capabilities of SURE. However at first installation you are confronted with a new tool, SURE in this case and you need to make decisions which require understanding of the capabilities of this tool in relation to the procedures currently applied within your organization.

Another complicating factor is that you might want to change currently applied procedures because SURE offers other capabilities. It is obvious that some of your current procedures must be changed when you are going to use SURE (example, your current procedure "assign a source to a developer" will change because that source is now loaded in SURE). Others of your current procedures can stay in place next to SURE.

SURE is capable to control up to eight different environments (design, develop, test, integration, production, history), each with their own set of objects in a separate run-time environment. Run-information about the objects can be stored into SURE. SURE has a flexible and powerful authorization mechanism that can limit the programmer's activities. SURE can address various modes of working, such as shared workspaces, individual workspaces, and baselines.

However, you do not have to use all functions of SURE, and you do not have to activate all desired functions at the beginning when SURE is installed.

From our viewpoint, we would like to stress that a new tool and new procedures introduce an additional risk and this may decrease the acceptance level within an organization. For this reason we advise to change procedures gradually afterwards if that is possible. In this way the users are used to the tool SURE when a procedure changes.

The SURE system allows you to define rigid procedures or it allows you to define flexible procedures. For example, SURE allows you to define that a programmer first requests a file, after which the project leader releases the file for this programmer allowing the programmer to check out the file. This example of a rather rigid procedure is fully supported by SURE with the appropriate commands and workflow queues. However SURE also supports a procedure where the programmer directly checks out the file for a specific task. This is a more flexible procedure, but it requires programmers that are more experienced.

### 9.4. (Configuration) Goal

This section contains background information, which allows you to configure a repository, load files in this repository and finally perform lifecycle support on these files.

### 9.5. (Configuration) Functionality

SURE is a task-based system, which requires defining a task for a functional change to an application system. All the functional changes for this task, to different files that may reside on different platforms, are connected to this task. This task or logical unit of work is taken into acceptance or into production. SURE supports this procedure by workflow queues, task distribution over different employee functions and support for the build and distribution process.

A task is a logical unit of work that matches a functional requirement. SURE offers functionality to group tasks together, so that those tasks are released at the same time. A task can also be used as a release task: all the changes for a specific release, carried out by different programmers, are grouped under this task and deployed as a new release.

SURE supports the implementation phase of software construction. This support deals mainly with source maintenance, which is referred to as life-cycle support. This life-cycle support deals with the simple straightforward procedures like check-in and check-out and a role based security mechanism. On the other hand SURE offers work flow queues for the various employee roles such as a To Do list for a programmer or a To Be Accepted list for quality assurance employees or a Ready and Busy list for a project leader.

These lists are present in every fixed stage into your source life-cycle. Within SURE, such a fixed stage into your source life cycle is called an environment. Furthermore, these lists may be distributed over employee roles and filtered for specific application systems or sub-systems.

Within the life-cycle support functions, the quick fix procedure and the impact analysis are main sub functions.

The quick fix procedure allows you to make fixes to your production software without disturbing the development process. Even if the file you are fixing is currently in maintenance by another programmer. Firstly, the SURE software will prohibit accidental overwriting of this fix, so the error cannot appear again in a new release by accident. Secondly, the SURE software offers automated functionality to incorporate the fixes into the standard development process.

The impact analysis allows you to detect whether different functional changes resulted in modification of the same file and it may provide you with a compile impact, a list of programs that need to be compiled for a change. This compile impact includes manipulation of copy or include-files. If a copy or include file is changed, SURE will recompile all the programs using this changed copy or include file.

Compilation and build support is an optional feature that is supported by SURE. This feature creates the executables from ClearPath server sources as well as for PC applications. For ClearPath server software, the compilation support consists of a program that compiles all the changed files with the appropriate compiler. The repository contains definitions for file security, file attributes, task attributes, binding and post compilation requirements. For ClearPath server files, the resulting object is given a unique release id, which matches the current file version of the compiled source. An especially useful command for ClearPath servers is "recompilation of sources using a specific dataset." Reorganizing a database always requires such a procedure, and this procedure is smoothly integrated in the SURE software. For PC and UNIX files, the repository contains the build commands. These build commands can be seen as a .BAT file which contains the appropriate commands that are required to create the executable. The build support for PC and UNIX files downloads all the changed files to a server. Secondly, it executes the build commands each file or each directory and in the final stage it loads the resulting executable into the repository.

Distribution support is an optional feature that is supported by SURE. Distribution support actually copies the files to the target destination. For ClearPath server files, this may be a user code and pack family on the same host or on another BNA host. Tape transfer is also supported for ClearPath server systems. For PC and UNIX files, a list of files is maintained of the actual files that need to be distributed. This list can be downloaded in a directory. The SURE system supports creation of this directory. Other sub systems must be used to perform the actual distribution. An example distribution system is Microsoft SMS.

The minimum configuration for SURE is using a simple form of life-cycle support. Using SURE in this fashion return direct benefit like, who changed what, where, and when. Secondly, you can view the file changes and rebuild any previous version. In this mode, a programmer can reuse his task repeatedly.

The next stage in using SURE consists of defining your stages in a source life-cycle. These stages (environments) will allow you to use most of the life-cycle support functions including workflow queues.

The final stage in using SURE implies using build and distribution support. Hereafter, you can run your system with only a single person controlling it. If the SURE build and distribution support functions are used, your total operation is automated and no manual actions are required.

## **9.6. (Configuration) File Entities**

- Environment
- System
- Project or sub-system
- File Type
- File

All definitions for files are assembled using the previously mentioned entities.

- An environment is a fixed version in the life-cycle of a file. For example, the test-versions of all files form together the test-environment.
- A system is a set of files grouped together based on physical coherence, so all files using the same database might belong to a system. At system level, different physical attributes are defined which apply to all the files of that system. A system always contains at least one project (sub-system), a project with the same name as the system name. On the other hand, a system may contain multiple different other sub-systems (projects).
- The word “project” can have two meanings:
  - A sub-system.
  - An amount of work to be done.

In the SURE context, the word “project” means “sub-system.”

“An amount of work to be done” is in the SURE context “a task.”

A project is mostly used for security reasons. Persons may be granted to perform functions on files of one or multiple projects. From a logical viewpoint, projects are a collection of files that are grouped functionally. The system level is mandatory within the SURE system; however, a detailed project level is obsolete in which case the system name equals the project name.

- A file type categorizes the files for SURE so that the same definition may apply to a group of files. A file type is a rather abstract definition that contains attributes such as:
  - A file of this category creates delta files.
  - A file of this category needs to be compiled.

Therefore, a file type categorizes files in such a way that SURE performs certain actions on those files. Notice the difference with the contents dependent file-kind attribute such as ALGOLSYMBOL or C files. Both of these files can be split over two different SURE file types, the first describing that it is an include file (ClearPath server) or a header file (PC and UNIX) and the second describing that it is a source which needs to be compiled.

### 9.6.1. (Configuration) Environment

A key entity in the SURE system is an environment, which can be considered as a fixed phase in a development life-cycle.

An environment is a fixed phase in a development cycle. Common names for environments are DEVELOPMENT, ACCEPTANCE, or PRODUCTION.

In this example:

- DEVELOPMENT defines the phase where programmers adapt files for implementation of new functionality. Module testing by a programmer is included in this phase.

- ACCEPTANCE is a phase where quality control performs integrated test scripts on the implemented functions. Therefore, the quality control accepts functional changes and marks them as ready for production.
- PRODUCTION is the phase where the actual software runs on the various production systems.

Another company may have the same phases in their development as the previous example; however, they require a hardware integration test. This company runs the application system on different hardware using different compilers and they require a final test on the production hardware before it is actually taken into production. This company calls this environment BETA\_SITE. Therefore, the environments for this company are called DEVELOPMENT, ACCEPTANCE, BETA\_SITE and PRODUCTION.

Above given example defines a software life-cycle often used in companies writing their internal software. Mostly, these companies do not support official releases of application packages; however, this is a default requirement by software manufacturers. The previous example shows a production environment that contains all the files required to support the production application systems of the company.

If we would consider a software manufacturer, then the environments would be used more dynamic. Again, one could support DEVELOPMENT and ACCEPTANCE as their lowest developments environments. The development process and quality control may be performed in the same sequential flow. However, after acceptance multiple different release environments may be present. So after the environment acceptance a RELEASE\_5.0 environment may be present and thereafter RELEASE\_4.0 and thereafter RELEASE\_3.0. A software manufacturer often supports multiple different releases because each release may still be in use by one of his customers for which he agreed a support contract. A customer running software release 3.0 may require patches that must be created through software updates on release 3.0. Another customer running on release 4.0 requires updates for that release.

The software manufacturer has the two lowest environments (development and acceptance) for life-cycle purposes. The number of environments after acceptance and their names may vary. If an old release is not supported any more, then the environment is deleted. If a new release is created, then a new environment for that release is created. Therefore, when release 6.0 is created, a new environment is created called RELEASE\_6.0 directly above the acceptance environment. The content of this RELEASE\_6.0 environment is copied upwards from the acceptance environment.

Now let us have a more detailed view on the contents of such an environment. First, each environment contains all sources. Each environment has a corresponding compile and builds process and that process uses the sources from that environment as input to create the object files and executable files. Additional definitions that are used to create the executables (such as task attributes, file attributes, translate tables and build commands) are stored in the repository. The repository also contains definitions that support the distribution process (object-locations).

As described, an environment contains all the sources; however, for PC files it will also contain the resulting executable. This is not true for ClearPath server files where the executables are not loaded in the repository.

For maintenance reasons the definitions for build and distribution support are usually not defined at individual file level. There are three entities that contain most of the build and distribution definitions: file type, system, and project.

An environment contains all files. Each environment can contain another version of a specific file, but a specific file version can also be in use in multiple environments. The files are grouped in application systems and a release notion can be defined for a system. Therefore, an environment contains multiple systems and a system contains a set of files for a defined release. An environment contains a release for each defined application-system.

### 9.6.2. (Configuration) System

System is the most important configuration entity.

A system is a set of files grouped together because they share the same physical attributes. In the SURE software, this applies to the attributes work-location and object-location. These work and object characteristics are defined each environment.

#### 9.6.2.1. (Configuration) Work-Location and Object-Location

The work-location is only applicable for ClearPath server files and it designates the directory (usercode and the pack name) where the shared resources are located. With shared resources, we mean copy files or include files, a database description file, and so on.

The work-location is used for two purposes:

- The work-location defines the shared resources directory.  
Shared and stable resources (copy files, description file) are used by programmers when they do a test compilation of a source that they have in maintenance. It is obvious that the version of a copy file can change each SURE environment. The programmer expects that the correct version of each copy file is available on disk in the shared resources directory; otherwise, he cannot compile his source correctly.
- The work-location determines the programmer's work directory.  
The programmer has to be logged on to this directory when he wants to work on a task (updating sources that are loaded in SURE). The sources-in-maintenance are placed in this directory.

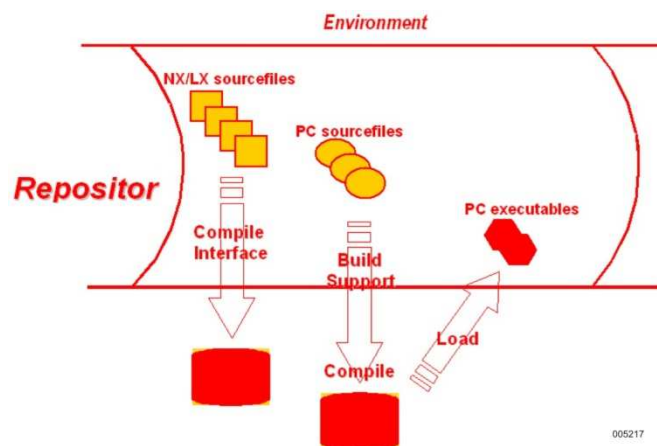
If a file has an object-location then it can be deployed. The object-location of a ClearPath server file determines the destination directory (usercode, pack, and host) for the resulting object after the source is compiled by the SURE batch. The object-location of a PC and UNIX file determines the logical server name where the build process is done.

The object-location of a system is the default object-location. When a new file is added for that system, then that file inherits the default object-location.



The object-location is applicable for both ClearPath server files and PC and UNIX files, but the implementation for both kinds is different.

- For ClearPath server files the object-location designates the default directory (usercode, pack name, and optionally the host name) for the resulting objects after they have been compiled by the SURE build support. In other words the object destination for the distribution support.
- For PC and UNIX files, the object-location is a logical server name, which is assigned as the build server. In other words, the executables for this system are built on the assigned server. Therefore, for PC and UNIX files, the object-location defines the server where the actual build process is executed. The build process copies all the changed files (with that build server) to the configured directories and starts the build commands. Therefore, you can just define the same build server for every environment if you configured different directories each environment for this server.



## Summary

The work-location of a system affects the programmer's way of working because it defines the directory where the shared resources are placed, and it determines the directory where the developer must be logged-on to be able to do his job. The object-location of a system is only a default value for the object-location of a file and it does not affect the programmers' way of working.

From the previous description, you might conclude that a system is merely a technical separation. However, in most circumstances a system is defined as an application system such as a mortgage system, a credit card system, a stock exchange system or a general ledger system. This is because all of the files within such a system normally share the same work and object-locations.

Refer to Section 15, "Copy Files," for more information on all the different options that influence the work-location, including a rather complicated option called "resources."

A SURE repository is sub-divided into multiple environments (for example, develop, test, and production). A file is modified in the develop environment, and transferred from develop to test and production. If a file is transferred to production then it is

equal in all environments, until somebody makes another modification to that file in the develop environment.

**Note:** A stable file is equal in all environments, and therefore it is important that all usercodes and packnames are removed from the sources that are loaded in SURE (because production usercodes/packnames are usually not the same as test or develop usercodes/packnames).

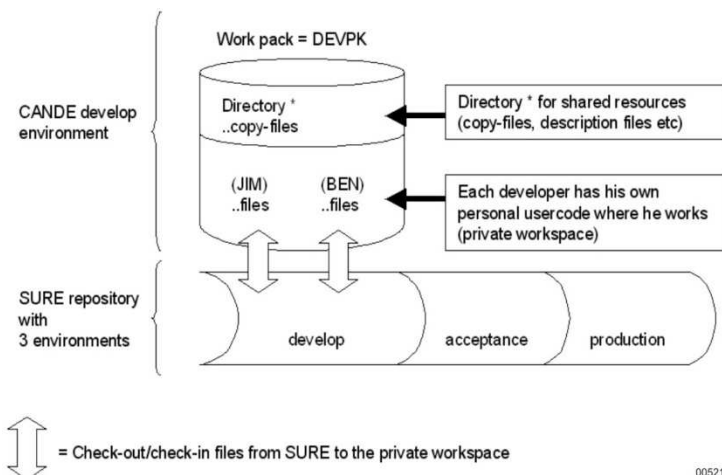
The following paragraphs explain some basic configurations on which you have to decide. These configurations depend basically on the definition of the work-location.

**Note:** The work-location is defined each system, and that makes it possible to use a different configuration for each application system.

### 9.6.2.2. (Configuration) Private Workspace/Shared Resources

#### Characteristics

- Each developer has his own personal usercode to log on.
- Each developer has his own personal workspace where he does his work (editing, compiling, and testing). This personal workspace is identified by his personal log-on usercode.
- The shared resources are placed in the global directory \* (without a user code) on the work pack.



The sources do not contain any usercodes and packnames.

A programmer logs on with his personal usercode and checks a file out of SURE. The source is placed on disk under his personal usercode. The copy files that are used by the source must be resident on disk; otherwise, it is not possible to compile that source.

The copy files are resident in the shared resource directory (\* on the work-pack) and these files are available for the developer through the standard visibility rules of CANDE, if a copy file is not resident under the developer's personal usercode, then that copy file is found in the shared resource directory.

If a copy file is checked out from SURE, then that copy file is also placed on disk under the developer's personal usercode, where the developer modifies that copy source. The modified version of the copy file is only visible for the developer who checked the file out. The other developers (who are logged on with other usercodes) still use the stable version of the copy file from the global directory \*.

If the new version of the copy file is checked in to SURE, then it is removed from the developer's personal workspace and copied to the shared resource directory (\* on work-pack) where it is visible for the other developers.

An advantage of this option is that other users are only aware of the modification of a copy file after that copy file is checked in. A developer changes a copy file in his own private workspace, and the test results of other developers are not affected by the changed copy file.

Mode "Private workspace with shared resources" can be defined with the following system options:

- The "shared resources" functionality is established by option "Put changed copy files in work-environment."
- The "private workspace" functionality is established by:  
Work-location-usercode = \*

### Example

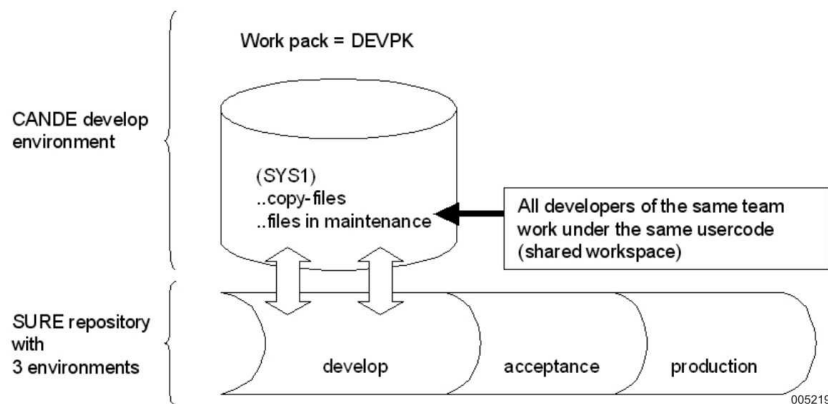
Consider copy file SYS1/COPY/001.

- Version 3.1 of that copy file is active in all SURE environments and that version is also available in the shared resource directory: \*SYS1/COPY/001 ON DEVPK.
- Programmer Jim does a check out of the copy file and the source is copied from SURE to his private workspace: (JIM)SYS1/COPY/001 ON DEVPK.
- Jim modified the copy file in his private workspace. Test compilation started by Jim use the copy file in his private workspace, because the source that references the copy file does not contain usercodes/packnames in the copy statements.
- Programmer Ben uses all the time the stable version 3.1 of the copy file.
- Jim finished his work on the copy file and does a check in. The new version of the copy file is loaded in SURE with version number 4.1; the copy file is removed from Jim's private workspace and copied to the shared resource directory.
- From now on version 4.1 of the copy file is used by all developers.

### 9.6.2.3.(Configuration) Shared Workspace/Shared Resources

#### Characteristics

- All developers modify their sources in the same workspace (usercode/pack).
- The source files and copy files are maintained under that same usercode.
- The shared resource files (copy files, description files) are placed on disk under that same usercode.



The main differences between this mode and the previous mode (private workspace with shared resources) are:

- The user does not have an individual work-environment. So, during the modification of files, every other user may be affected during the module test.
- All copy files are always available in the shared workspace. That makes it possible to change a copy file without officially checking it out from SURE. This situation is not critical for the production environment because such a file cannot be checked in, but it may cause temporary questions (who changed his source and why did he do that?).

Each developer still logs on to SURE with his own personal usercode. This log-on usercode is only used for identification purposes. If the developer checks a file out, then that file is placed in the shared workspace.

Mode "Shared workspace with shared resources" can be defined with the following system options:

- The "shared resources" functionality is established by option "Put changed copy files in work-environment."
- The "shared workspace" functionality is established by: Work-location-usercode = <usercode>.

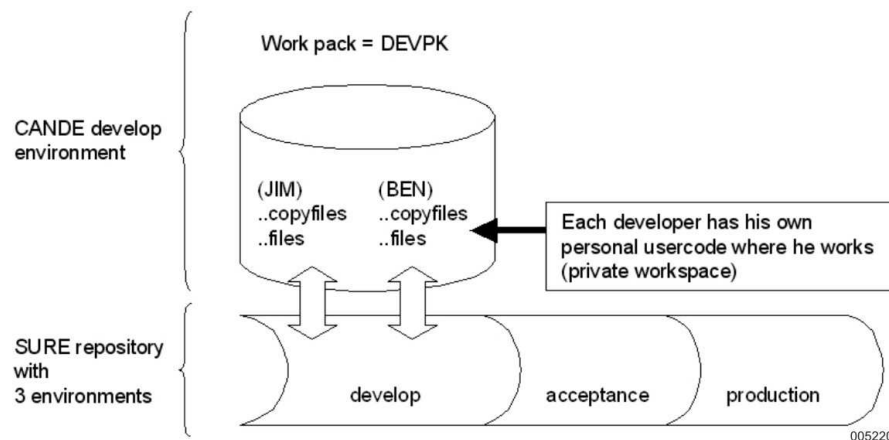
**Example**

Consider the shared workspace (SYS1) ON DEVPK:

- User Jim logs on to SURE with his personal usercode JIM.
- He checks a file out of SURE, and the file is placed on disk as (SYS1)SYS1/PROG/001 ON DEVPK.
- He checks a copy file out of SURE. The copy file is placed on disk as (SYS1)SYS1/COPY001 ON DEVPK.
- The source and copy file are now visible for all other team members that work on system SYS1.

**9.6.2.4.(Configuration) Private Workspace/Private Resources****Characteristics**

- Each developer has his own personal usercode to log on.
- Each developer has his own personal workspace where he does his work (editing, compiling, and testing). This personal workspace is identified by his personal log-on usercode.



A checked out source is placed in the private workspace of the developer. All copy files that are referenced by the source are also copied in the same private workspace. When the source is checked in then the copy files are checked and cleaned-up (copy files that are not referenced by other checked out sources are removed from the private workspace).

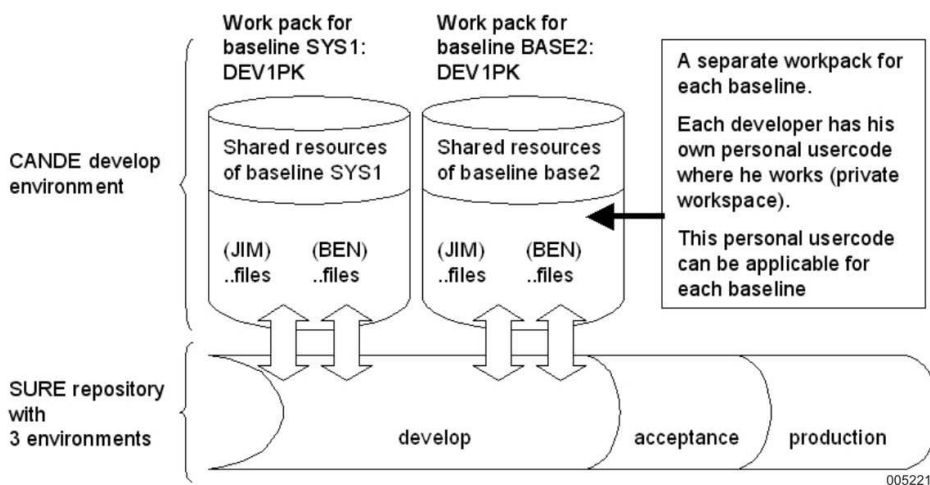
This mode is defined with the following system options:

- The "private resources" functionality is established by resetting option "Put changed copy files in work-environment"
- The "private workspace" functionality is established by:  
Work-location-usercode = \*

### 9.6.2.5.(Configuration) Multiple Baselines

#### Characteristics

- Each developer has his own personal usercode to log on.
- Each developer has his own personal workspace where he does his work (editing, compiling, and testing). This personal workspace is identified by his personal log-on usercode.
- The total development staff that works on the same application system is subdivided in several development teams.
- Each team works on a separate big task (also known as a baseline).
- Modifications that are made for a specific baseline may not be visible for other baselines unless the project leader of the other baseline explicitly accepts these modifications.



Each baseline is linked to a separated workpack.

If option "put changed copy files in the work-environment" is set for a specific baseline, then that baseline works with shared resources.

If option "put changed copy files in the work-environment" is reset for a specific baseline, then that baseline works with private resources.

If Work-location-usercode = \* for a specific baseline, then each developer of that baseline has a private workspace.

If Work-location-usercode = <usercode> for a specific baseline, then all developers of that baseline work under the same shared workspace.

A patch (modification) to a source is done within a baseline, and the name of the baseline is linked to that patch. Patches of a specific baseline are not visible (in copy files or in compiled objects) for the other baselines, unless the project leader of another baseline explicitly accepted the patch.

The source can be checked out for two different baselines at the same time.

The patches of all baselines are all merged into the source in the first environment where the baselines are not defined.

### **Example**

The objects on pack DEV1PK are created from the development environment in SURE and contain only the patches of baseline SYS1.

The objects on pack DEV2PK are created from the development environment in SURE and contain only the patches of baseline BASE2.

The objects that are created from the acceptance environment in SURE contain the patches of both baselines SYS1 and BASE2.

Extra baselines are defined through system option "Is baseline of."

### **Example**

Consider system SYS1 and an extra baseline for that system BASE2.

BASE2 is an extra baseline of SYS1; BASE2 is baseline of SYS1 with workpack DEV2PK

SYS1 must be a baseline of itself; SYS1 is baseline of SYS1 with workpack DEV1PK.

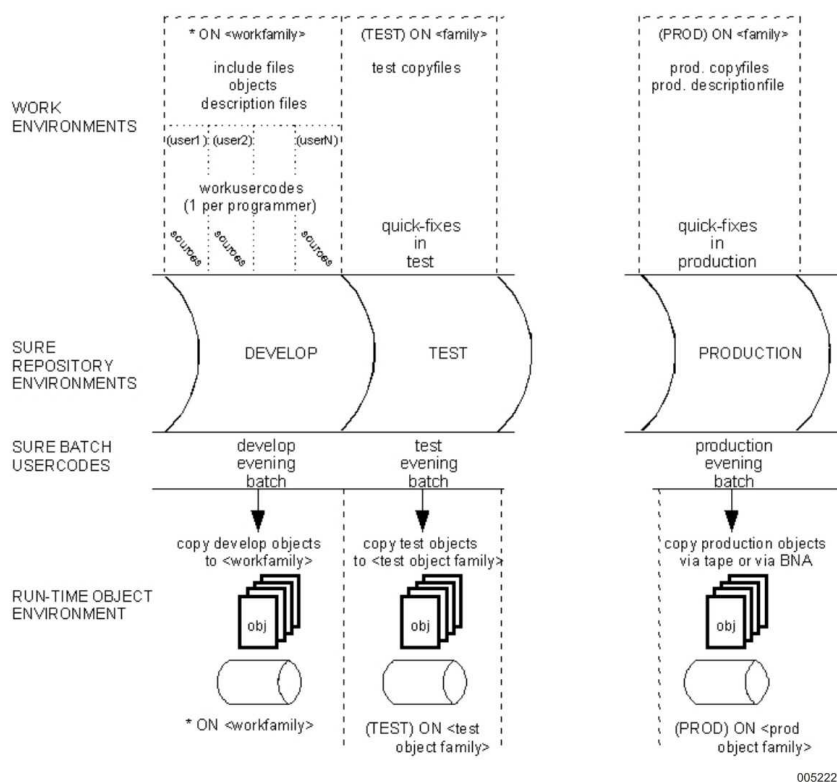
## **9.6.2.6.(Configuration) Hybrid Configuration**

Many complex configurations are grown from using own tools and procedures. This may imply generation of jobs with different object names for test and production, replace utilities that replace the copy or include statements for test and production, and so on.

The work-environment and object environment are used for the build support, distribution support and the workspace of the programmer. You can always use SURE for check-in and check-out purposes in combination with your current procedures. In this mode, you do not use the automated mode for build and distribution support and you only use part of the life-cycle support. However, you still can use many other SURE features like the auditing, log, delta files, PC editing, and impact analysis. Using this start, you could convert different systems gradually if required.

## **9.6.2.7.(Configuration) Summary**

The following picture shows a develop environment with shared resources and private workspaces, and test and production environments with shared resources and a shared workspace.



The following options are applicable in the discussion of workspaces:

- File Type Use as Copy file (Refer to Section 33, "File Type," for more information)

Files that are actually referenced by other programs as a copy file behave automatically as a copy file (and will be placed in the shared resource directory if option "put copy files in the work-directory" is set).

It is possible to link this copy file behavior to other files through file type option "use as copy file."

- System- Put Copy files in the work directory (See "(Copy Files) <Put Copy files in the work directory>" in Section 15)

This option puts each copy file and each file that behaves like a copy file in the shared resource directory after it has been changed.

- System- Work-environment (See "(Copy Files) Work-Environment" in Section 15)

This option defines the shared resource directory

Workspace	Resources	Put Copy in Work	Work User Code
Private	Shared	Y	*
Shared	Shared	Y	<usercode>
Private	Private	N	*



### 9.6.3. (Configuration) Project

A project is a sub-system.

The project definition within the SURE system is optional. When you have created a system, a project with the same name as the system name is implicitly defined within the SURE system. So effectively, you can have an installation without specific projects defined. From a hierarchical point of view, a system can have multiple projects and a project can have multiple files. However, a file or a task can only belong to a single project.

#### Example

Consider system SYS1 with project PRJ1 and system SYS2 without specific projects.

A system is always a project of itself, so the following system and project combinations are available.

System	Project
SYS1	SYS1
SYS1	PRJ1
SYS2	SYS2

If a file is added for project PRJ1 then it is automatically linked to system SYS1.

If a file is added for project SYS2 then it is automatically linked to system SYS2.

It is not necessary to define specific projects. Each system is always available as project, and if the system and project are always the same then there is no need for extra projects. On the other hand, a project may be used just to define new tasks. There is an automatic naming standard where the name of a new task is based on the project name. This makes it possible to identify a task by its names as being a task of a certain project.

The major functions that apply to projects are:

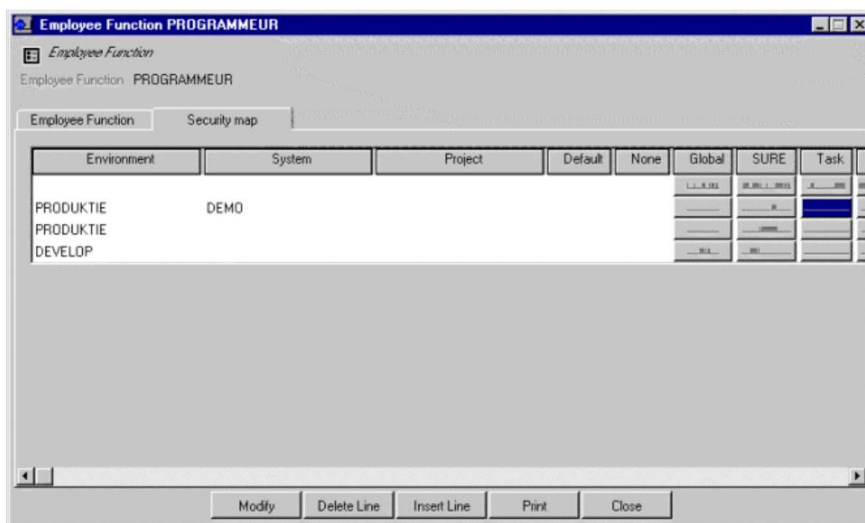
- Security (Refer to Section 34, "Authorization Mechanism and Logon," for more information )

The security map is defined each employee function. This is commonly seen as "role based security." The granularity of a security map can differ. It can be global and then it applies to all environments and to all projects (who can do what?); or it can be specific and it applies to a specific environment (who can do what in which environment?) or to a specific project (who can do what in which environment for which group of files?).

It is possible to define securities global, each application system and each project. The most detailed level of security definition is at project level.

- Global securities apply to all systems (and all projects).
- Securities that are defined for a system apply to the system itself and to all projects of that system.
- Securities that are defined for a project apply only for that project.

The example screen shows a security definition for an employee function programmer who has different securities for DEVELOP and PRODUKTIE and in PRODUKTIE extra securities for system DEMO.



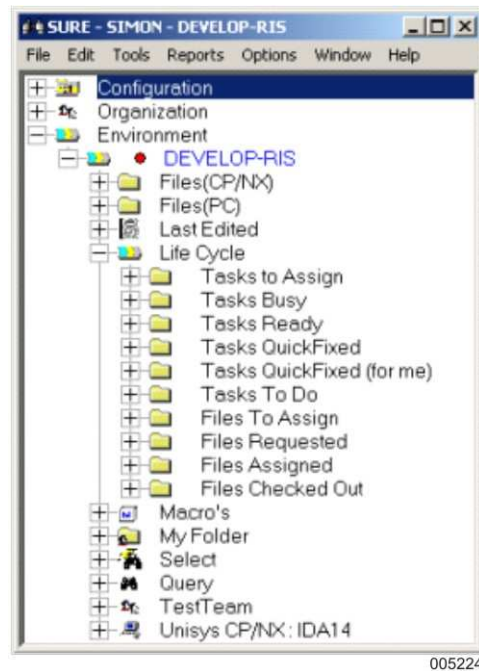
005223

- Task Names (Refer to Section 37, "Task," for more information )

The name of your task, constructed by the SURE system at task entry, is by default derived from the project. The default name-standard is the project name separated by a dash and a four-digit number (example, PRJ-1234). It is possible to overrule this naming-standard by your own formula, or by a naming-standard for task-groups.

- Task Routing (Refer to Section 37, "Task," for more information)

The task routing function affects the content of the workflow folders that are maintained by the system: Tasks To-Do, Tasks Busy, Tasks Ready, Tasks To-Assign, and Tasks Quick Fixed. The workflow queues are available for each environment and they contain tasks with a specific status (the ready-queue in the develop environment contains all tasks that have status DEVELOP and are marked as ready-(to-transfer)). These queues are visible in the user interface under folder.



- Life-cycle

The SURE system performs the task routing based on a number of variables, one of them being the project for which the task is defined. The other variables that are evaluated are the employee function and the team:

- A user can be assigned to a team and to one or more employee functions.
- One or more project can be assigned to a team or a user.

The tasks To Do list contains the tasks that are assigned to your personal usercode or to one of your employee functions or to one of your teams, and that belong to a project which is linked to your team or your personal usercode.

Using projects in combination with security can provide distinct separated logical parts in the repository. Note that the same functionality applies if you did not create projects, projects; however, the project name then is equal to the system name.

- Naming Standards (Refer to Section 39, "Name Standards," for more information)

Naming standards are used by the SURE system for the construction of task names (see earlier in this section) and file names.

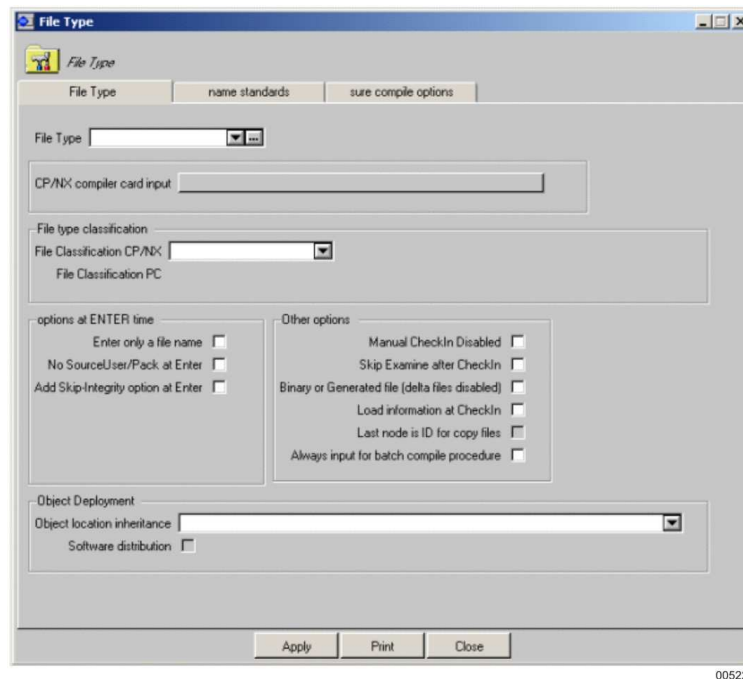
Naming standards for file names can be implemented as follows:

- The new file name is determined by SURE according to a predefined naming standard formula.
- The new file name is determined by the user, but checked by SURE according to a predefined naming standard formula.
- A mix of these two options.

The project name is one of the variables that you can use in this name standard definition.

- The naming-standard formula must be defined for each file type.
- The naming-standard formula must be defined for each system whether or not used.

### 9.6.4. (Configuration) File Type



The file type definition categorizes files with the same SURE characteristics. The file types are separated for the ClearPath server files and the PC files where both allow for slightly different characteristics. This document addresses the most commonly used file type characteristics as follows:

- ClearPath server shared resources

These are copy files (or other files) that must be visible for all team members in a shared work-environment.

If you are using a shared work-environment (see earlier in this chapter), then a file type is required for copy files (or other files) that you want to be refreshed in that shared work-environment after such a file is changed and checked in to SURE.

The file type option “Use as Copy file” in combination with the system option “Put copy files in the work-location after Check-in/Load” will make SURE copy a file of this type to the work-location. The work-location contains a stable version of each copy file. The developers use these stable copy versions to compile their programs.

- ClearPath server source files

Build support for the ClearPath server files requires an object-location to be defined for a file. This object-location triggers the build support program (RESPECT/SURE/COMPILE) to invoke the appropriate compiler when the file must be compiled. The appropriate compiler is determined from the file kind of the file. Source files are usually defined with a file type with option "Inherit default object user/pack" set. This option triggers that the file inherits the object-location defined for the system of the file. Therefore, the usercode, pack family name, and hostname defined in the object environment for the system of the file, are applied to that file. As a result, the appropriate compiler is invoked for this source file resulting in an object file, which is copied to the destination by the distribution support.

By default, the object name for a source file is named OBJECT/<source>. For individual files, you can alter this default under the file properties. If there is a companywide name standard for object names, then that name standard may be coded in a library, which is defined in the Global Options.

Copy files usually do not have option "Inherit default object user/pack" set, and that results in skipping the copy files for the actual compilation.

- ClearPath server data files

If you have data files that are stored in the SURE repository, a number of options may be of interest.

- First of all the SURE build support will never invoke a compiler, because the file kind does not match a supported compiler on the ClearPath server system.
- If you want these data files to be refreshed in the shared work-location, you need to set the file type option "Use as Copy file" in combination with the system option "Put copy files in the Work-environment after Check-in/Load."
- If you want these files to be copied into your production environment by the distribution support, you need to set file type option "Inherit default object user/pack."

**Note:** You can check in, check out, and transfer any type of file, also binary files. The binary files require two additional definitions, one which will skip the delta file creation and one that will skip the examine process. The corresponding options are Skip delta files after check in and Skip examine after check in. These two options are also applicable for ClearPath server data files.

- PC input build files

PC input build files are files that are used as input in a build process that creates PC executables. Example of PC input build files are: make-files, command-files, source-files, header-files, and project-files etc.

These build files are the primary input files to the build process. This definition excludes the intermediate files and it also excludes the resulting binaries or executables.

The build process on the PC is responsible for downloading changed files to your PC and thereafter starting the build command for a file or a directory. A file must have its object-build- server defined to make this download happen.

The mechanism is similar to the object-inheritance mechanism for ClearPath server files: The file type option "Inherit default object user/pack" in combination with the system attributes for the object environment defines the inherited build server for a file.

With this object server defined for a file, the build process has its parameter for download and the build process.

**Note:** *The build server name is a virtual name; it does not have to be an actual server name. Furthermore, the download directories are defined in the PC SURE options, and they are usually different each environment. For this reason, the build server name may be equal for every environment.*

- PC distributed files

PC binary files are files for which no delta files are created. When no delta files are created, no difference and reconstruct facilities are present.

A PC binary file is a manually checked in file, where SURE is used as archive mechanism. So, Microsoft office files or any other file may be used in this mode.

For example, a "manual checked-in distributed file" has the file type option "no delta files after check in" set, which results in ignoring delta files for every file of this type.

- PC copy files

PC copy files are files that are used as copy or include files within PC languages recognized by SURE. Current supported languages are C, C++ and Micro Focus COBOL. The SURE software scans sources of the previously mentioned languages for references to copy files. Often, only a part of the name is identified in the include statement, namely the last node of the file. The PC compiler finds the actual file using an environment variable like INCLUDE. The examine process uses the file type option called "Last Node is used as ID for copy files" to achieve the same type of behavior. As a result of this option, the source file is connected to the actual full copy file name if it exists for the node.

- PC C, C++ or COBOL source files

PC source files are files containing a language recognized by the SURE examine process. The examine process scans sources for references to copy files and stores that information in the repository. Currently, the languages C, C++ and Micro Focus COBOL are supported. Although, any type of language can be loaded into the SURE, the examine process only recognizes the above-mentioned languages.

If the examine process relates a source with a copy file, it will use this information in the compile process. If the copy file is changed, and therefore automatically compiled, the SURE system will implicitly recompile all the sources using this copy file. This side effect of the examine process is mostly used for the supported languages.

Setting this option requires the files to be loaded with a file type containing the indication "Use as C/C++" or "Use as Micro Focus COBOL."



- PC text files

A PC text file is any file containing plain text, of which you want to track the changes. Therefore, a PC text file may be an INI, BAT, PAS, TXT, and other such file. In principle, you could edit such a file with notepad. These files are stored in the SURE repository with their delta files. Using the delta files you can always trace the changes that are made to the source.

Defining PC text files requires a file type with the option “Skip examine at Check In” set.

The following table summarizes the described file types.

Type of File	Use as Copy File	Put Copy in Work	Inherit ... Object User/ Pack	Skip Delta File	Skip Examine	Last Node Used as ID for Copy	Software Distribution	C, C++, COBOL
CP/NX shared resources	✓	✓						
CP/NX source files			✓					
CP/NX data files		?		✓	✓			
PC input build files	?		✓			?		?
PC distributed files				?			✓	
PC binary files (MSOffice)				✓				
PC copy files	✓					✓		✓
PC C, C++ or Cobol source files	?	?	?					✓
PC text files							?	

 = This option is required for this file-type.  
 = This option may be applicable for this file-type.

005226

### 9.6.5. (Configuration) File

The file is the smallest entity stored in the repository. The file (and its contents) is stored in every environment in the repository.

Except for the file contents, a lot of other information is stored around the file entity and used within the SURE functionality:

- At Load or check-in time, the file content is stored in the repository. At check-out-time, the file content is retrieved from the repository and made available through a standard file. Therefore, SURE is required at check-in or check-out-time. On the other hand, a developer does NOT need SURE while he is working on the source.

In general, you can use any tool in the development environment, so development may still be done in the same way as always is done. This applies to ClearPath server files and PC and UNIX files.

For ClearPath server files, SURE offer a partial workbench, which allows usage of various PC editors integrated with the compilers on the ClearPath server machine. SURE allows local file editing in combination with a smooth integration with the ClearPath server compilers. Thus, the file is edited on the PC and compiled on the ClearPath server machine without any special actions from the developer.

If you are using NX/Edit, which also contains a partial workbench, then you won't use the full SURE workbench. In that case SURE is only used for check-in or check-out but NX/Edit handles the compilation of the source.

- SURE contains a version indication, which consists of a major and a minor number separated by a point. The major number indicates how many times a file is transferred from one environment to another environment. The minor number indicates how many times the file was checked in to the environment. An example is given in the following table.

<b>Environment</b>	<b>Step 1: Check Out/In</b>	<b>Step 2: Check Out/In</b>	<b>Step 3: Transfer</b>	<b>Step 4: Transfer</b>	<b>Step 5: Check Out/In</b>
DEVELOP	5.1 -> 6.1	6.1 -> 6.2	6.2	6.2	6.2 -> 7.1
TESTING	5.1	5.1	6.2	6.2	6.2
PRODUCTION	5.1	5.1	5.1	6.2	6.2

Step 1 and 5: The first check-out or check-in in a cycle raises the major number by one.

Step 2: A second check-out or check-in a cycle raises the minor version number by one.

- SURE does not allow checking out a file in the same environment by more than one user code at the same time (unless the baseline method is activated).



However, it does allow checking out the file in different environments at the same time. Therefore, while the file is checked out in development, it can also be checked out in production. This mechanism allows solving production errors without disturbing the development process. An example is given in the following table.

<b>Environment</b>	<b>Step 1: Check Out/In</b>	<b>Step 2: Check Out/In</b>	<b>Step 3: Reprocess QF</b>	<b>Step 4: Transfer</b>	<b>Step 5: Transfer</b>
DEVELOP	5.1 -> 6.1	6.1	6.1 -> 6.2	6.2	6.2
TESTING	5.1	5.1	5.1	6.2	6.2
PRODUCTION	5.1	5.1 -> 5.2	5.2 (reprocessed)	5.2	6.2

Step 1: the source is changed in develop.

Step 2: the source is quick fixed in production.

Step 3: the quick fix of production is reprocessed in develop.

Step 4 and 5: the develop sources (with the develop patch and the reprocessed quick fix) is transferred to acceptance and production.

- SURE keeps delta files of two successive file versions in an environment. With these delta files, it is possible to rebuild any old version of an individual file.

**Environment    Delta files between file version**

DEVELOP	- 5.1 and 6.1- 6.1 and 6.2
TESTING	- 5.1 and 6.2
PRODUCTION	- 5.1 and 5.2- 5.2 and 6.2

- Source management is a main function for SURE; however, other integrated functions such as Build support, Distribution support, Life-cycle support through task management also require definitions at file level. For this reason, the file entity contains a lot of additional information, which can be set through the file property function. A summary of information stored at file level is given in the next table.

Build Support	Build Server	PC and UNIX
	Build Command	PC and UNIX
	Executable	PC and UNIX
	Task Attributes	ClearPath Enterprise Servers
	Binder specification	ClearPath Enterprise Servers
	Multiple Object Names	ClearPath Enterprise Servers
	Post Compile Processing	ClearPath Enterprise Servers
Distribution Support	Distributed file	PC and UNIX

- |                    |                     |                              |
|--------------------|---------------------|------------------------------|
|                    | Object-Location     | ClearPath Enterprise Servers |
| Life-cycle Support | Current Task        |                              |
|                    | Historical Task     |                              |
|                    | Delta Files         |                              |
|                    | Log                 |                              |
| Query Support      | Run-Time Statistics | ClearPath Enterprise Servers |
|                    | Copy/Include Files  |                              |
|                    | Database            | ClearPath Enterprise Servers |
|                    | Dataset             | ClearPath Enterprise Servers |
|                    | etc                 | ClearPath Enterprise Servers |
- Files can be entered in the system in various ways. Initially files will be loaded each directory. For both file categories (ClearPath server and PC), a batch load facility is present in SURE.
    - ClearPath server files  
Select a directory or file from the ClearPath server source directory and select function "Load in SURE" from the property menu.
    - PC files  
Select the function Tools from PC-Environment and click Load files in SURE from the top-level menu. Files can be loaded from the defined local work directory in the SURE option.
- Loaded files require the attributes "Project" and "File Type." SURE uses these attributes to complete the individual file definition.
- Files can also be entered in the system using the Edit/New/File function from the main menu. This function will normally be used when new files are created in an existing project. However, the load function is always available and can always be used to load a single file or a directory of files.

The following table shows the functions that affect the life-cycle of a file.

Creation	Modification	Deletion
Edit/New/File	Check Out/Check In	Delete
Load using initial load option	Recover	Purge
	Replace	Rename
	Reprocess Quick Fix	
	Apply Quick Fix	
	Load using replace option	

## 9.7. (Configuration) Task Entities

- Task Priority
- Task Type

- Task Group
- Task

All definitions for tasks are assembled using the previously mentioned entities.

- A task priority categorizes a task in different priorities.
- A task type defines a task category including the start environment for the task. So normally you would create task types for each environment where you are going to make changes.
- A task group may combine a task type and a project. In this case, the name of the task is derived from the task group. If a task group is not used, the project name must be entered at task definition, and the name of the task is derived from the project name.

Refer to Section 37, "Task," for more information on tasktypes, task-groups, and about the flow of a task in the Electronic Data Processing (EDP) department

### **9.7.1. (Configuration) Task**

The task is the major controlling entity for life-cycle support, and defined at a global level in the repository. Global level means, the attributes of a task are visible and equal in all the environments. So, if you select a task in the environment production and you show the properties, you will see exactly the same information as in develop or in any other environment.

This differs from the FILE concept, because a file is stored in the repository at environment level. The attributes of a file in environment production are not always equal to the attributes of a file in the develop environment.

The task is the functional unit of work. It contains various descriptive texts or an attachment. Changes to files are linked to a task by issuing the task name at check-out-time. A task can be transferred to the next environment when all linked files are checked in. If the task is transferred to the production environment, then the task-status becomes "solved," and all linked files are added to the history of the task, and the task is added to the (logical) history of each linked file.

The task life-cycle can be integrated with an email system. If a task is assigned to a group of people or to an individual, an email can be sent to an email address. If a task changes from status, an email may be sent to the issuer of the task.

The task entry form can be customized so that it requires minimal input.

The following table defines the functions that affect the life-cycle of a task.

Creation	Status Change	Close
Edit/New/Task	Assign	Close
	Priority	Deny
	Ready	Delete
	Transfer	
	Activate	

### 9.7.2. (Configuration) Task Type

005227

The “task-type” is the only attribute for a task that has impact on the life-cycle functions that can be performed using the task. This is because the environment where the task starts (the task-environment) is inherited from the tasktype. A newly added task inherits the task-environment from its tasktype.

The task-environment determines the environment where fixes for tasks of this type must be made. You can only change a file if you are linked to a current task, and the changes must be made in the task-environment of that task.

The task-status determines if it is allowed to make changes for the task at this moment. You can only change a file if the status of your current task is equal to the task-environment of your current task.

Example

- Suppose you have a task with environment DEVELOP and status DEVELOP. It is now possible to make this task current in the develop environment and to change a file because of this task in the develop environment.
- If you transfer the task to the acceptance environment, then the task-status becomes ACCEPTANCE. You cannot make changes in acceptance because the task-environment is still DEVELOP, and you cannot make changes in develop because the task-status is ACCEPTANCE.
- If you reactivate the task again in develop task again in develops then the task-status becomes DEVELOP. You can now make additional changes in the develop environment.

A second important attribute of the tasktype is the quick fix level.

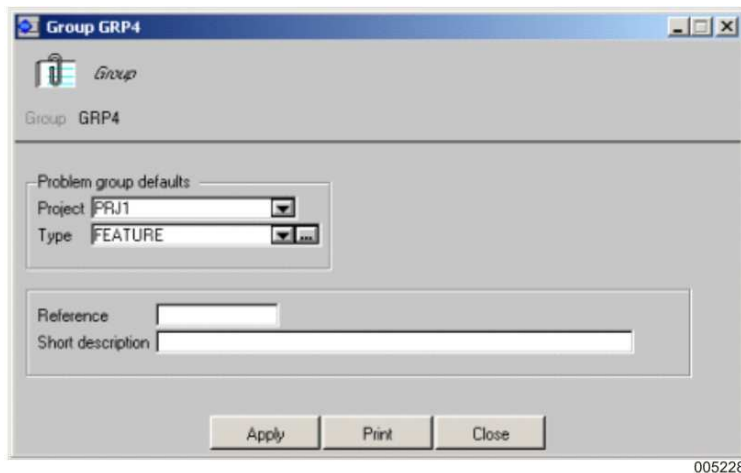
A quick fix is a modification to a file in an environment, which is not the lowest environment for that file. For example, consider a file that was created in the develop environment and transferred to acceptance and production. The lowest environment for this file is now the develop environment and a quick fix is a modification to the file in acceptance or in production.

The quick fix level identifies the environment where a task with this tasktype can be quick fixed. If this field is not entered, then the task cannot be used for quick fix purposes.

A separate task type is required for each environment where you want to make changes to files. For quick fixes, you can set the environment and quick fix level to the named environment. The following table shows the settings for the previous example.

Tasktype FEATURE:	Environment	= DEVELOP
	Quick Fix level	= <empty>
Tasktype QUICK FIX:	Environment	= PRODUCTION
	Quick Fix level	= PRODUCTION

The files can be modified in develop for regular maintenance and in production for quick fixes.



### 9.7.3. (Configuration) Task Group

A task group combines a task type with a project and additionally a short description.

Task groups are used for two reasons:

- The task naming facility.  
Task-names of newly added tasks are automatically determined by SURE. If you are using task groups then the new task name is based on the name of the task-group. If you are not using task groups, then the new task name is based on the project name.
- A shortcut when a new task is added:
  - A project can be linked to a task-group.
  - A tasktype can be linked to a task-group.
  - A task-environment can be linked to a tasktype.
  - A priority can be linked to a tasktype.

Each task must have a tasktype, a project and a task-environment. These three attributes are required if a new task is added to SURE. (Priority is an optional task attribute). A newly added task inherits the tasktype and project from the task-group, and the task-environment from the tasktype. If these three attributes are found through inheritance, then it is enough to enter the task-group when a new task is added; otherwise, the missing attributes have to be entered too.

A static task-group is a task-group that contains all three attributes: each task that is added through a static task group has a similar name (based on the name of the task group) and the same tasktype, project, and task-environment.

A dynamic task-group is a task-group that misses one (or more) of the three attributes: each task that is added through a dynamic task group has a similar name (based on the name of the task group), but the tasktype, project, or task-environment can differ.

### **9.7.4. (Configuration) Task Priority**

The task priority is used for query and reporting purposes.

The customer can define his own list of task priorities through explorer folder Configuration/Drop Down box value/Task Priority.

It is possible to define special customized workflow folders through the "Macro" function. The task-priority can be used in the task-selection-expression of such a customized workflow folder.

The task popup menu allows changing of this priority field. For this reason, the priority field can be used for moving tasks from one customized workflow folder to another.

## **9.8. (Configuration) Organization Entities**

- Employee Function
- Team
- User

For a user there are two important entities:

- The employee function or the user's role in the organization. The employee function contains a security map, which defines the actions that may be performed. Secondly, an employee function is used for task routing purposes. A task can be assigned to an employee-function and then all users with that employee-function receive the task in their To Do lists.
- A team is a group of people teamed together as a work unit. A team can also be used for task routing purposes. A task can be assigned to a team or to an employee function. Together, with the assigned projects for a user, the workflow queues are assembled.

The following paragraphs describe the above-mentioned entities with more detail.

Refer to Section 34, "Authorization Mechanism and Logon," for more information on usercodes, employee functions, teams, and the authorization mechanism.

Refer to Section 37, "Task," for more information on To Do lists.

### 9.8.1. (Configuration) Employee Function

The employee function is a role within a company performed by a group of employees. A usercode can be linked to zero, one or more employee functions. For this reason, one should choose elementary roles within a company.

The employee function is used in two different ways:

- Security

Every employee function may contain a security map. A user is linked to one or more employee functions. The combined security maps of these different employee functions determine the function matrix of the user.

- Task Routing

A task can be assigned to an employee function. Every user who has the employee function gets that task in his To Do list.

If a user is assigned to one or more projects then only the files or tasks of those projects are visible for him. For example, tasks of other projects would not appear in his To Do list, even if these tasks are routed to one of his employee-functions.

### 9.8.2. (Configuration) Team

A team is a group of usercodes. A usercode can be linked to zero, one, or more teams. Each user of a team may have different employee functions.

The team definition is used for the following:

- Project assignment

A team can be linked to zero, one, or more projects. A user inherits the (merged) project-lists of his teams. The merged team-project-list overrules an individual project list of the user.

The project-list works as a filter: if user does not have a project-list (directly or indirectly through his teams) then he can see all files. If he has a project-list then he can only see the files of his projects.

- Task Routing

A task can be assigned to a team. Each member of the team gets this task in his To Do list. The security is still defined at employee function.

- Contents of Life-cycle Work Flow folders

The content of the various team life-cycle folders is affected by the team definition. For example, the folder Life-cycle/Team Task/Busy contains all the busy tasks of the members of all my teams.



### **9.8.3. (Configuration) User**

The user entity contains an entry for each user that can log-on to the system. A usercode that is not announced in SURE cannot log-on to SURE. The log-on procedure validates the usercode and password against the userdatafile. For this reason, each SURE user also needs to be defined in the userdatafile.

The user definition supports (amongst others) the following:

- **Project Definition**

A user can be linked to zero, one or more projects. If a user is NOT linked to any project then all files or tasks of all projects are visible for him. If a user is linked to one or more projects, then only the files or tasks of those projects are visible for him.
- **Team Definition**

A user can be linked to zero, one, or more teams. These teams determine the project assignment of the user and affect the contents of his life-cycle workflow folders.

A project-list that is inherited from the team(s) overrules an individual project-list.
- **Employee Function**

A user can be linked to zero, one or more employee functions. These employee functions determine the security map for a user and affect the contents of his To Do list.
- **Individual Security**

Security may be defined at individual user level. The usercode where the SURE installation is started contains an individual security map. With this installation user the rest of the system can be configured. An individual security map overrules the security map that was inherited from the employee functions.
- **Default Environment**

A user is linked to a SURE-environment. Each time when he logs on to SURE, he will be routed to his default environment. From there, he can navigate to other environments.



## Section 10

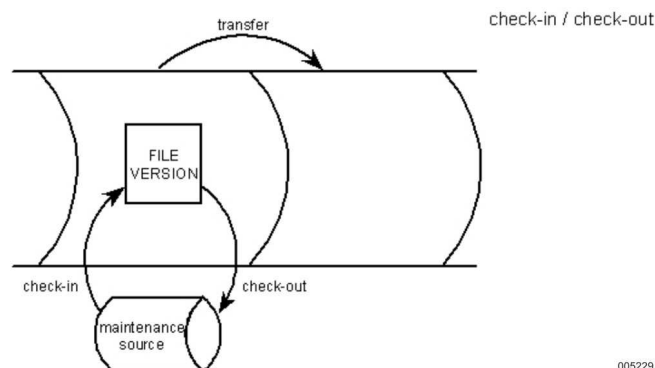
# Life Cycle Support

The life cycle of a source spans the time it is created, maintained, and deleted. SURE supports these phases through commands and automated procedures. Within the various phases, the programmer interacts with the SURE Windows user interface, and instructs SURE through formal commands. This interaction deals with check-in and check-out procedures. During source maintenance, the interaction with SURE is not required. Maintenance is done using tools such as CANDE System/Editor or a PC editor in combination with the SURE-local-edit option.

The life cycle support within SURE includes the creation and maintenance of the executables. Where the programmer deals with sources, SURE creates objects on the Windows build server and MCP mainframe because of check-in actions, source transfer, and the definitions in the repository. In this aspect, SURE differs from traditional source control systems.

The life cycle support covers the following procedures.

- Check in a source: store the source in the repository after maintenance.
- Check out a source: retrieve a source from the repository and make it available for maintenance.
- Transfer a source: transfer a task with sources to another environment (that is, Acceptance) after completing maintenance.
- Enter a new source: enter a single source or load a directory with sources.
- Delete a source: delete a source from the repository.



## 10.1. Check-In Procedure

The check-in procedure in SURE stores a source into the repository and removes its maintenance copy from the development environment. If a source is in maintenance, it exists as a physical file outside the repository, the maintenance source. Besides this maintenance source, the repository contains the original version of the source that was checked out. The maintenance source is modified by a developer, and later on stored in the repository because of this check-in procedure. The completion of the check-in procedure removes the maintenance source after storing it in the repository.

Function check-in can be used for a single file, for multiple selected files, and for a directory, as follows:

- Single file: right-click the file and use check-in from the popup menu. The check-in option is disabled if the file is not checked out.
- Multiple selected files: select multiple files, right-click and use function check-in from the popup menu. The check-in option is enabled if the first file in the selection is actually checked out. Files that are selected but not really checked out are skipped. Files that are checked out but not really changed get an undo check-out.
- Directory: all files in the directory that are checked out are checked in. Files that are not checked out are skipped. Files that are checked out but not changed get an undo check-out.

The main actions of the check-in function are:

- Store the maintenance source in the repository.
- Remove the maintenance source from disk.

Some other actions are triggered by the check-in function, such as:

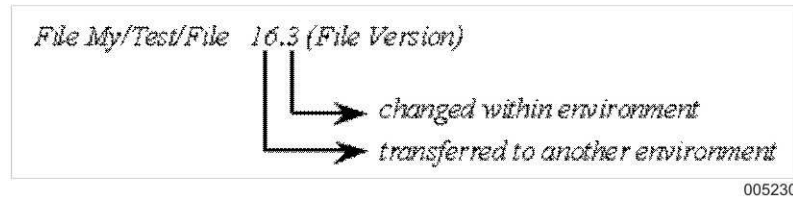
- Create a delta file between the new version and the previous version of the source.
- Scan the new version of the source for references to other files.
- Compile the new version of the source, and deploy the object to the correct location.

These actions are described in detail later on.

### 10.1.1. FileVersion Attribute in SURE

The FileVersion repository attribute describes the version of a source in the repository. A FileVersion consists of two numbers. The first number indicates the number of times a file is transferred from the development environment to the next environment. The second number identifies how many times the source was checked out and checked in before it was transferred.

### Example



This table lists a number of SURE actions and the resulted FileVersion attributes in some example environments.

Environment development	Environment acceptance	Environment production
New		
Check-in 1.1		
Check-out		
Check-in 1.2		
Check-out		
Check-in 1.3		
Transfer 1.3 →→	1.3	
Check-out		
Check-in 2.1	1.3	
Check-out		
Check-in 2.2	1.3	
Transfer 2.2 →→	2.2	
	Transfer 2.2 →→	2.2
Check-out		
Check-in 3.1		
	Check-out	
	Check-in 2.3	(Quick fix)
		Check-out
		Check-in 2.4
	Check-out	
	Check-in 2.5	
Transfer 3.1→→	3.1.	
	Transfer 3.1→→	3.1

Development is the default maintenance environment for this file type.

## 10.1.2. File Attribute RELEASEID

The RELEASEID file attribute is only applicable for MCP mainframe sources.

In MCP, the RELEASEID file attribute is used by SURE to identify and recognize sources and objects created by the SURE software. If a file is retrieved from the repository (for example, function copy, and check-out) or when it is compiled by SURE (after check-in or transfer), the value of RELEASEID is set by SURE with a value matching the FileVersion of the source in the repository.

### Example

```
RELEASEID = "SURE: Release <Environment>;Version <FILEVERSION>"

ON IDRD
*OBJECT : DIRECTORY
. RIS : DIRECTORY
. . MENU : DCALGOLCODE CODEVERSION=41.2 CP-ED PRIVILEGES: PU, TASKING
        ALTERDATE=01/05/1996 22:52:06 AREALENGTH=15120 (504 RECORDS)
        AREAS=38 AREASECTORS=504 BLOCKSIZE=270 BLOCKSTRUCTURE=FIXED
        CREATIONDATE=01/05/1996 22:52:06 CRUNCHED CYCLE=1 EXTMODE=SINGLE
        FILELENGTH=561480 FILEORGANIZATION=NOTRESTRICTED
        FILESTRUCTURE=ALIGNED180 FRAMESIZE=48 IDENTITY="RIS-M"
        LASTACCESSDATE=01/09/1996 12:20:17 LASTRECORD=18715 MAXRECSIZE=30
        MINRECSIZE=30 RELEASEID="SURE: Release DEVELOP-RIS; Version 76_1"
        CHECK-INFACOR=999 SECURITY=PUBLIC (I/O)
        TIMESTAMP=01/05/1996 23:02:03
        TOTALSECTORS=18720 VERSION=0
        Warning 133: The 'PRINTERCONTROLFILE' FILEKIND is not used. It will
        be deimplemented on SSR 43.1 (Scheduled for 02/1996).
```

As the example shows, the environment and the FileVersion are used to construct the RELEASEID attribute. The RELEASEID is under the control of Unisys MCP, if a file is modified, the RELEASEID is cleared, and a RELEASEID cannot be changed after it is set. This behavior is used by SURE to verify the validity of disk files in relation to repository versions.

Examples of verification using the RELEASEID attribute:

- If a file is checked in and the RELEASEID matches the current FileVersion in the repository (environment and FileVersion), SURE gives a warning that the file has not been changed.
- If a file is checked out of the repository and the disk file is resident with the RELEASEID matching the current FileVersion in the repository, this disk file is used.

The use of this RELEASEID attribute by SURE provides a visible match for disk files on the MCP mainframe. Each file created by SURE (source files and object files) gets this RELEASEID attribute with a SURE specific content. This makes it easy to identify these files and to separate them from manually created files.

### 10.1.3. MarkID

The MarkID is a specific column range for MCP mainframe files, which is used to store information about the origin of this line. For ALGOL or COBOL files column 80-90 is used.

A line that is modified during the editing phase gets an empty MarkID. If a file is checked in, then all lines with an empty MarkID (= all changed lines) get a SURE MarkID, consisting of the initials of the developer and the FileVersion. Lines with a markid that are not empty keep that MarkID.

The MarkID has the following layout.

Position	What
1 – 3	The initials of the developer who did the check in.
4 – 8	The new version number that the file obtained after the check in.
9 – 10	The release number of the system on which the file is residing. If the system has release number "YY" then these two characters are substituted with the current year. This gives a global indication how long ago a change was made.

If option "put task name in markid" is set (under global options, on the tab SURE), then the name of the task that was used during the check out is placed in the MarkID instead of the initials, file-version, and release.

If a new source is checked in for the first time, then each line gets an empty markid. This happens also when an existing file is used as a template to create a new file.

This mechanism makes it easy to see who changed a specific source line, because the MarkID is shown by SURE in the various inquiry functions. The RESPECT/SURE/SOURCELIST program shows the MarkID in the source listing.

### 10.1.4. Source Related Attributes

If a file is checked out, then SURE saves the information of the file location where it is placed. This information is used to verify that the same file, although changed by the programmer, is stored into the repository.

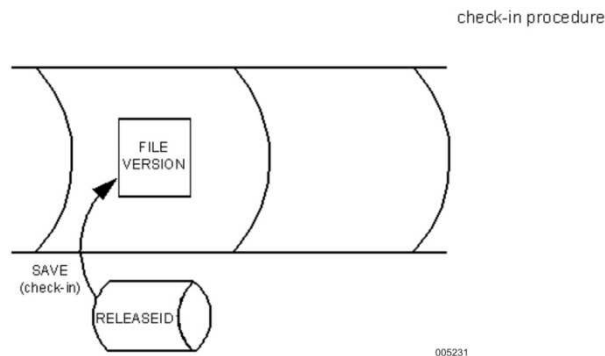
Using this mechanism, it is not possible to check out a file for USER\_A, copy it to USER\_B, and store it from USER\_B.

SURE uses four source related attributes of which two of them are used to verify the correctness of the file.

Attributes	Used to verify	Meaning
SOURCE-USER	x	usercode and userID
SOURCE-PACK	x	pack name
SOURCE-HOST		BNA host name
SOURCE-COPY		PC file name

The source related attributes are also used by the backup procedure: all files that have source related attributes are retrieved from the repository for maintenance purposes. A daily procedure copies each of these files from its maintenance location (indicated by the source related attributes) to tape. This ensures that the latest CANDE version of each in-use program can be restored in case of a calamity.

### 10.1.5. Function CHECK-IN



The CHECK-IN command is used to store a file into the repository. This command can only be used for files that are regularly placed on disk (using the CHECK-OUT or NEW functions).

A file can be checked out for editing using CANDE on the mainframe or it can be checked out for local-editing on a PC. If the file was checked out for local editing then option "local edit" is enabled on the Check In dialog. If the file is checked in with local edit enabled, then the current version is file-transferred from the local workdirectory on the PC to the CANDE-workdirectory on the mainframe, and from there checked in to SURE. It is possible to disable the option "local edit" on the Check In dialog and in that case the file-transfer from the PC to CANDE is skipped.

An authorized user can do a check in for another user.

**Note:** SURE loads the version of the checked out source in SURE that is at the moment resident under the work location of the original users on the mainframe.



**Example**

- Usercode SIMON has the source checked out.
- His work location is (SIMON) ON IDR.
- The source is checked out locally, which means that the modifications are done using a local PC editor such as MultiEdit.
- With a compilation, the source is uploaded from the local workdirectory on the PC belonging to Simon to (SIMON) ON IDR at the mainframe.
- When the source is checked in by another user, then the version of the file which is at that moment resident under (SIMON) ON IDR is loaded in SURE. The PC version of the source (which is actually the most recent version) is not loaded and remains on the PC of the original user who did the check out (user SIMON).

## 10.2. Check-Out Procedure

The check-out procedure of SURE copies a file out of the repository, as a maintenance copy, that is used by the programmer for maintenance purposes. This maintenance copy is modified and tested using CANDE System/Editor or a PC editor and needs to be checked in after completion of the maintenance.

The check-out procedure consists of three phases allowing an organization to split responsibilities over different functions. The three phases are REQUEST, ASSIGN, and GET. A developer requests a source, and links it to a task. The project leader checks the request (for example, is it really necessary to modify that source because of that task?), and assigns it to the developer if he agrees. The assign function connects the file to the developer and prohibits other programmers to get the source until it is transferred to another environment or explicitly assigned to that other developer. Function "get" copies the file from the repository as a maintenance copy to the workspace of the developer.

These three phases may not be required for every organization. Therefore, they are also combined as one function. Therefore, if check-out command is executed, and the employee is allowed to perform all three actions, then all three actions are done.

Function check-out can be used for a single file, for multiple selected files and for a directory.

- Single file: right-click the file and use the check-out function from the popup menu. The check-out function is disabled if the source is already checked out.
- Multiple selected files: select multiple files, right-click and use the check-out function from the popup menu. The check-out option is enabled if the first selected file is not checked out. Files in the selection that are already checked out are skipped.
- Directory: all files in the directory are checked out. Files that are already checked out are skipped.

### 10.2.1. Check-Out Attributes

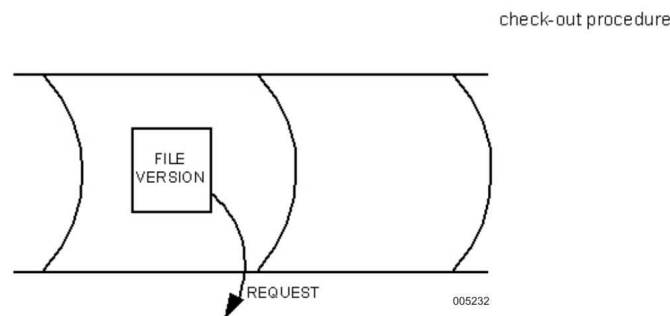
In the check-out procedure, SURE uses a number of attributes to control this procedure.

The REQUESTED (<USER-ID>) relation is a result of the REQUEST function and indicates the user that requested the source for maintenance.

The ASSIGNED (<USER-ID>) relation is a result of the ASSIGN function and assigns the file to this specific user until the user releases the file. Releasing the files is performed by promoting it to another environment or by an assignment to another user (if the file is not checked out). This attribute of SURE prohibits that no other user can modify this file simultaneously.

The source-relations SOURCE-USERCODE (<USER-ID>), SOURCE-PACK (<PACK NAME>), SOURCE-HOST (<BNA-HOSTNAME>), and optionally SOURCE-COPY (<PC FileName>) are set when the maintenance copy of the file is created. SURE uses these attributes to validate the file at check-in time.

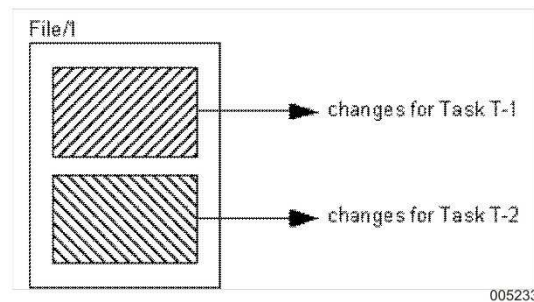
### 10.2.2. REQUEST



The REQUEST command (file popup menu) assigns a task to a file and makes the file available for a specific user through the REQUESTED (<USER-ID>) relation.

The task assigned to the file is the current task assigned to this user. The Windows interface allows changing this task by selecting the task button from the user object.

Requesting a file performs the impact overlap verification and this may result in an overlap warning. It must be emphasized that overlapping tasks result in changes for each of these overlapping tasks in the same file. Hereafter, you cannot transfer the file to another environment without the modifications of the other tasks. At transfer time, the file with all changes is transferred and SURE will not create a file with the changes for a specific task.

**Example**

Task T-1 and task T-2 are overlapping because source File/1 contains changes for both tasks. If task T-1 is transferred, File/1 is transferred to the next environment but it also contains changes for task T-2.

A quick fix or using patch files may avoid overlapping tasks.

The REQUESTED (<USER-ID>) relation is removed after function check-in. A request can be performed for another user while the file is requested, assigned, maintained, or after the file is maintained, by the programmer. The ASSIGN (<USER-ID>) relation just enforces the right for a programmer to maintain the file.

It is possible to have more REQUESTED (<USER-ID>) relations each file. This can be for one user or for different users.

**Examples**

Requested by	JOHN, FRANK	The file is requested by users JOHN and FRANK. The file is assigned to user JOHN.
Assigned to source-usercode	JOHN	
Requested by	JOHN	The file is requested by user JOHN.
Assigned to source-usercode	JOHN	
Requested by	JOHN	The file is requested and assigned, however the user never got the file. Otherwise, relation REQUESTED (<USER-ID>) was removed.
Assigned to source-usercode	JOHN	
Requested by	JOHN	The file is in maintenance by the user.
Assigned to source-usercode	JOHN	
Requested by	JOHN	The file has been maintained by the user and he has checked the source into the repository. The assign relation prohibits any other user to check out the file.
Assigned to source-usercode	JOHN	
Requested by	FRANK	The file has been maintained by JOHN and is requested now for FRANK. After the file is assigned to user FRANK, this programmer may check out the file.
Assigned to source-usercode	JOHN	

Requested by	FRANK	The file has been requested by and assigned to FRANK. The file is not yet maintained by FRANK; however, it may have been maintained prior to the request as this set of conditions shows.
Assigned to	FRANK	
source-usercode		

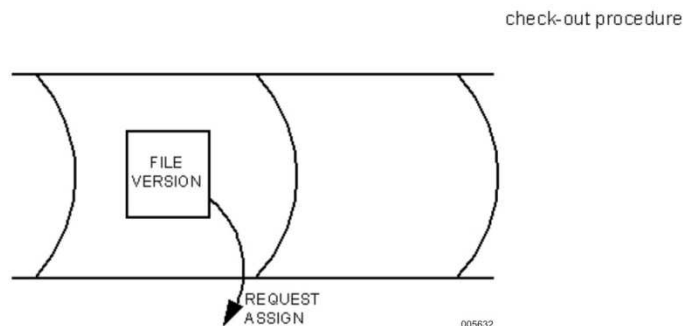
**Note:** The *UNDO REQUEST* (file popup menu) command restores the previous status for the file in respect to the *REQUEST* command.

If there is more than one request outstanding for a file, the UNDO REQUEST shows a continuation screen with all the tasks that requested the file. One has to click on one of those tasks to reset it.

### 10.2.3. LINK

The LINK command links a file to the current task of the user. Sometimes, a file has to be linked to a task, but the file should not be modified. The LINK command can be used for that purpose.

### 10.2.4. ASSIGN



The ASSIGN (file popup menu) command assigns a file to the user who requested it. Now, the file is available for maintenance by the assigned user.

The ASSIGNED (<USER-ID>) relation is removed after the file is transferred to another environment. SURE allows assigning the source to another user, if that other user requested the source. In other words, the assigned user has private access to this file until it is transferred or requested again.

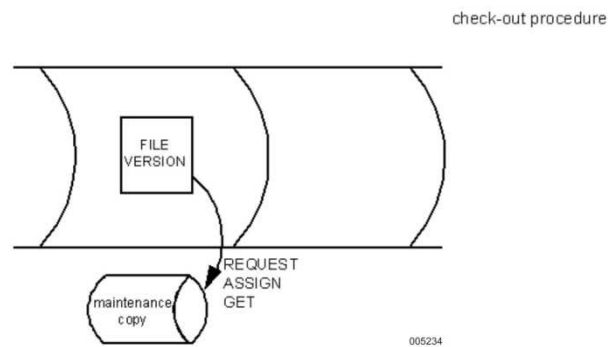
**Note:** *UNDO ASSIGN* (file popup menu) command restores the previous status for the file in respect to the *ASSIGN* command.

### 10.2.4.1. Assign a File to Another user+task Using Drag/Drop

The drag/drop a file on a task does the following:

- If the task is not yet assigned to anybody: the task and the file are assigned to myself.
- If the task is already assigned to a user: the file is also assigned to that user.
- If the task is already assigned to one of my teams or employee-functions: the file is assigned to myself.
- If the task is already assigned to another team or employee-function: error.

### 10.2.5. Function CHECK-OUT



The CHECK-OUT (right-click a file and select check-out) function copies the file from the repository as a maintenance copy that can be maintained by the programmer. SURE verifies that the file is assigned and that it is not in use by another developer through the ASSIGNED (<USER-ID>) and the SOURCE-USERCODE (<USER-ID>) relations.

ONLY	Copy files that are used by the source are not copied to disk.
OVERRIDE	An already existing disk file with the same name is overwritten.
DOMAIN	The source is marked with a DOMAIN relation. Only users with that employee-function are allowed to adapt this source.

**Note:** The CHECK-OUT command implicitly performs tasks of the REQUEST and ASSIGN commands on the file if the user is authorized. Therefore, at installations where the programmer is allowed to perform all three steps, the CHECK-OUT command is sufficient.

The disk location of the file is determined by the definition of the work-environment for the system of the file. The system definition contains the attributes WORK-USERCODE (<USER-ID>), WORK-PACK (<PACK NAME>), and WORK-HOST (<BNA-HOSTNAME>).

### Source Location from Windows Interface

The Windows interface creates a file using the system defined work attributes. If the work usercode equals to "\*", the usercode logged on is used.

The file is copied as a maintenance copy on its location with the same ClearPath server file attributes it was checked in (FILEKIND, AREASIZE, MAXRECSIZE, BLOCKSIZE, UNITS). Additionally, the RELEASEID attribute is set to indicate the FileVersion from the repository environment. This RELEASEID attribute is cleared after the file has been updated.

After executing the CHECK-OUT function, the source is available for maintenance and test procedures.

## 10.2.6. UNDO CHECK-OUT

The UNDO CHECK-OUT function removes the check-out indication and restores the previous state of the file.

The work-version in the file in the local work-directory is removed.

Authorized users can do an "undo check-out" for other users.

UNDO CHECK-OUT can be used for a single file and for multiple selected files.

OOPS is a term sometimes used when clicking a windows dialog and just realizing what you did. This can also be the case when doing an UNDO CHECK-OUT and answering YES to the question whether to remove the modified work file. For this reason, SURE creates a <file>.<ext>.bak file when doing an UNDO CHECK-OUT.

This <file>.<ext>.bak file is also created when checking in an MCP source, because the patch file merge mechanism may alter the uploaded file. The .bak file contains the original file.

### 10.2.6.1. Function UNDO CHECK-OUT Leaves the Source on Disk (renamed)

The UNDO CHECK-OUT function changes the CANDE file to directory UNDO/= instead of removing it. If the CANDE work-directory is also the copy file directory then restore the source from the repository if it is a copy file.

Undo check-out:

- If the CANDE version of the source was not yet changed, remove it.
- If the CANDE version of the source was changed (using COMPILE or UPLOAD ) change to UNDO/<source>.
- If the source is an include-file and the CANDE work-directory is also the include-file directory, restore the previous version of the include-file from the repository.

The advantage of this method is that the last changes are kept on disk in a separate file. So, if the undo check-out was done by mistake, then you can always restore that version.

```
lfile undo :releaseid
#RUNNING 1389
#?
ON IDRDR
(SIMON) : DIRECTORY
. UNDO : DIRECTORY
. . RESPECT : DIRECTORY
. . . SURE : DIRECTORY
. . . . COMPILE : DMALGOLSYMBOL
#
```

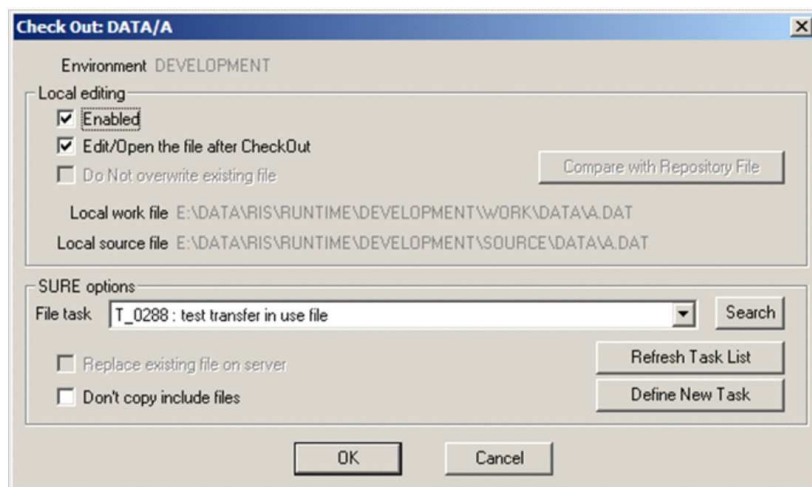
It is the responsibility of the developer to cleanup his personal usercode directory.

**Note:** An automatic cleanup is possible with the RESPECT/SURE/CLEANER program.

### 10.2.7. Manipulate MCP Data Files Using Local Edit

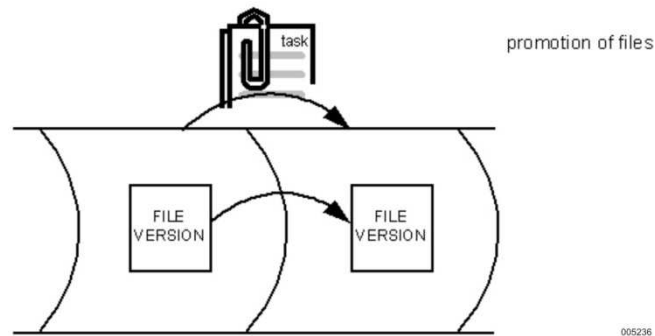
SURE software handles all kinds of MCP filekinds. Although, most of the MCP file are source-files (for example, COBOL, ALGOL, job, and seqdata), it is also possible to load DATA files in SURE.

If you check out a DATA file and set the local edit option, the default SURE editor is started. The file is downloaded and CR and LF are inserted after each logical MCP record. When saving the file, these CR and LF characters are removed again and the original RECORD and BLOCKSIZE is set for the file.



005235

## 10.3. Transfer



Transfer of files to another environment is performed by transferring a task. Transferring a task verifies that files have been modified and checked in into the repository. If a file, assigned to the task to be transferred, does not match both criteria, the task cannot be transferred.

These error conditions in a task transfer may be solved by:

- UNDO REQUEST function, if the file did not require modification.
- CHECK-IN function, if the file was still in maintenance.
- ASSIGN, CHECK-OUT, or CHECK-IN function, if the modification for the file was forgotten.

Transfer of a task to another environment copies all attributes in the group file to this other environment and it triggers the file change procedure in the transferred environment. Every file assigned to the task is automatically checked in to the transferred environment, causing the same actions as for a manual change.

Because the environment may contain different variables and definitions, the result may be different in each environment. For example, the match result or examine results may not be the same in every environment.



### 10.3.1. FileVersion Attribute

The FileVersion attribute consists of a version and a cycle number. This version number is incremented after transfer of a file from the default maintenance environment to another environment. The cycle number is a unique number within a version for a file.

Transferring a file from the default maintenance environment causes the version number to be incremented next time the file is checked in to the default maintenance environment.

The receiving environment contains a copy of the FileVersion after transferring. This attribute is derived from the STOR (NULL) relation that contains the FileVersion attribute in its data.

### 10.3.2. STATUS Attribute

The STATUS attribute informs the user with a mnemonic value about the status of a source. Although the status of a file may contain a complex combination of situations, the status field encapsulates the status as a single value.

Each environment in SUREforWindows is marked with a unique color. A non-assigned file is marked with a status-icon. The color of that status-icon is equal to the COBOL of the environment from which this file is a copy.

In one phrase, the status is the value of the higher environment from which it is a copy. Thus, if the file is changed in an environment, the status of the file in this environment equals the name of this environment. If the file is transferred to a higher environment, the status is set to the name of the transferred environment.

#### Example

Action	Environment develop	Environment acceptance	Environment production
Original status	production	production	production
Check-out, modify, check-in	develop	production	production
Transfer	acceptance	acceptance	production
Check-out, modify, check-in	develop	acceptance	production
Transfer	acceptance	acceptance	production
Transfer	production	production	production
Quick fix in acceptance	develop	acceptance	production

### 10.3.3. Anchored files

Anchored files are environment specific. These files contain information that is only applicable for a specific environment that is why such a file cannot be transferred, it is anchored to the environment.

An example of an anchored file is a configuration file. The content of a configuration file is often environment specific. It does not make sense (and it may cause run-time errors) to transfer such a configuration file from one environment to the other environment.

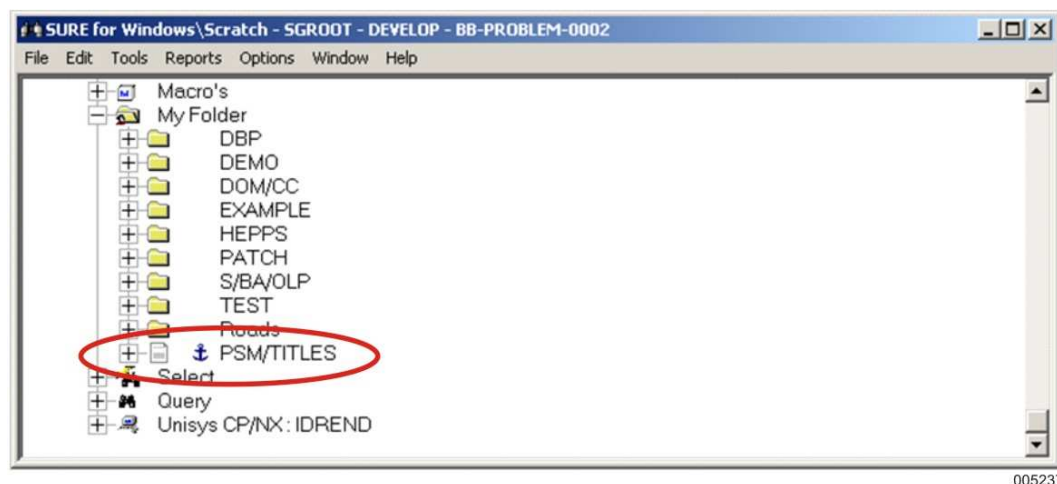
Anchored file can be modified without a task and they are not attached to a task. File attributes can also be modified without a task. Anchored files can be checked in or checked out without a task in the appropriate environment, and they cannot be transferred to another environment.

Anchoring a file is done as follows:

1. Right-click the file name.
2. Click **Miscellaneous**.
3. Click **Anchor**.

Or a file can be anchored as follows:

1. Right-click the file name.
2. Select **File Properties**.
3. Select **Sure**.
4. Select **File is Anchored**.



An anchored file is marked with an anchor.

## 10.4. Enter a New File

Entering a new file into the system consists of a first step, which registers the file into the SURE repository. In combination with the registration, additional functions may be performed by SURE depending on the configuration. This function is performed by the selecting File under New menu. You can load a file in the batch mode by using the LOAD utility located on the Tools menu under CP/NX or PC.

**Note:** The file is available in other environments after transferring the task. Until that time, the file is only available in the environment it was entered.

### 10.4.1. (Life Cycle Support) FileName

The value of the FileName is dependent on the NAMESTANDARD option configured for the application system of this file. The Project field is a required item when a new file name is added to SURE. This project is linked to a system in the System or Project configuration. The SYSTEM definition contains an option that defines if the NAMESTANDARD option is set for the system.

If the name-standard option is disabled, the FileName entry field contains the value for the actual FileName, which is to be entered in the system. This mechanism is familiar to programmers, where they can choose their FileNames. (This may be standardized by a written description).

If the name-standard option is enabled, the FileName may be empty or it may contain a parameter for the name standard routine executed by SURE. The file type contains syntax for the construction of the FileName. Depending on this syntax, the FileName is a parameter, or it may be empty. The SURE response provides the FileName.

#### Example 1

```
PROJECT      MYPROJ
SYSTEM       MYSYS      (NAMESTANDARDS = SET)
FILE-TYPE    PROG       (Syntax for name standard)
                  'S/' <PROJECT> '/' (XXX)
```

```
Entered FileName      ABC
SURE Provided FileName S/MYPROJ/ABC
```

#### Example 2

```
PROJECT      MYPROJ
SYSTEM       MYSYS      (NAMESTANDARDS = SET)
FILE-TYPE    PROG       (Syntax for name standard)
                  'S/' <PROJECT> '/' { 'PROG' <PROJECT> , 3 }
```

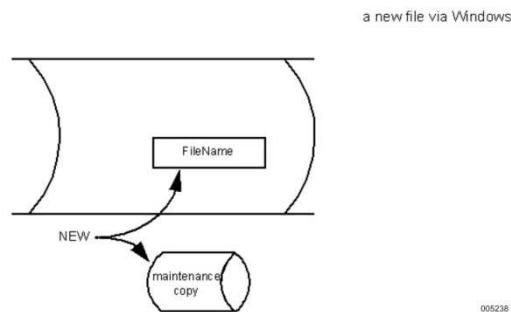
```
Entered FileName      <empty>
SURE Provided FileName S/MYPROJ/001
                  S/MYPROJ/002
                  ..S/MYPROJ/nnn
```

### 10.4.2. Create a Maintenance Copy of the File

Registration of the FileName in SURE is the first step in creation of a new file. After the FileName has been registered in the SURE repository, it has a checked-out status. This means that the maintenance copy may be checked in.

The NEW or FILE command requires entering a FILEKIND for ClearPath server files. The maintenance copy of the source is then created in the workspace of the developer.

### 10.4.3. New/File



The NEW function provides entry fields for additional attributes for a file.

A number of attributes are mandatory and used by SURE to evaluate options and determine the process behavior. The entry of the attributes, PROJECT, FILE-TYPE, and FILEKIND are mandatory.

All other attributes for the NEW function are optional and documentary only. An organization may use these attributes to structure their application systems and use these attributes in queries.

The NEW command provides options for ClearPath server origin files and PC origin files. ClearPath server origin files may be maintained on the ClearPath server. PC origin files however, can only be manipulated using the SURE (Windows) interface. The repository is used as an archive and versioning module for these types of files.

## 10.5. Load Files in SURE

The LOAD function can be used to load large groups of new files in SURE, or to reload files again in SURE without having them checked out earlier.

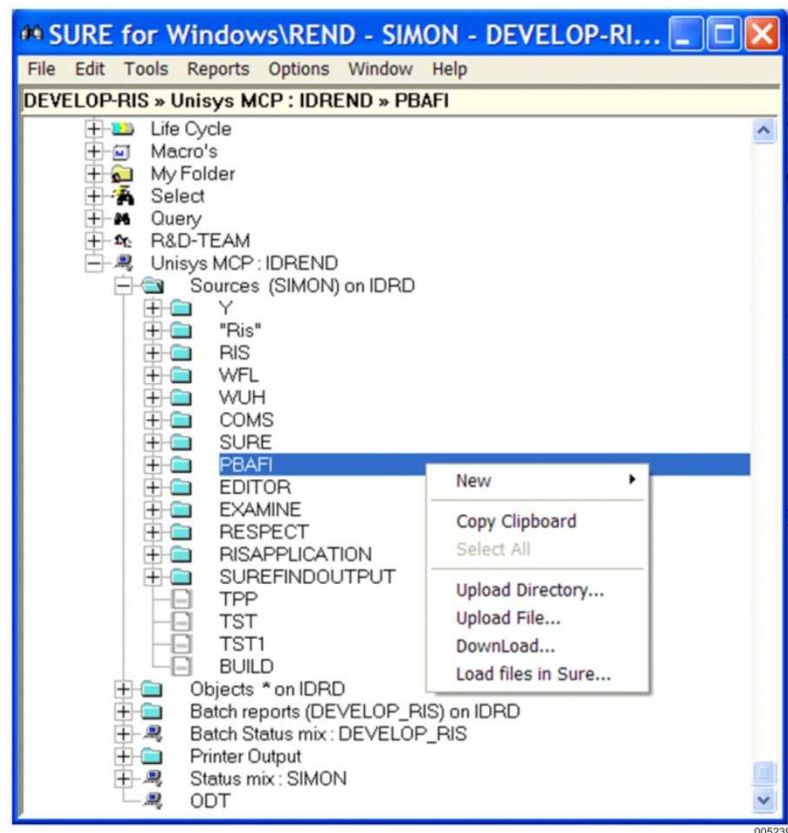
Many PC developers use the LOAD function to introduce new files in SURE, where MCP developers often use the NEW function on File menu for that purpose.

### 10.5.1. Load MCP Files in SURE

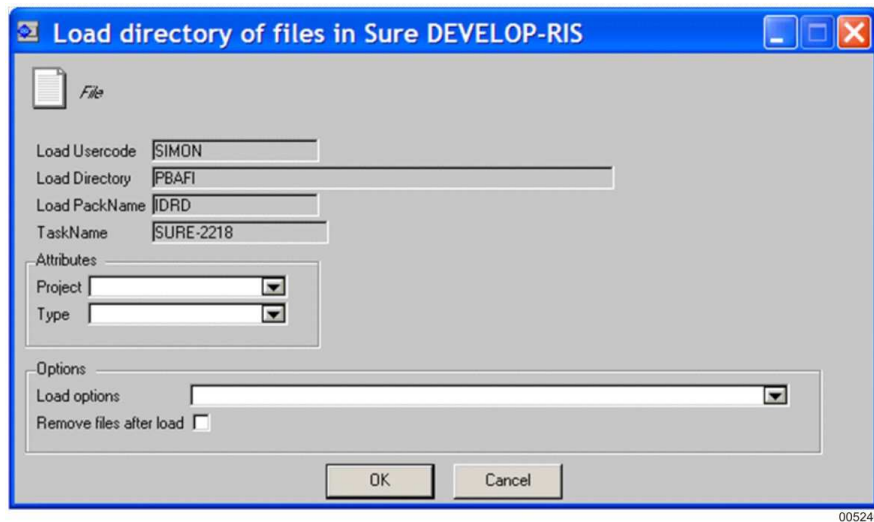
The “Load MCP files in SURE” function is normally used when all sources of an application system have to be loaded in SURE for the first time.

Load MCP files as follows in SURE:

- Copy the directory with files to be loaded to your CANDE- work-environment on the mainframe.
- Go to the SURE browser, open the folder Unisys MCP, Sources (see the screen snapshot below).



- The directory to be loaded is visible between all the other directories. In the screen snapshot it is directory PBAFI/=.
- Right-click the directory and use the “Load files in SURE” function.
- A continuation screen is given where you must choose a task. All the loaded files will be linked to the task that you select.
- On the second and last continuation screen (see below) you must enter a project and a file type.



- All loaded files will get the entered project and file type. If you want to load files with different file types or with different projects, then you must load those in a next phase.
- With the "Load option" you can choose to load new files or to reload existing files (or both).
- If option "Remove files after load" is checked, then all files that are loaded correctly are removed from your CANDE directory.

### 10.5.2. Load PC Files in SURE

Function "Load PC files in SURE" is used in the following cases:

- All sources of an application system are loaded in SURE for the first time.
- Some new programs of an existing application are loaded in SURE for the first time.

Load PC files in SURE as follows:

- Copy the directory tree of the sources to be loaded as a subdirectory in your local work-directory. Your local work-directory is defined on the local options tab under Options, SURE Options.

#### Example

Suppose the directory tree of sources to be loaded is:

```
C:\Car\Lease\*.*  
C:\Car\Buy\*.*
```

The local workdirectory is C:\SURE\Develop\Work\

In this case, the directory tree must be copied as follows:

```
C:\SURE\Develop\Work\Car\Lease\*.*  
C:\SURE\Develop\Work\Car\Buy\*.*
```

Go to the SURE browser and choose one of the following functions:

1. From <menubar>, click **Tools**.
2. Click **PC environment**.
3. To load some directory in SURE, select **Load files in SURE**.

All the loaded files will get the same file type.

1. From <menubar>, click Tools.
2. Click PC environment.
3. To load an entire directory in SURE, select Load directory of files in SURE.

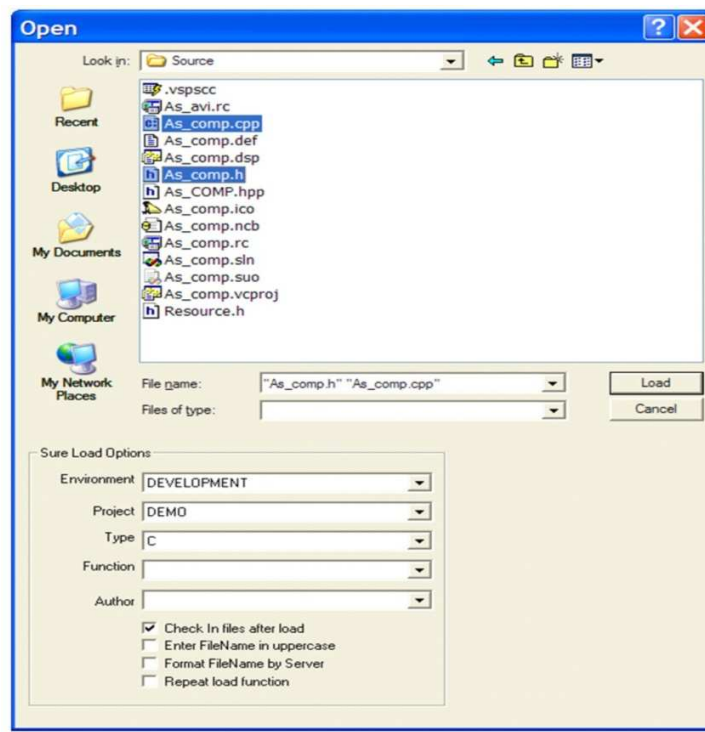
For each file you can choose a specific file type.

### 10.5.2.1. Load PC Files in SURE

1. From <menubar>, click Tools.
2. Select PC Environment.
3. Select Load PC files in SURE.

A continuation screen is given where you must enter a task. All the loaded files will be linked to the task that you select.

On the second and last continuation screen (see below) you must select the files to be loaded.



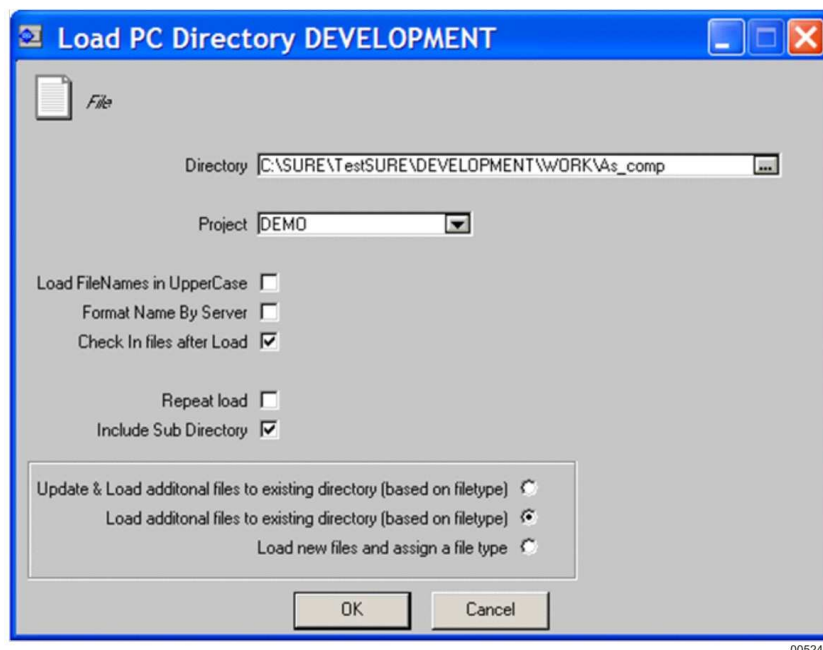
- The example screen shows the contents of a directory with files. Two files are selected to be loaded in SURE: as\_comp.cpp and as\_comp.h.
- All loaded files will get the entered project and file type. If you want to load files with different projects or different file types, then you must load those in the next phase.
- If option “Load filenames in uppercase” is checked, then each file name is translated to uppercase before it is loaded in SURE.
- If option “Format name by server” is checked, then the first character of each node of the file name is translated to uppercase, and the rest of the node to lowercase.
- If options “Load filenames in uppercase” and “Format name by server” are both off, then no uppercase or lowercase translation is done on the file name.

### 10.5.2.2. Load Directory of PC Files in SURE

1. From <menubar>, click **Tools**.
2. Select **PC environment**.
3. Select **Load directory of files in SURE**.

A continuation screen is given where you must enter a task. All the loaded files will be linked to the task that you select.

On the second continuation screen (see below) you must enter a directory to be loaded.

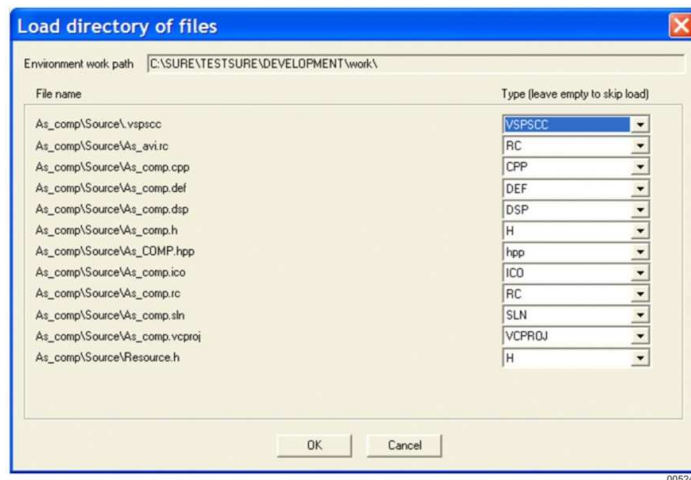


- The example screen shows a directory C:\SURE\Testsure\development\work\As\_comp. The name of the local



workdirectory is C:\SURE\Testsure\development\work\, so the directory to be loaded is AS\_comp\\*.\*

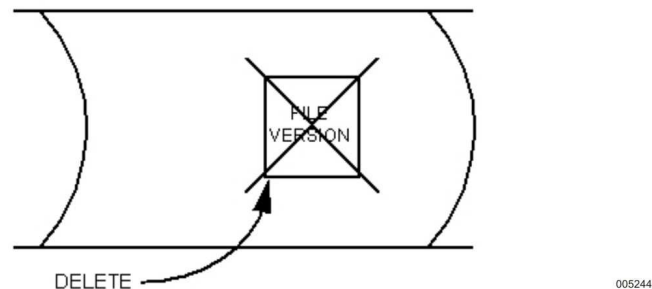
- All loaded files will get the entered project. If you want to load files with different projects, then you must load those in a next phase.
  - If option “Load filenames in uppercase” is checked, then each file name is translated to uppercase before it is loaded in SURE
  - If option “Format name by server” is checked, then the first character of each node of the file name is translated to uppercase, and the rest of the node to lowercase.
  - If options “Load filenames in uppercase” and “Format name by server” are both off, then no uppercase or lowercase translation is done on the file name.
  - Notice option “include subdirectories.” If this option is checked, then the files in subdirectories are loaded too.
  - Function “Load new files” loads a new directory.
  - Function “Load additional files” adds new files to an existing directory.
  - Function “Update & Load additional files” loads each changed file (where the modify timestamp of the file on disk does not match to timestamp in SURE). If the file is not yet known in SURE, then the file is loaded too.
- On the third and last continuation screen (see below) you must enter the file types.



- SURE assumes a default file type (equal to the file-extension) for each file in the directory. It is possible to overrule that with a specific file type.

## 10.6.Delete a File

delete a file



It is possible to delete a file logically or physically.

Deleting a file logically is done with the command "Logical Delete" (file properties maintenance). The file can be reactivated with the command "Undo Logical Delete" (file properties maintenance). These functions need a current task. The logical delete is only executed for the current environment. The logical delete is transferred with the task to other environments.

Deleting a file physically is done with the command "Purge in this Environment" (file properties maintenance). In this case, the command "Undo all Changes" (file properties maintenance) can be used to recover (an older version of) the file from the next environment. The physical delete does not need a current task. The physical delete is only executed for the current environment, and is never transferred. The physical delete is mostly used for cleaning purposes.

Deletion of the file is done in the current environment for the user. For this reason, a logical deleted source still exists in the higher environments, until the task is transferred to status solved.

The environment status indicates the deletion by character "R" or "r" for the environment where the file is deleted.

## Section 11

# Local Source Editing on PC

The local source editing option offers a workbench for the ClearPath server developer.

With SURE, a source file can be copied from the repository to the PC, where it can be edited by any editor the user prefers. While in maintenance, it can be compiled at the mainframe, with the results of such a compilation being returned to the PC.

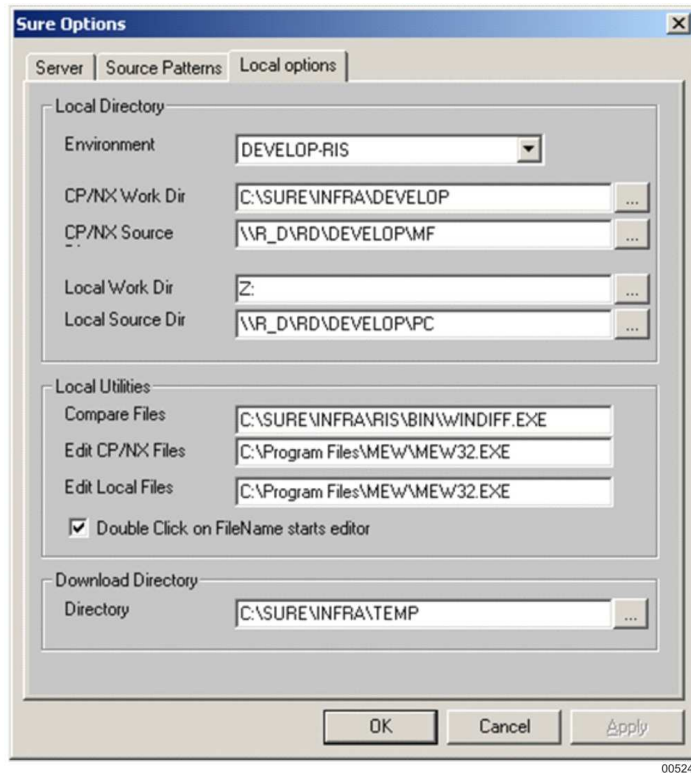
In this way, one can use “the best of two worlds,” the speed and ease of PC editors, combined with the compilation power of the mainframe.

The functionality for this feature is concentrated in the following utilities:

SURE	SURE user interface for Windows.
AS_COMP	Starting a compilation from a Windows environment.
AS_RUN	Merging error files.
Editor support	Configurations and tools for a number of PC editing environments.

### Configuration

A work and a source directory must be defined for each SURE environment. This must be done on the Options menu, Sure options window, Local options tab.



The "CP/NX Work Dir" and "CP/NX Source Dir" fields are used to identify the work and source directory for PC files.

### Entry

CP/NX  
WorkDirectory

### Usage

Directory to store the local work file for the ClearPath server and the transferred error file. This directory is empty when all files are checked in.

CP/NX  
SourceDirectory

Directory to store the local source file for the ClearPath server. This local source is a copy of the file that is stored in the repository. The local source is read-only. The developer should not modify any files in this directory. SURE synchronizes the file in the source directory automatically with the repository if that is necessary.

It is required that the work and the source directories are different.

The source directory should be a directory on a server that is visible for all developers. If a developer wants to edit or view a file and the correct version of that file is already available in the source directory, then that version is used. If the file is not available in the source directory or if an invalid version is resident, then the version in the source-directory is first synchronized with the correct version from the repository.

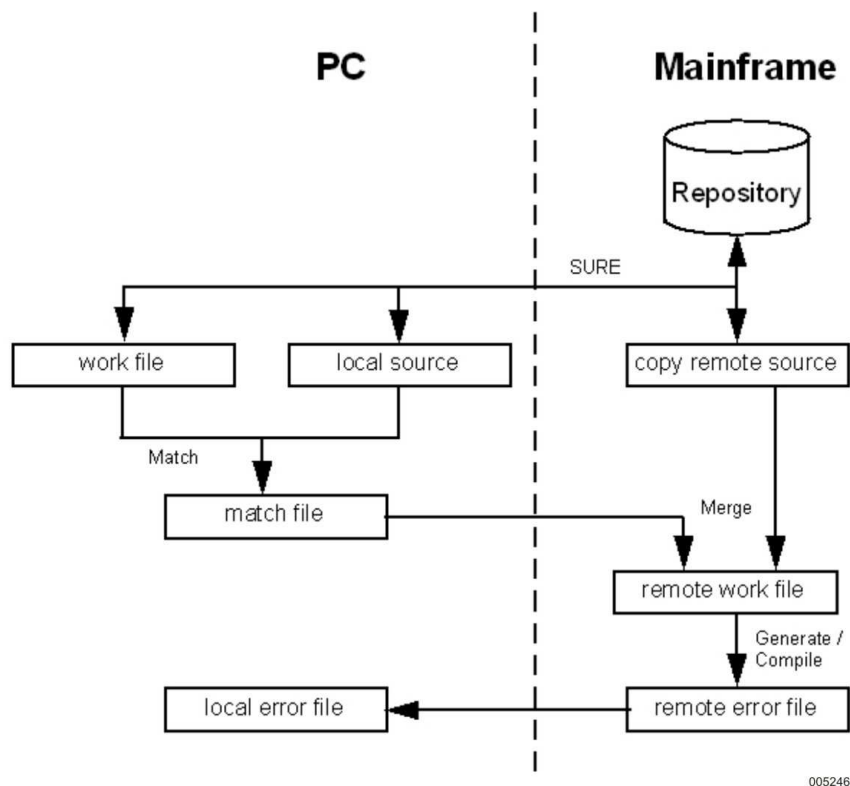
The source directory and work directory are local options and can differ for every PC.

A commonly used situation is that all desktop PCs have the same settings for these two attributes: a work directory on a private drive and a source directory on a shared drive, and that the laptop and PC both have the work directory and the source directory on a local drive.

### File Relations

To avoid having to transfer the entire source to the mainframe every time you want to compile it, the SURE software uses a match and merge method to minimize the communication overhead.

A local copy of the mainframe source is kept at the PC, while editing is done on a separate work file on the PC. When a compilation is started, this work file is matched to the local source, and the resulting match file is transferred to the mainframe, where it is merged into the source available there.



005246

### File Kinds and Extensions

Each source file kind has a preferred PC file extension associated with it. These extensions make it simpler, for both user and software, to see which language is used in a source file. SURE uses the preferred extension to determine the source filekind, and vice versa. You cannot override these defaults.

## Local Source Editing on PC

---

ClearPath Server FILEKIND	PC File Extension
Default	XXX
ALGOL	ALG
BACKUPPRINTER	DOC
BASIC	BAS
BINDER	BND
COBOL74	C74
COBOL85	C85
COBOL	CBL
CC	CC
DCALGOL	DCA
DMALGOL	DMA
ESPOL	ESP
FORTTRAN77	F77
FORTTRAN	FTN
JOB	JOB
NDLII	NDL
NEWP	NWP
PASCAL	P
PL1	PL1
RPG	RPG
SANS	SAN
SEQDATA	SEQ
SORT	SRT
TEXTDATA	TXT
XFORTRAN	XFT

### File names on the PC

SURE uses the following naming standard for the local work file and the local source file:

- All slashes in the name are replaced by back-slashes.
- The total name is preceded by the name of the work or source directory and completed with a PC file extension.

### Example

*Consider the cobol85 source PROG/ABC/001.*

*The name of the local work directory is C:\CPNX\DEVELOP\WORK\*

*The name of the local source directory is P:\CPNX\DEVELOP\SOURCE\*

*The name of the local work file is C:\CPNX\DEVELOP\WORK\PROG\ABC\001.C85*

*The name of the local source file is*

*P:\CPNX\DEVELOP\SOURCE\PROG\ABC\001.C85*

So, the directory structure of the file is maintained on the PC.

### New/Checkout

SURE creates a local work file in the work directory and a local source file in the source directory.

The local source file is not user accessible, and is only used by SURE and Compile for the matching process. If the local source file already exists, because of a previous editing session, then SURE checks if that copy is the same as the one last saved. If so, it copies this local source as the work file, without having to transfer it from the mainframe.

The developer edits the local work file.

### Check-In

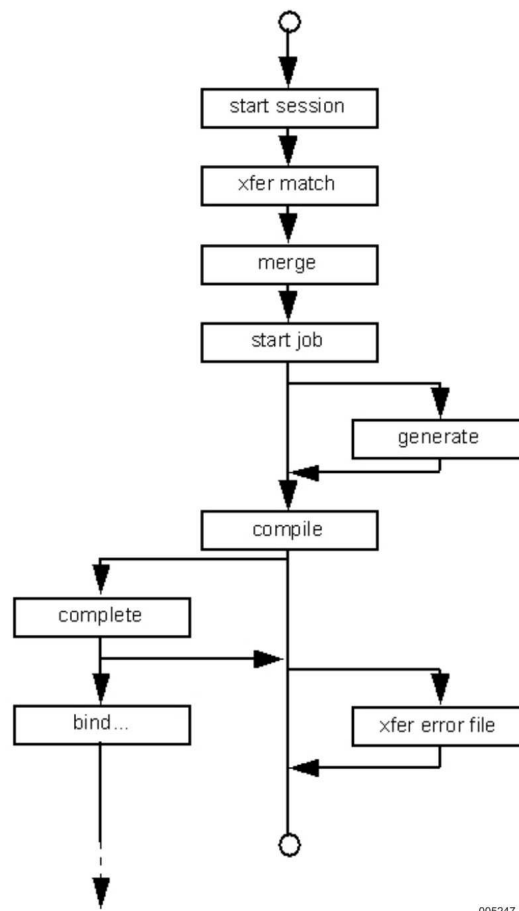
SURE matches the local work files and the local source files and transfers the result to the mainframe where it is merged into the remote source file. The merged source is then saved in the repository.

Then the work file is copied as the local source, thus bringing the local source up to date with the source in the repository. The work file is removed, to indicate that it is no longer in maintenance.

## 11.1. Compile MCP Files Using the PC

With this utility, source files that have been copied to the PC for local editing can be compiled for testing at the mainframe.

A compilation comprises a number of phases that are executed depending on the exact conditions of the compilation.



005247

Given a source file, Compile first calls the mainframe to acquire a session number. Then, Compile requests a FileTransfer of the work file. The FileTransfer matches the work file to the local source, and sends the result to the mainframe, where it is merged into the copy of the source available there. On completion of this process, Compile queues the compilation at the mainframe in a system queue.

The compile job performs all the necessary and requested phases for this source file. On completion, if errors have occurred, the compiler error file is transferred back to the PC.

If copy or include files are used in the source that is compiled, and these copy or include files are checked out, SURE will ask to upload these files.



### The Matching Process

In this phase, the PC work file is analyzed, and some conversion may take place to ensure the file has a correct source format for its file kind.

- Changed lines are automatically numbered including moved or copied blocks.
- If you want to retain your own inserted sequence numbers, check in the file with the “send complete work file” option set.
- Lines in the sources that are too long are split at the record length determined by the source file kind.

These alterations are written back into the work file to ensure that compilation errors can be bound to the correct source lines.

### Compile Request

The compilation process supports the following modes:

- Compile a file that is checked out with Local Edit set to TRUE.
- Compile a file that is checked out with Local Edit set to FALSE.
- Compile a file from the repository that is not checked out.
- Compile a file in the work-environment of the ClearPath server.
- Compile a file from the defined download directory on the PC.

### Configuration

Paragraph [COMPILE] of the `AW_OBJ.INI` file is used to configure the Compile process.

Entry	Usage	Default
MinimumIncrement	Minimal line number increment that is applied to unnumbered source lines in the matching process	1
MaximumIncrement	Maximal line number increment	2000
AutoModeTimer	Interval between two Status screen updates, in milliseconds	10000

#### 11.1.1. Compiling from a Windows Application: AS\_COMP

A compile session, with all defaults assumed, can be started in Windows by running AS\_COMP utility with the name of the work file as its parameter.

AS\_COMP signals SURE to start a new compile session and waits until the session is finished. Multiple instances of AS\_COMP can be active at one time, each controlling a separate compile session.

```
Usage: AS_COMP [/A] [/U] [/S<Server Name>] [/I<Ini File>] [/B] <source file>
/A = Automatic mode, do not show option screen
/U = Unattended mode, DS job if it is waiting for any file
/S = Server which is used for communicating
/I = INI file which is used for statistics
/B = Silent mode, which runs in the background
```

The /S option is used if multiple SURE sessions are open against multiple different servers. The SURE sessions should hold a SERVICENAME=<Server> in the GLOBAL section. The AS\_COMP utility can then select one of the running servers. If SERVICENAME is not entered in the SURE definition, this option should be omitted.

### 11.1.2. Merging Error Files

The AS\_RUN utility can perform a number of Windows functions from any DOS application. Mostly, this utility is used to convert error files to Visual C++ error files, so that any editor can support the syntax merge process.

```
Usage: AS_RUN [/n] [/c] [/r:nnn] <source file>
/n = do not perform compilation, just display errors
/c = convert error file to C-compiler message format
/r:nnn:nnn = return value <nnnn> if run OK, <nnn> if errors
/n
```

This option can be used to short-circuit the compile process. The actual compilation will not be performed, but the most recently generated error file will be displayed in the desired mode.

/c

Many editors require error messages to have a special formatting to be able to browse through the errors in the source. This switch will convert the error messages from the ClearPath server compiler formats to a format that closely matches that of the Microsoft C-compiler(s). Most editors will recognize this format.

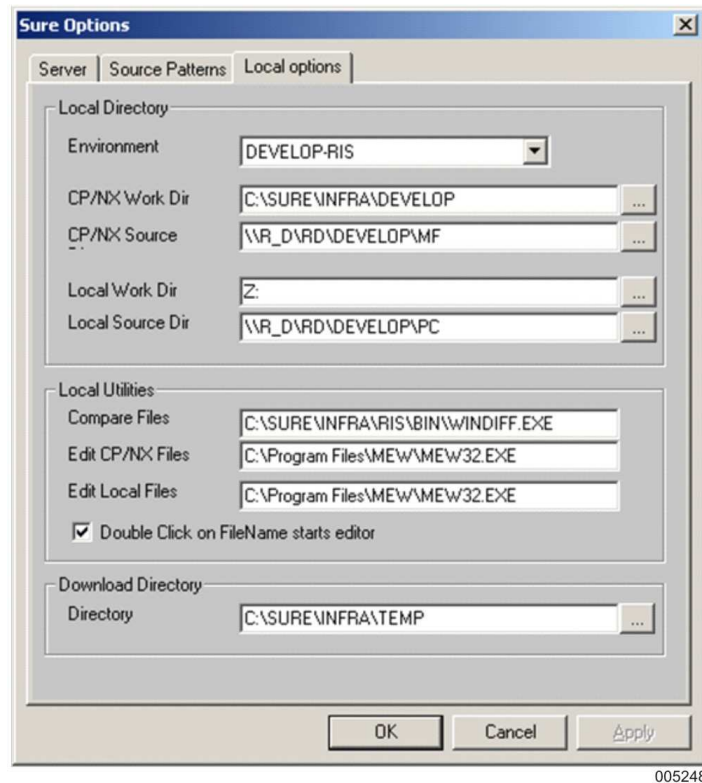
/r

Some utilities require a compiler to have a zero return value (DOS error level) if the compilation was completed successfully, others require a non-zero return value. The first numeric part of this option determines the value that will be returned if AS\_RUN utility has run without errors (independent of the presence of syntax errors or warnings in the source, which is determined by the contents of the error file). The second numeric part determines the return value if AS\_RUN could not complete correctly.

## 11.2. Editor Support

Support has been added and tested for a number of editors and programming environments.

The name of the local editor must be defined on the Options menu, Sure options window, Local options tab.



The “Edit CP/NX files” field defines the local editor for ClearPath server files.

Each of these environments requires a different approach to use Local Source Editing. Below, the supported environments are described in detail. For editors or environments not mentioned below, the following guidelines are usually applicable:

1. If a Windows program can be run from within the environment, use the command:

```
AS_COMP \workdir\source-name
```

2. If an error tracker utility that supports visual C++ is present, use the command:

```
AS_RUN /N /C /R:0:1 \workdir\source-name
```

3. If a MAKE file can be built, use a production rule, similar to:

```
\workdir\source.C74: \workdir\source.ERR
AS_RUN /N /C /R:0:1 \workdir\source.C74
```

### 11.2.1. ED for Windows

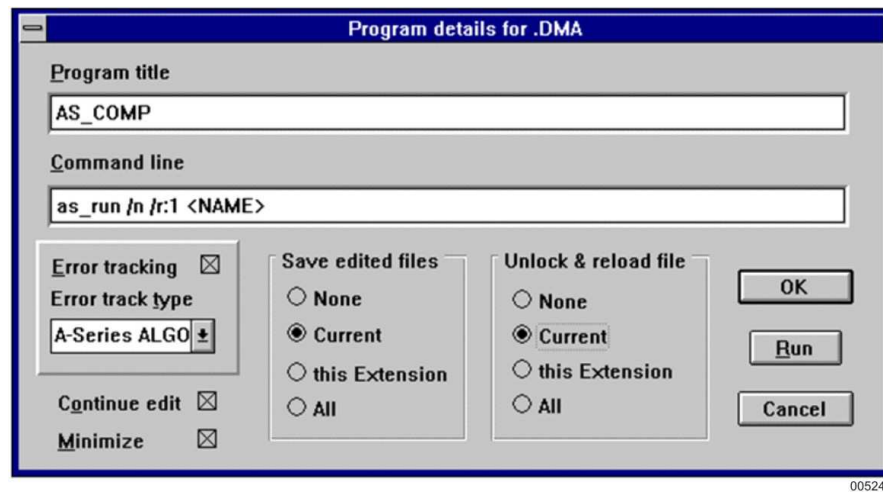
In ED for Windows, AS\_RUN can be used directly, as well as from a MAKE file.

Error trackers are available for ClearPath server ALGOL (ALGOL, DMALGOL, and DCALGOL) and COBOL (COBOL, COBOL74, and COBOL85). To use these error trackers, make sure the files `FEXT_PRG.CES` and `PRG_ERRS.CES` are placed in the ED for Windows main directory.

To use AS\_RUN directly, do one of the following:

- If you want to use the COBOL or ALGOL error tracker, do the following:
  - Open a file with the correct file extension (for example, C74 for COBOL-74).
  - Under Tool menu, select **Programs**.
  - Select **Extension specific**.
  - Click **Add**.
  - Enter a title (for example, "AS\_RUN").
  - As command line, enter: `AS_RUN /R:1:-1 <NAME>`.
  - Select **Error tracking**.
  - In the "Error track type" list box, select.
  - "CP/NX Series COBOL" or "CP/NX Series ALGOL."
- If you want to use the C error message conversion:
  - Under Tool menu, select **Programs**.
  - Select **General**.
  - Click **Add**.
  - Enter a title (for example, "AS\_RUN").
  - As command line, enter: `AS_RUN /R:1:-1 /C <NAME>`.
  - Choose **Error tracking**.
  - In the "Error track type" list box, choose **MSC, MASM**.
- Choose **Continue edit** and **Minimize**.
- Under Save edited files, choose **Current**(or "All").
- Under Unlock & reload file, choose **Current** (or "All").
- Click **OK**.

ED for Windows Tool/Programs example.



To run this tool, open the appropriate source, and choose "Tool/Programs/Run."

- To use AS\_RUN from a MAKE file:
  - Enter a production rule for the source file, similar to:

```
\sourcedir\MYSOURCE.C74: \workdir\MYSOURCE.C74
AS_RUN /R:0:1 /C \workdir\MYSOURCE.C74
```

This production rule checks the work source against the copy source.

- In ED for Windows, enter the MAKE command as a Tool, choosing "MSC, MASM" error tracking, as mentioned above.
- In "Save edited files" and "Unlock & reload file", choose "All," to ensure that all files mentioned in the MAKE file are available.

### 11.2.2. Microsoft Programmer's Workbench

Enter a production rule for the source file(s), similar to:

```
\sourcedir\MYSOURCE.C74: \workdir\MYSOURCE.C74
AS_RUN /R:0:1 /C \workdir\MYSOURCE.C74
```

A MAKE file with this structure is created after getting the file as a PC file by the SURE software called <file>.mak.

This production rule checks the work source against the copy source, created from SURE, or updated by the last compilation.

### 11.2.3. Unisys NX/Edit (Release 4.1)

The NX/Edit utility allows editing files on ClearPath servers. For this reason, local edit is disabled when defining NX/Edit as the editor.

The exact procedure is as follows:

- SURE function checkout copies the source from the repository to the correct workspace in CANDE and marks the file as "checked-out."
- SURE starts NX/Edit with the source name in the command line.
- NX/Edit presents a log-on screen where the user has to enter his usercode and password.
- NX/Edit continues after a correct log on, and loads the work file.
- The file is modified using NX/Edit.
- NX/Edit function SAVE saves the work file in the CANDE workspace.
- SURE function check-in loads the modified file in SURE and removes it from the workspace in CANDE.

The SURE software recognizes the use of NX/Edit by the executable name being NXEDIT.EXE.

#### 11.2.3.1. NX/EDIT Template Support

SURE now supports creation and usage of ClearPath Programmer's Workbench (NXEDIT) project files (.PWP). When enabled, SURE opens ClearPath server files for editing using a customizable project file.

##### **SURE Local Options**

To enable usage of PWP files, these files must be placed in a special location on the workstation.

This location can be configured on the Options menu, Sure options window, Local options tab. This tab now contains a field "CP/NX Project Dir," where you can enter a directory to store the PWP files for the selected environment. This is equivalent to entering the INI file key

```
[ <environment> ]  
PROJECTDIRECTORY=<location>
```

The naming of a PWP file associated with a ClearPath server file must follow the SURE standard mapping of ClearPath server file names on the Windows platform, for example, ClearPath server file

MY/PROGRAM

with ClearPath server Project Dir

C:\DEVELOP\Project

assumes its PWP file to be

C:\Develop\Project\My\Program.PWP

When custom made PWP files are manually placed in the appropriate directories, SURE will use these files to start an editing session.

### Template File

You can define a template PWP file to create a standard project for every ClearPath server file that is being edited.

This file has the same format as a PWP file, but can contain a number of mnemonics, on copying the template file to the project location, will be replaced with the matching properties of the file to be edited.

The available mnemonics are:

{FILENAME}	ClearPath server file name.
{FILEKIND}	ClearPath server file kind, for example, COBOL74SYMBOL.
{FILETYPE}	SURE file type.
{ENVIRONMENT}	The current environment.
{FILEVERSION}	Current file version.
{SYSTEM}	The file's system name.
{PROJECT}	The file's project name.
{USER}	The user code where the file resides.
{PACK}	The pack name where the file resides.
{HOST}	The host name where the file resides.
{OBJNAME}	The name of the compiled object.
{OBJUSER}	The user code where the compiled object resides.
{OBJPACK}	The pack name where the compiled object resides.
{OBJHOST}	The host name where the compiled object resides.

For example, the template file contains a line

```
[ FILES ]  
SYMBOL=?? {HOST} / {PACK} / ( {USER} ) / {FILENAME}
```

This line is expanded to

```
SYMBOL=??MYHOST/PROGPACK/(MYUSER)/MY/PROGRAM
```

which loads the work file from its correct location when the Programmer's Workbench is started.

The template file to use can be configured in the INI file:

```
[NXEDIT]
Template=<filename>
```

**Note:** Only one template can be defined.

### Order of Editing Actions

Depending on the entered settings, SURE will go through the following steps to determine what to do with an Edit request, assuming the ClearPath Programmer's Workbench (NXEDIT) is defined as the editor for ClearPath server files:

1. If a project location is defined but no template is configured, SURE will search for the project file associated with the file to be edited. If found, the project file is opened. If not found, SURE will start NXEDIT without a project.
2. If a project location is defined and a template is configured, SURE will search for the correct project file. If this file is not found, it is created using the template file. Subsequently, the (new or existing) project file is opened.
3. If a template is configured, but no project location is defined, SURE will create a temporary project file from the template file and open this project file. The file will remain in the TEMP directory.

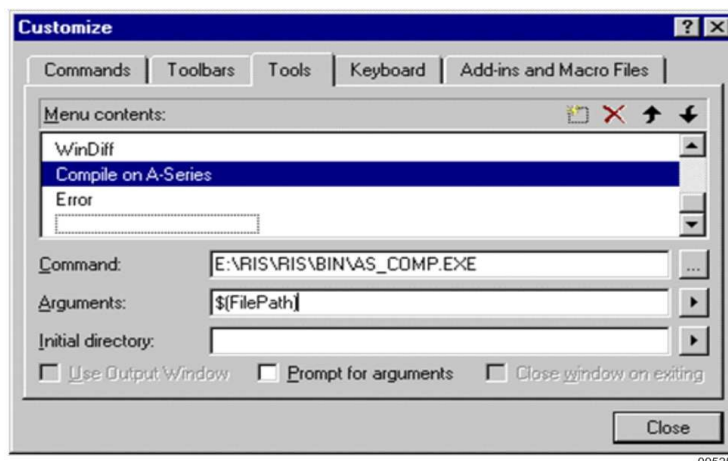
**Note:** SURE will never overwrite an existing project file, allowing manual project files to be created, and allowing generated project files to be manually changed after their creation.

### 11.2.4. Microsoft Visual Studio C++ (Release 6.0)

The Microsoft Visual studio C++ environment allows compiling and error file merging using the tool menu.

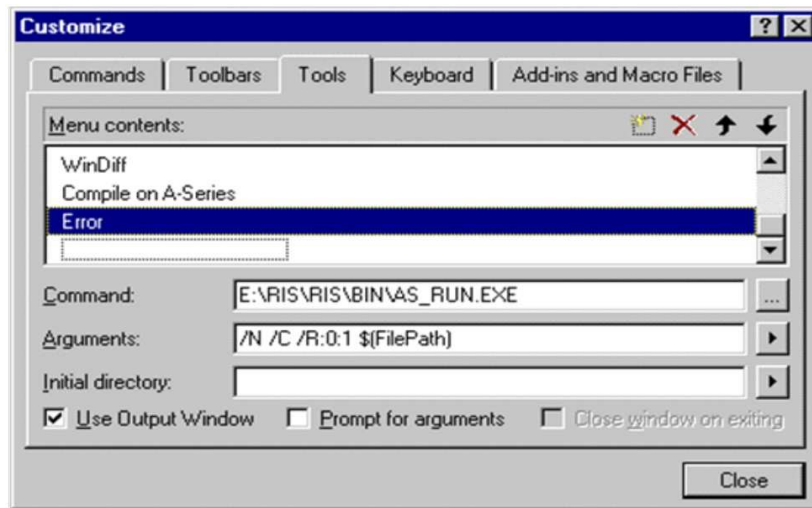
In Visual Studio you define tools using the menu choice Tools or Customize and thereafter selecting the tab sheet Tools.

Define the Compile tool as the following window snap shot.



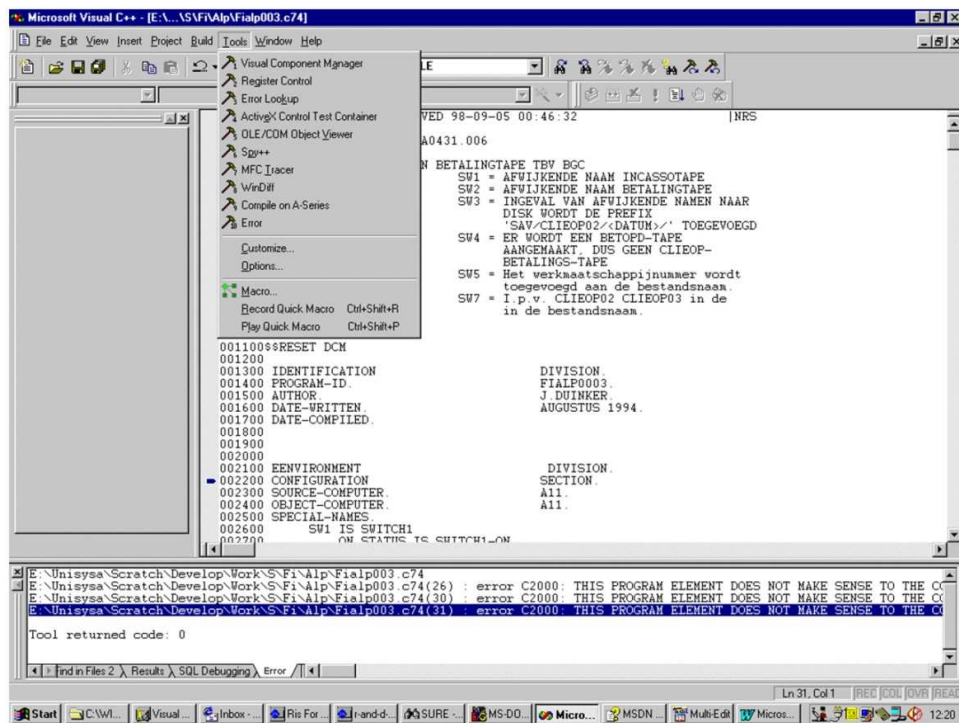


Define the Error tool as the following window snapshot.



005251

Hereafter you are able to compile and merge error files from the tools menu as the following example shows. The errors are shown in the output window and double click on them will position the file to the correct line.



005252

If Visual Studio (MSDEV.EXE) is started with a command line parameter, it will open a file in a separate instance. Therefore, each time you do edit from SURE, a new copy of MSDEV is started. A way to avoid this is defining a workspace in MSDEV, which contains a group of connected files.

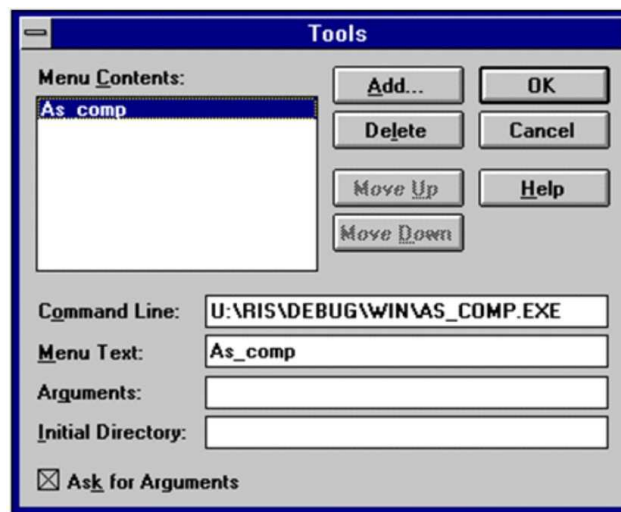
### 11.2.5. Microsoft Visual C++ (until Release 4.2)

The Microsoft Visual C++ environment does not cooperate nicely with Unixcorn. Therefore, AS\_RUN cannot be used to start AS\_COMP. However, it can still be used to display or convert the error messages.

First, enter AS\_COMP as a tool:

- From Options menu, select Tools.
- Click **Add**.
- Choose "AS\_COMP.EXE" from the SURE executables directory as file name.
- If you want a separate tool entry for every source file:
  - Enter a "Menu Text" containing the source name.
  - Enter the source name as "Argument".
- If you want one tool for all source files:
  - Choose "Ask for Arguments".

#### MS Visual C++ Option/Tools Example



005253

Then, enter AS\_RUN as error converter in the MAKE file, using a production rule similar to:

```
MYSOURCE.C74: MYSOURCE.ERR
    AS_RUN /N /C /R:0:1 MYSOURCE.C74
```

A MAKE file with this structure is created by the SURE software after getting this file called <file>.mak.

This rule will check the work source against the error file, showing the error file only if it has been created by the AS\_COMP session.

If a source has to be recompiled, first run the appropriate tool, and where necessary enter the source name as an argument. This can be done simultaneously for all the sources involved.

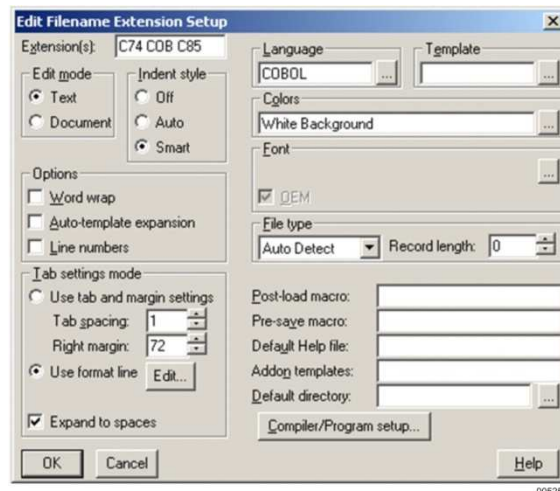
Make sure that the work files are unlocked by `SHARE`, before starting a compilation, since the compile process will rewrite the copy source and the work file.

Wait for (all) the `AS_COMP` session(s) to terminate, and then build the `MAKE` file.

### 11.2.6. MultiEdit

From Tools menu, select Customize, and click `FileNameExtensions`. A definition must be created for every supported extension, such as `ALG` (ALGOL), `C74` (Cobol74), and `C85` (Cobol85).

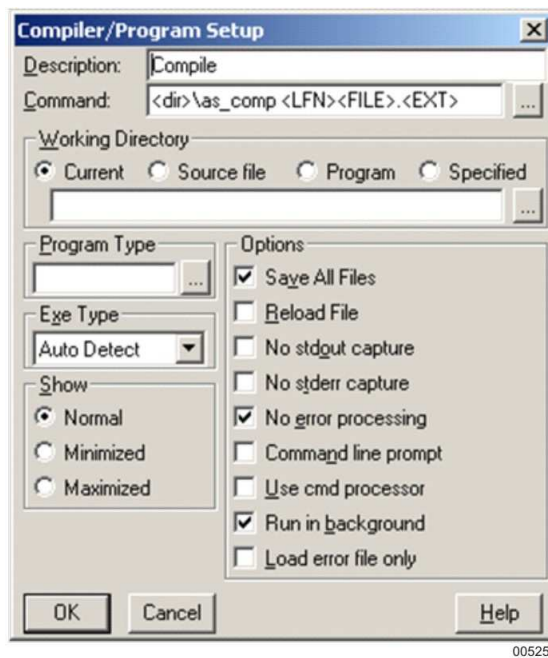
The first screen defines the general editor layout and language support (Language support and tab spacing is adjusted).



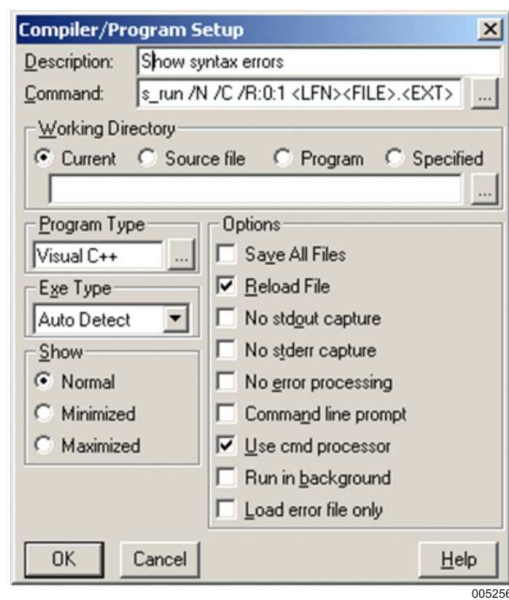
**Note:** This definition supports the extensions `COB`, `C74` and `C85` (COBOL, Cobol74, and Cobol85). ALGOL-programs require an ALGOL-specific definition.

From Compiler/Program Setup window, two utilities have to be added, one for the compile and one for the syntax merging process.

## Local Source Editing on PC



Command = <SURE installation directory>\ris\bin\as\_comp  
<LFN><FILE>.<EXT>



Command = <SURE installation directory>\ris\bin\as\_run /N /C /R:0:1  
<LFN><FILE>.<EXT>

### 11.2.7. SCC Interface

SURE supports the standard SCC (Source Code Control) interface. Many IDEs (Interactive Developers Environment) use the SCC interface to create a direct connection to a source control system. This interface offers the following source control functions in the PC-editor:

- Add
- Check-out
- Check-in
- Undo check-out
- Get latest version

SURE requires that each source is linked to a project and a file type. An extra requirement is that each source must be linked to a task when that source is added or modified.

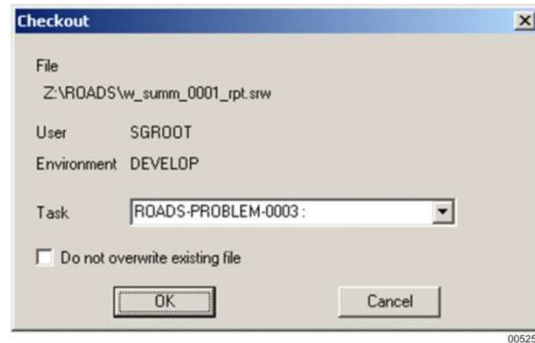
SURE presents the SURE-SCC-add-dialog when a source is added using the SCC-interface:

- Each file in SURE must be linked to a system, the files in this example are linked to system ROADS.
- A file can only be loaded or changed because of a task. The current task in this example is ROADS-PROBLEM-0002.
- Each file in SURE is categorized with a file type. The file type in this example is POWERBUILDER.



SURE gives the SURE-SCC-checkout dialog when a source is checked out using the SCC-interface:

- A file can only be loaded or changed because of a task. The current task in this example is ROADS-PROBLEM-0003



If multiple files are checked out or added to SURE during the same IDE session, then the project, task, and file type that were used previously are automatically pre-entered on the above screens. The developer can verify and change them (if necessary) before he clicks OK.

For some editors, the connection between the PC-editor and SURE must be established by defining SURE as the source control system for that editor.

### Examples

#### Macro Media

No special actions required.

If you do not want the standard SURE dialogs, you can set the silent mode in the INI file using the next lines in the SURE INI file (by default AW\_OBJ.INI)

```
[Macro Media]
SILENT=*. *
```

#### Visual Studio 6.0

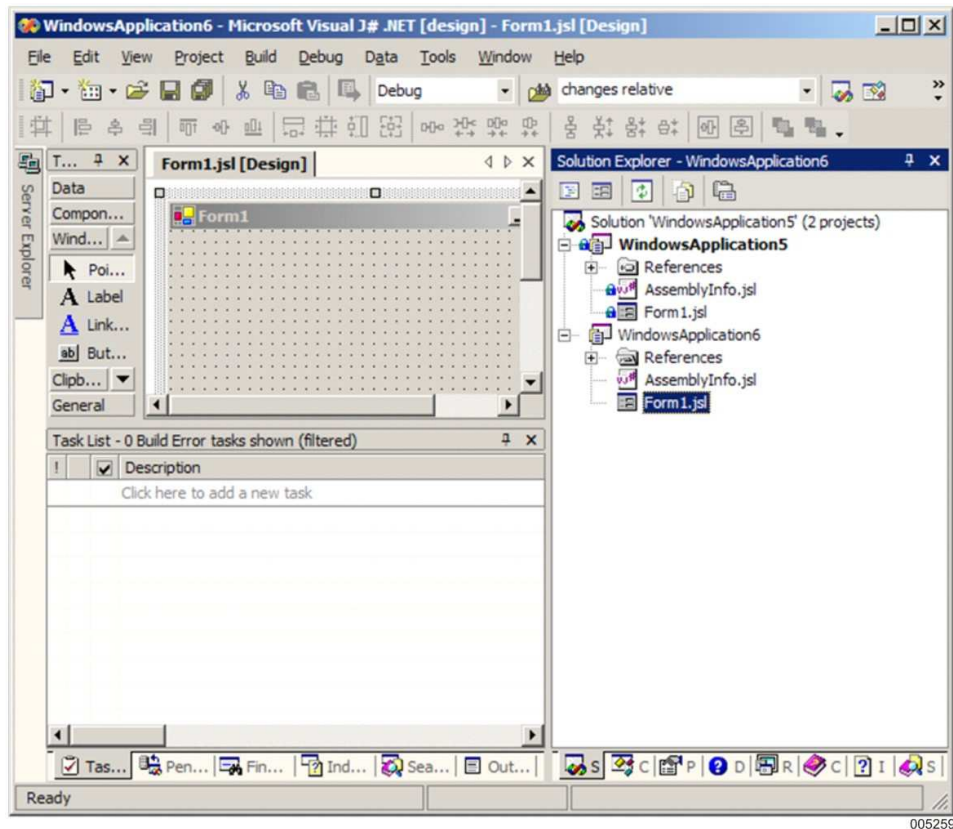
If you do not want the standard SURE dialogs, you can set the silent mode in the INI file using the next lines in the SURE INI file (by default AW\_OBJ.INI)

```
[Microsoft Visual C++]
SILENT=*. *
[Microsoft Visual Basic]
SILENT=*. *
```

## Visual Studio .NET

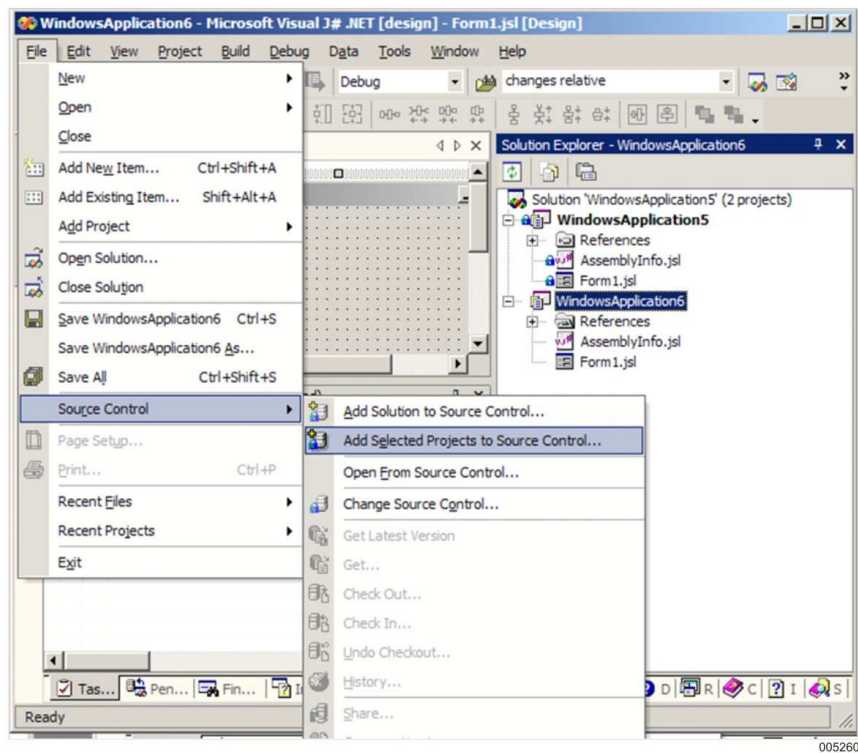
### Add Selected Projects to Source Control

Creating a new project in the Microsoft IDE does not automatically add the project to the source control provider. The following screen shows a project controlled (WindowsApplication5) and a project not controlled (WindowsApplication6).

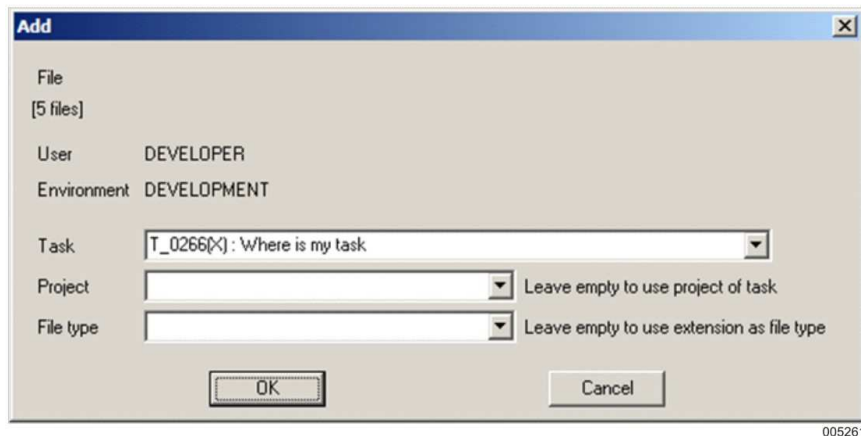


To add WindowsApplication6 to the SURE, from File menu select Source Control and click Add. The selected projects are added to the Source Control as shown by the next screen example.

## Local Source Editing on PC



The SURE interface shows the next screen to complete the action.



**Note:** If the file type field is left blank, all extensions of the loaded files need to be defined in SURE as a file type in capitals.

The Microsoft Visual Studio .NET solution and project files do not easily change from source control provider. It requires modification of these solution and project files to allow for another source control provider than Visual SourceSafe. For this reason, a small utility is created which converts the project files, so that SURE can be used as source control provider. This utility can be found under Tools menu on the PC Environment option.



**Note:** *The project or solution file needs to be checked out in order to run the utility, because this file will be altered.*

If you do not want the standard SURE dialogs, you can set the silent mode in the INI file using the next lines in the SURE INI file (by default AW\_OBJ.INI)

```
[VSSCC]
SILENT=*, *
```

### Micro Focus Net Express

Micro Focus Net Express automatically links to the installed SCC provider. No manual actions are required.

Net Express 4.0 has some characteristics in handling its SCC calls that a developer should be aware of.

- The IDE does not automatically refresh the source control status of a project when it is loaded; instead, it shows the status as it was when the project was last closed. It is recommended to issue the command Source Control or Refresh Status before issuing any source control commands on items in the project.
- If a command Add to Source Control fails, the IDE will still show the file to be "Sccs controlled." A Refresh Status command is required to correct this.
- When selecting a file and issuing a source control command, the IDE will show an intermediate dialog, in which files can be selected or deselected, before actually issuing the command. If the selection of files is changed in this dialog, the IDE will show an error message when returning from the source control command. This error "Access violation ... MSVCRT.dll" is not fatal, and after choosing "Continue" in this dialog, the IDE will continue without problems.



## Section 12

# Windows or UNIX Files in SURE

It is possible to load Windows or UNIX files in the SURE repository. SURE is used for archiving files and file transfer integrated with a task registration system. In this aspect, SURE does not handle Windows or UNIX files different from MCP Series files.

For this reason, transferring a task may transfer MCP Series files, PC source files, PC executables or PC Word document files, UNIX source files, UNIX script files, and so on.

In the case of MCP sources, transfer of files compiles implicitly the sources and deploys the objects to the run-time environment.

For Windows or UNIX files this works slightly different, because the build mechanism and the distribution mechanism for Windows or UNIX files differs. If a PC file is checked in or transferred, then it is added to the PC build queue of the corresponding environment. The SURE build support starts user-defined scripts that create the executable files (EXE, COM, DLL, and so on). An extra option is to store the created executables in the repository, so that they can be distributed by SURE at a later stage.

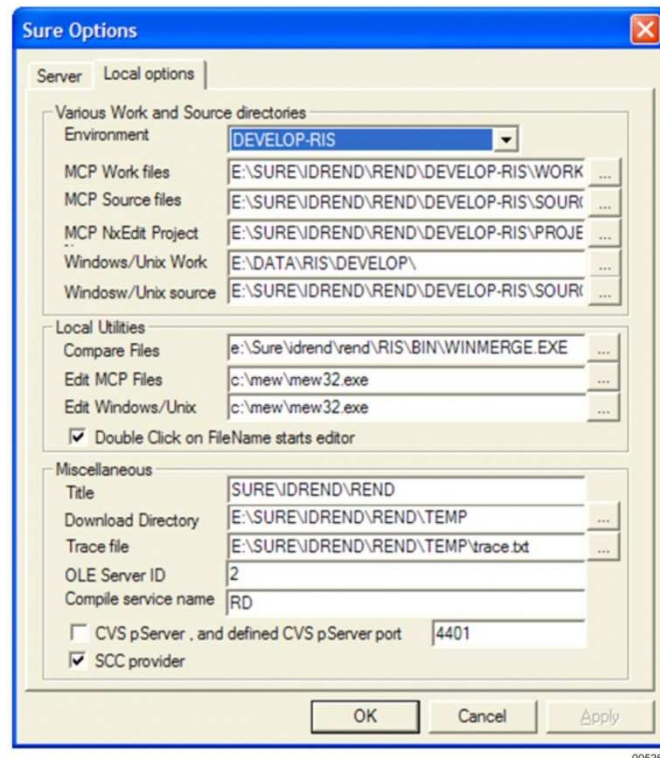
Windows or UNIX files and MCP Series files are shown in two different main folders.

Note that the SURE explorer runs on a Windows client. Therefore, a Samba software is needed when handling UNIX files.

### 12.1. (Windows or UNIX Files) Configuration

For every environment, a work and a source directory is defined through:

1. Select **Options**.
2. Select **SURE options**.
3. Click the Tab: **Local Options**.



The “Windows or UNIX Work Dir” and “Windows or UNIX Source Dir” fields are used to identify the work and source directory for Windows or UNIX files.

### Entry

### Usage

PCWorkDirectory

Directory to store the Windows or UNIX work file

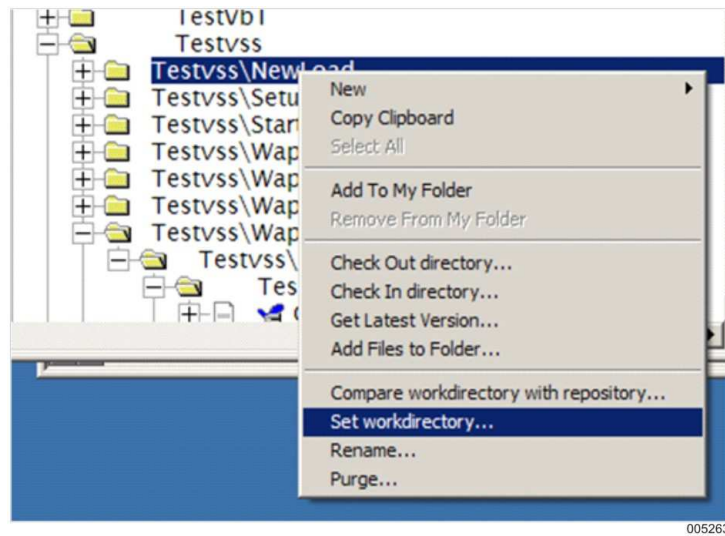
PCSourceDirectory

Directory to store the Windows or UNIX local source which is only used for delta file processing

By default, there is a single work-directory for all PC files. This means that all files that are viewed or checked out have the same base directory. In the above example screen the default base directory is E:\DATA\RIS\DEVELOP\.

One work-directory for all PC files may not be workable in all situations. There might be cases that (for example) documentation is located in My Documents and that developed web services are located in directories differently than Visual Basic programs. For those situations, SURE allows configuring different work-directories according to SURE folder. Right-click a SURE PC(files) folder and select Set Work Directory that allows setting a work and source directory for this specific SURE folder.

**Note:** The PC files, the SOURCE directory is only used for file compare functionality. Therefore, it is not useful to define different SOURCE directories.



## 12.2. (Windows or UNIX Files) File Name

The file name in SURE contains the full PC file name excluding the work directory. With this full name, the files are stored in the repository and temporary versions may exist on the MCP Series disk.

The following table shows some examples of Windows or UNIX files and the corresponding file name in SURE. The name in SURE depends on the name of the work directory.

PC File Name	Work Directory	SURE File Name
C:\WORK\TEST\MYF.TXT	C:	WORK\TEST\MYF.TXT
C:\WORK\TEST\MYF.TXT	C:\WORK	TEST\MYF.TXT
C:\WORK\TEST\MYF.TXT	C:\WORK\TEST	MYF.TXT

The file name in SURE is the PC file name minus the work-directory.

**Note:** *You can only access files from this work directory.*

The NAMESTANDARD routine also applies to Windows or UNIX files. The path should be present in the name standard syntax. Using this function may construct PC file names according to defined name standards.

The files are entered in SURE from the Edit menu, New, File function or from PC Tools, PC Environment, Load Files option in SURE. In both ways, the SURE server may adapt the case (upper or lower case) of the file name. If the directory part is already present, then this existing name is used. Furthermore, the name will be proposed in the Windows standard form (all first letters of a directory in capitals).

### Example

Consider the following file

```
C:\WORK\TEST\MYF.TXT
```

If the work-directory is	then the proposed SURE file name is
C:	Work\Test\Myf.txt
C:\WORK	Test\Myf.txt
C:\WORK\TEST	Myf.txt

**Note:** When loading files from a UNIX or LINUX system, the case should be preserved because file names are case-sensitive on these operating systems. Furthermore, the LF and CRLF differences for WINDOWS and UNIX or LINUX files do not impact on these files in SURE. The file contents are stored without any alterations. This means that opening a UNIX file in notepad results in a single long line, because there are no CR characters present. There are many PC-editors that run on the Windows platform and that can handle UNIX sources as well as Windows-PC-sources. An example of such an editor is Textpad ([www.textpad.com](http://www.textpad.com)).

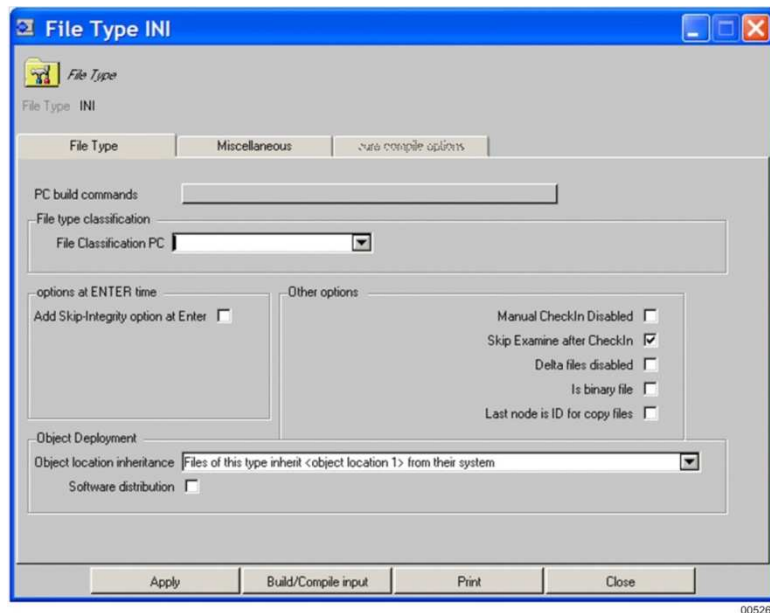
## 12.3. (Windows or UNIX Files) FILE-TYPE

Each file has a file-type. The file-type definition contains major attributes for SURE in handling a file. The behavior of SURE depends on the file-type of a file.

For Windows or UNIX files in general, a file-type is created for every extension that is loaded into the SURE repository. So, suppose that \*.html, \*.txt, \*.cpp, \*.doc, \*.xls, and \*.bat files are loaded, then the file-types HTML, TXT, CPP, DOC, XLS, and BAT should be defined. There may be circumstances that this rule does not apply; however, this is exceptional and initially SURE should be configured as advised.

**Note:** There is a difference between Windows or UNIX files and MCP files. Windows or UNIX files are mostly stream files and MCP files are mostly fixed record length blocked files.

Windows or UNIX files are separated in two major groups: "binary files," like bitmaps, MS-word documents, and spreadsheets and source-PC-files, like JAVA, C#, VB, C++ programs, or other sources. Another option configured in SURE is the indication "Delta files disabled." If disabled, you can only rollback to a version in another SURE environment. If not, you can rollback to any intermediate stage (checked in at a given time). For non-binary files, the delta files (diff files) are used to rollback the source. For binary files, the full file is archived in the repository and used for rollback purposes.



005264

### Skip Examine after CheckIn

After a source is checked in, it is examined by SURE. SURE scans the source for references to other files, for example "include" statements to copy files. This feature makes only sense for source-files. It does not make sense for binary files, such as word documents; excel spreadsheets, and bitmaps, because the syntax-checker does not recognize these kinds of files. Therefore, this option should be set for binary files.

### Is binary file

After a source is checked in, it is compared with its previous version that was loaded in SURE. The differences are stored in SURE in a delta file. This feature makes only sense for source-files, because the DIFF algorithm works only on these files. In case of a binary file, the full old-version of the file is archived in the repository and used for rollback purposes.

### Delta files disabled

At every check-in time, a delta file is stored in the repository when this option is reset. For binary files, the delta file contains the complete previous version. For non-binary files, the delta file is a constructed diff file. These delta files are used for compare utilities and rollback functions.

### Last node is ID for copy files

After a source is checked in, it is examined by SURE. SURE scans the source for references to other files, for example "include" statements to copy files.

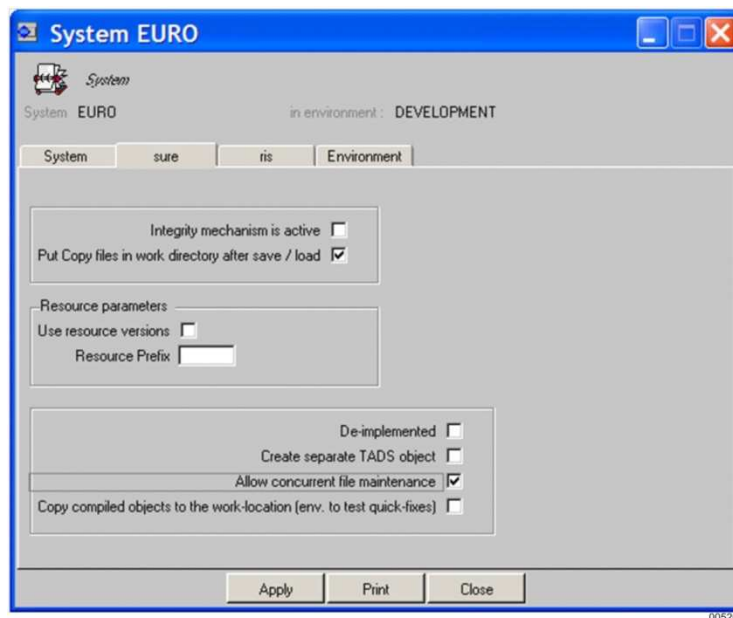
The <last node is ID for copy files> option handles the case that a COPY or INCLUDE statement in the PC source uses only the last node. This last node is used as a key to find the full name of the include file. This option is often used for include files, because the PC compiler is often given a path name in which the copy files are present.

## 12.4. Concurrent Source Maintenance

The SURE software supports concurrent source maintenance for Windows or UNIX files. This means that multiple developers may change files at the same time in their work-environment. The SURE software integrates these changes at check-in time.

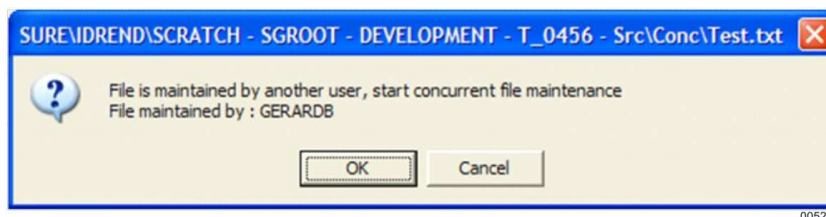
**Note:** *Concurrent source maintenance requires time and effort when changes overlap. In this case, the developer needs to make the final decisions for the resulting code. The SURE software can only assist and show the conflicts.*

The SYSTEM definition contains a check box that enables concurrent file maintenance for all files within a system.



If the concurrent file maintenance option is set at SYSTEM level, a file may be checked out by multiple developers at the same time and at check-in time SURE will merge the changes. The check-out and check-in processes are described for these conditions.

When checking out a file that is already checked out by another developer, the following warning is issued.



When the file is modified, and the file is checked in after that the other developer checked in the file, SURE will merge the changes. Note that in this condition, the file in the repository contains changes that are not present in your file.



The first developer who does the check in gets a warning that someone else is also working on the same file. However, the file can be checked in without complications, because it is not changed in the repository yet.

The second developer who does the check in gets the Check In dialog with the concurrent file support buttons enabled.



By pressing the OK button, SURE will merge the changes and save the file automatically. The file to be saved can be previewed using the button "Preview merge."

If SURE detects conflicts, an error dialog is given and the developer needs to modify the file manually. The button "Apply merge and edit" lets SURE to merge the changes and allows the developer to modify the file afterwards.



## **12.5. (Windows or UNIX Files) Source Files**

The SURE repository can process any kind of PC source files. After a source is checked in it is scanned for references to other files, and a delta file is created (see previous paragraph). These functions only make sense for source files and not for binary files. Therefore, any PC source file (C-source (Pascal, Fortran, Delphi, Bat, and so on) should contain a file-type definition with options "Skip examine" = RESET, "Delta files disabled" = RESET, and "Is Binary file" = RESET.

If a file-type is defined like this, delta files are created after the file is checked in. Creation of the delta files happens on the PC through a DIFF algorithm. For this reason, saving a PC file results in two file transfers: the delta file and the source file.

The SURE user interface for Windows or UNIX files is similar to the user interface for MCP files. For this reason, most of the commands are equal for both files.

## **12.6. (Windows or UNIX Files) Detailed Description of Delta File**

A delta file is the result of a comparison of two successive source files. The delta files are stored in SURE and are used for inquiry purposes but also to rebuild old PC FileVersions. Using the function "Compare" you can compare two PC FileVersions in a tool like WINDIFF. However, if you list the delta file, the native content is displayed. This delta file content is described here in detail.

The normal output format consists of one or more hunks of differences; each hunk shows one area where the files differ. Normal format hunks look like this:

```
CHANGE-COMMAND
< FROM-FILE-LINE
< FROM-FILE-LINE...
---
> TO-FILE-LINE
> TO-FILE-LINE...
```

There are three types of change commands. Each consists of a line number or comma-separated range of lines in the first file, a single character indicating the kind of change to make, and a line number or comma-separated range of lines in the second file. All line numbers are the original line numbers in each file. The types of change commands are:

### **`LaR'**

Add the lines in range R of the second file after line L of the first file. For example, "8a12,15" means append lines 12-15 of file 2 after line 8 of file 1; or, if changing file 2 into file 1, delete lines 12-15 of file 2.

**`FcT'**

Replace the lines in range F of the first file with lines in range T of the second file. This is like a combined add and delete, but more compact. For example, "5,7c8,10" means change lines 5-7 of file 1 to read as lines 8-10 of file 2; or, if changing file 2 into file 1, change lines 8-10 of file 2 to read as lines 5-7 of file 1.

**`RdL'**

Delete the lines in range R from the first file; line L is where they would have appeared in the second file had they not been deleted. For example, "5,7d3" means delete lines 5-7 of file 1; or, if changing file 2 into file 1, append lines 5-7 of file 1 after line 3 of file 2.

**Example of Normal Format****First Line**

```
This is the first version of this text.  
It shows the normal output of a DIFF command.  
The second version is shown below.  
Therefore, some enlightening differences exist  
between the two files.
```

**Second Line**

```
This is the second version of this text.  
It shows the normal output of a DIFF command.  
Therefore, some enlightening differences exist  
between the two files.  
This is one of them:  
some lines of text are added  
at the end of the file.
```

**Diff output**

```
1c1  
< This is the first version of this text.  
---  
> This is the second version of this text.  
3d2  
< The second version is shown below.  
5a5,7  
> This is one of them:  
> some lines of text are added  
> at the end of the file.
```

## 12.7. (Windows or UNIX Files) Binary Files

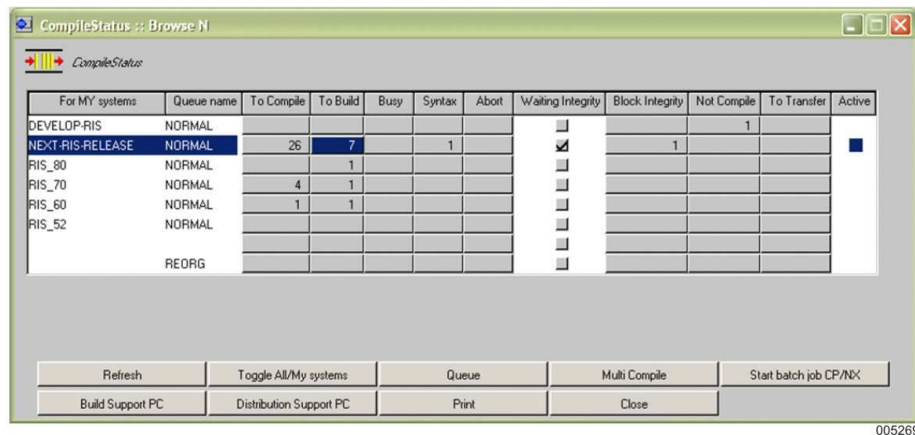
The SURE repository can process any kind of Windows or UNIX files including EXE or DLL files. For these binary files, no match functionality is required. The file-type definition for binary files should therefore set the options "Skip examine" and "is Binary file."

Binary files are stored in a compressed (ZIP) format in the repository. Compression is done at the PC at check-in and check-out time.

Setting the option "Delta files disabled" stores a single version of the file for each environment. If this option is reset, the previous versions remain in the repository until the "delete old deltafiles" condition is met.

SURE can be used for these files to synchronize file promotion, but some functions that apply to source files are non applicable to binary files.

## 12.8. (Windows or UNIX Files) Build Support



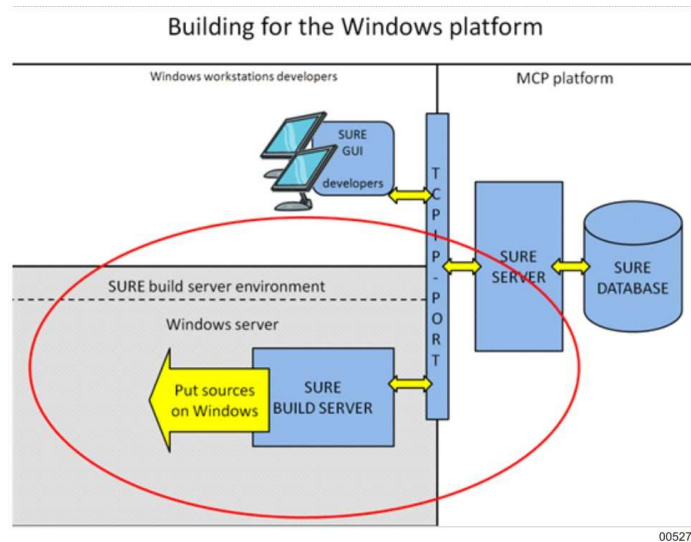
Source files that are changed in an environment are placed in the To-Compile queue (MCP-files) or in the To-Build queue (PC and UNIX files).

The Windows interface of the SURE software contains build support for PC executables. From Tools menu, select Compile Interface click Build Support PC button, a window is shown with two key fields: the Environment and the Server name.

The Environment selects the appropriate environment for which the build support is required, that is the environment for which you want to create executable files.

The server name defaults to the computer name on which the build process is running. This name is defined in the INI-file and must match with an Object-Server that is defined in the SURE repository. On System level, one can define a default Object-Server-name, which defines the logical build server where the changed PC sources of that system are downloaded to build new executables. This default Object-Server-name may be overwritten on file basis.

The following picture shows the SURE build server which must build software for the Windows platform. The SURE build server is a regular SUREforWindows GUI installation, which is afterwards configured as SURE build server. You can start it using a Windows bat-file and it runs unattended, the logon, the synchronization of the source files from the repository to the build environment, and the start of the build-scripts is all done automatically.



By default, the build support window shows all the files with `COMPILE-STATUS(TO-COMPILE)` and `OBJECT-HOST(<computer name>)`. Altering the environment and server name and pressing on REFRESH returns the queue for the requested environment and server.

**Note:** Each changed or promoted file is marked with a relation `COMPILE-STATUS(TO-COMPILE)`.

Pressing the COMPILE button performs the following two steps:

- Download all changed files that are linked to this logical build server.
- Start a build process for every target directory.

Initially, a directory can be downloaded to the build-server by selecting a PC folder in the proper environment, right-click and select download folder from the menu. Thereafter, updating the changed files through build support maintains the directory up-to-date.

The first step in the build process is controlled by the SYSTEM, PROJECT, FILE-TYPE, and FILE definition in SURE. Each of these objects may contain a BUILD CMD definition. The available definitions are combined by SURE to a file called <filename>\_BUILD.BAT, and executed on the Windows platform. So file <filename>\_BUILD.BAT consists of:

<SYSTEM>	build commands
<PROJECT>	build commands
<FILE-TYPE>	build commands
<FILE>	build commands

If either the FILE-TYPE or the FILE contains build commands, this first step is executed.

A build command consists of Windows BAT file commands. However, during the construction of this BAT file, a number of special constructs are resolved.

### Example

- The work directory is C:\WORK\
- Target file C:\WORK\PROJ\MYF.CPP has PROJECT(PROJ) in the SURE repository.
- Another file C:\WORK\PROJ\MY.DSP has FILE-TYPE(DSP) in the SURE repository.
- The following table shows the generic tokens that are supported, and how they are replaced when file PROJ\MYF.CPP is built.

Token	Is replaced by
<FILE>	MYF
<FILENAME>	MYF.CPP
<EXT>	CPP
<PATH>	PROJ\
<FULLNAME>	PROJ\MYF.CPP
<WORKDIR>	C:\WORK\
<FILE:FILE-TYPE(DSP)>	MY.DSP (this construct looks for a file in the same directory as the current-file, but with file-type = DSP. The result is MY.DSP)
<PROJECT>	PROJ
<ENVIRONMENT>	Current SURE environment

Token	Is replaced by
<SELECTFILES:selection-expression>	The statements in line-1 to line-n are repeated for all files that are selected by the selection expression.
line 1	File name-related generic build-tokens (<FILENAME> <FULLNAME> <PATH> <EXT> ) that are used between <SELECTFILES:...> and <END> are replaced by the corresponding value of the selected file. If these tokens are used outside <SELECTFILE:...> and <END>, then they are replaced by the corresponding value of the source for which the build script is executed.  The selection-expression searches files with a specific class(asset) in the same directory (including sub-directories) as the owner of the build-script.  Any selection-expression that can be entered on the query screen can be used in this command.
...	
line n	
<END>	

If a build file is found, it creates a file called BUILD.ERR in the same directory and executes the BAT file. The contents of the BAT file are freely defined and may contain constructs as shown in the previous table. Therefore, it can contain NMAKE utility or a build command to Visual Basic or Visual C++.

Thus, SURE synchronizes the WORK directories with the latest version of the source files, SURE starts and translates the build.bat files and SURE reacts on the existence of the build.err file.

The only feedback to SURE is that the BUILD.BAT file must remove the BUILD.ERR file if the build process was successful. Every file which is in the directory is marked with relation `COMPILE-STATUS ( COMPILED )` (if the BUILD.ERR file was removed) or with relation `COMPILE-STATUS ( SYNTAX )` (if the BUILD.ERR file is still present).

If an executable is defined for a file, for which a build command is processed, and the executable is created in the build process, and there is no BUILD.ERR file present, then this executable is loaded into the SURE repository.

**Note:** *If a BUILD.BAT file is used, the executable should be defined under this BUILD.BAT file. If a build command is used for a file, then the executable should be defined under this file.*

The second step in the build process consists of executing a BUILD.BAT file from the most global directory for every changed file. Therefore, if a file is changed in the directory C:\ASERIE\PROD\PC\FILE\TEST.CPP, the build support searches for a build file in the following order:

1. C:\ASERIE
2. C:\ASERIE\PROD
3. C:\ASERIE\PROD\PC
4. C:\ASERIE\PROD\PC\FILE

If there is no BUILD.ERR file in any of the directories, the files are marked with `COMPILE-STATUS (COMPILED)`.

### Example of a BUILD.BAT

```
set WORKDIR=<WORKDIR>

cd /d "%WORKDIR%\<PATH>"
set ERRLOG=build.err
set prev_path=%path%

call %WORKDIR%\env\Builder.bat
call %WORKDIR%\env\utility.bat

set Config=Release

set build=build <PATH><FILE:FILE-TYPE(SLN)>.SLN

echo start %build%>%errlog%
echo Builds with %BUILDER%>>%errlog%

call %WORKDIR%\env\buildsln.bat <FILE:FILE-TYPE(SLN)>.SLN %Config%
IF ERRORLEVEL 1 (
    echo %build% failed>>%errlog%
    echo %build% failed
) else (
    del %errlog%
)
set path=%prev_path%
```

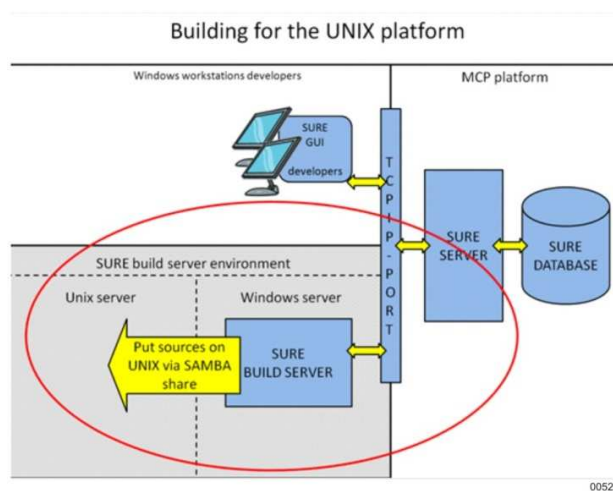
**Note:** The generic tokens that are used in this script are `<WORKDIR>`, `<PATH>`, and `<FILE:FILE-TYPE(SLN)>`. These tokens are replaced by the real names just before the build script is started.



### 12.8.1. (Windows or UNIX Files) Build Support UNIX Extensions

A slightly altered setup of the build command layout allows easy access to software build servers running on a UNIX or LINUX platform. The current implementation supports UNIX in general as build platform.

The following picture shows the SURE build server which must build software for the UNIX platform. The SURE build server is a regular SUREforWindows GUI installation, which is afterwards configured as SURE build server. You can start it using a Windows bat-file and it runs unattended: the logon, the synchronization of the source files from the repository to the build environment on the UNIX platform, and the start of the build-scripts on the UNIX platform is all done automatically.



To use a build server running UNIX as software build server for sources stored in SURE, a number of steps need to be taken care. In the following text, these steps are explained through an example installation session using generalized names and commands.

#### Network Configuration

Most UNIX systems include software to give access to the file system through network protocols that are also supported by Windows. These protocol daemons may need to be activated or configured to allow the Windows build server access to the UNIX environment.

On the UNIX server named "unixserver," assign a user name "myuser" that runs the build commands to create the software release.

Create a link, named "sources" in the user's home directory that links to the work directory where the sources for the selected environment (for example, Production) physically reside as follows:

```
ln -s /central/appsdire/Production sources
```

**Note:** A separate central source directory must be created for each SURE environment on which you want to use this feature.

On the PC running the SURE build server, create a drive mapping X that links to the home directory of the user:

```
Net use X: \\unixserver\myuser
```

In the AW\_OBJ.INI file, set the work directory for the appropriate environment to the linked directory using the drive mapping, for example:

```
[DEVELOP]
PCWORKDIRECTORY=X:\sources
```

**Note:** UNIX is case-sensitive while Windows is not, therefore the path entered here must contain the linked directory name in the exact case it is created with, or the build commands executed at the UNIX host will not be able to execute.

In the same INI section, place the indication that the build target is UNIX, for example:

```
[DEVELOP]
BUILDTARGET=UNIX
```

This setting ensures that the build command file is created in UNIX layout, using LF as line separator, instead of PC layout, using CR+LF as line separators. Also, file names that are placed in the build command file through the mnemonics <WORKDIR>, <PATH>, <FILE>, and <FULLNAME> will have a UNIX name format, relative to the user's home directory, instead of a DOS or Windows full path format.

### RSH

Any remote command execution tool can be used to run the build commands on the UNIX server. The description hereunder uses RSH as an example, because this is a tool that is installed by default on all UNIX and Windows systems. Configuration for other tools will most likely be similar.

Make sure that the PC user has RSH access to the UNIX host. To do this, it is usually required to place the user's machine ID in the user's \$HOME/.rhosts :

```
+ WS123
```

and the user's machine ID and name in /etc/rhosts.equiv:

```
WS123 myuser
```

The RSH connection can be tested by issuing a command from the Windows command prompt, for example

```
rsh unixserver -l me ls
```

should display the contents of the user's home directory.

If an "access denied" message is returned, security may require extra steps to successfully establish an RSH connection.

**Note:** If the Windows user name differs from the UNIX user name, then the connection to the RSH daemon will be made using the Windows user name, while the RSH commands are issued using the UNIX user name. This may be very difficult to configure in a well secured environment.

Once successfully connected, add in the AW\_OBJ.INI file a BUILDSTARTER key containing the correct Windows command line to start a command on the UNIX host:

```
[DEVELOP]
BUILDSTARTER=rsh unixserver -l myuser.
```

**Note:** The dot at the end of this command line is significant. This “dot” command allows interpretation of a shell script that has no execute permission set.

When SURE build support has created a build file for a project, for example, using the above settings, the build file for mainapp\mainsrc.c is X:\sources\mainapp\mainsrc\_BUILD.BAT, and SURE will issue the following command:

```
rsh unixserver -l myuser . sources/mainapp/mainsrc_BUILD.BAT
```

which executes sources/mainapp/mainsrc\_BUILD.BAT as a shell script from the user's home directory.

### Build Commands

SURE creates a file named BUILD.ERR in the directory where it executes the build commands. It is required that the build commands remove this file on successful completion of the build script or place its error messages in this file on error.

A set of build commands for a file in SURE looks typically something like this:

```
cd <WORKDIR>/<PATH>
make -f <FILE>.mk 2>BUILD.ERR
test $? -eq 0 && rm BUILD.ERR
```

This creates a shell script (sources/mainapp/mainsrc\_BUILD.BAT) containing:

```
cd sources/mainapp/
make -f mainsrc.mk 2>BUILD.ERR
test $? -eq 0 && rm BUILD.ERR
```

The first line ensures the proper working directory.

The second line executes the make script mainsrc.mk to build the software release, piping the standard error channel to the file BUILD.ERR.

The third line ensures that the file BUILD.ERR is removed if the exit value of the make command equals zero.

**Note:** The file mainsrc.mk in the above example is a project file loaded in the SURE repository as sources/mainapp/mainsrc.mk.

### 12.8.2. Make Naming of Build Script Files Configurable

During the SURE build process, names of the script, error, and list files are automatically generated, but the user has some level of control over the naming of these files.

A number of entries can be added to the INI file to control naming of build related files.

The values of these entries contain a simple way to include the name of the current build target by including the literal <> in the value.

For example, FILESCRIPTNAME=Do\_<>\_Build.bat will expand to the build script name Do\_TEST\_Build.bat when building the target TEST.SLN.

The following entries can be defined:

FILESCRIPTNAME	the name of the build script for a single file
DIRSCRIPTNAME	the name of the build script for a directory
MAKESCRIPNAME	the name of the make file when build step MAKE is used
ERRORNAME	the name of the error file created by a build
LISTNAME	the name of the listing file created by a build

### 12.8.3. Starting Build Scripts

The SURE build server does basically three things for each file that is in the build queue:

- Synchronize the source-file in the build-environment with the source-file in SURE.
- Prepare and start the build script.
- Report the result of the build (Syntax or OK) back to SURE.

There are several methods to start build scripts:

- Create and start an individual build script for each source-file that must be built.
- Start one pre-defined build script (a MakeFile) that does the build for the entire application.

#### 12.8.3.1. Individual Build-Script for Each Source

This method is described in the previous paragraphs.

Build commands are defined in SURE at four levels: source-file, file-type, project, and system.

The following is done for each source-file that is mentioned in the build queue:

- Synchronize the source file in the build environment with the source file in SURE.
- Use the build commands that are used to generate a build script.
- The build-script is started.
- The result is reported back to SURE.

The build scripts are started one by one. SURE waits until a build script is done before the next one is started.

The progress of the build can be followed using the SURE compile interface: Each environment has a <to build> button with the number of files that are still to be bound. This number is updated during the build. Clicking the refresh button shows the most recent values.

This mode requires the following settings in the AW\_OBJ.INI configuration file:

```
[BATCH]
UNATTENDED=TRUE
ENVIRONMENT=<SURE environment>
SERVER=<object-server>
```

### 12.8.3.2. Common Make-File for the Entire Application

In the case of a common script file for the entire application, the following happens:

- The SURE build server copies all source-files that are mentioned in the SURE-build-queue from the repository to the build-environment.
- The SURE build server creates a data file called BUILD.LST. This file contains the names of the source-files that are mentioned in the SURE-build-queue. Files that are mentioned in the SURE-build-queue must be rebuilt, because they are changed or they have to be rebuilt for other reasons (for example, a changed copy-book).
- The build Make-file is started. The SURE-build-server waits until the make-file is ready.
- The Make-file must create a file called BUILD.ERR. This file contains the results of all the builds. The SURE-build-server reads this file and reports the build-results back to SURE. That means that at least the files that were mentioned in the BUILD.LST file have to be reported back using the BUILD.ERR file; otherwise, file will remain forever in the SURE-build-queue. BUILD.ERR can be extended with build results of source-files that were not mentioned in BUILD.LST. The SURE-build-server reports all the mentioned build results back to SURE, even for files that are not known in SURE.

In this case, there is one build scripts: the make-file. SURE-build-server waits until it is done, before it can to report the build results back to SURE. That means the progress of the make-file cannot be followed using the SURE compile interface.

This mode requires the following settings in the AW\_OBJ.INI configuration file:

```
[BATCH]
UNATTENDED=TRUE
SERVER=<object server>
ENVIRONMENT=<SURE environment>
COMMANDS=MAKE
MAKECOMMAND=<the full title of the make file>
```

### 12.8.4. Integrate Build on Windows with SURE-Batch on MCP

This paragraph describes the configuration of how to start a build for the Windows or UNIX platform from the SURE compile program (RESPECT/SURE/COMPILE) that runs on the MCP.

#### Detailed Information

The RESPECT/SURE/COMPILE program can start a job using a START-JOB mechanism. It is possible to link a START-JOB to the name of a build-server. SURE-compile starts that job when a PC-source which is connected to that build-server is in the build queue.

This method has the following two advantages:

- The build on the Windows platform is automatically started.
- The PC-build results are used in the integrity-check of RESPECT/SURE/COMPILE. COMPILE waits until the PC-build is completed. It may happen that PC-programs and MCP-programs are changed because of the same task, and that the SURE integrity mechanism must cover both platforms. If a PC-build fails with syntax errors, then MCP-programs that are modified because of the same task will not be deployed, until the syntax error is fixed.

#### Technical Details and Considerations

Part of the configuration must be done on the MCP and another part on the Windows server where the SURE build server is installed.

#### Configuration on the MCP Platform

Create start-job WFL/AT\_BUILD\_SERVER

- This job is started by the RESPECT/SURE/COMPILE when one or more PC-programs are in the build queue.
- Create the job with the following content (copy and paste):

```
00000500 BEGIN JOB WFL/AT_BUILD_SERVER(String ENVIRONMENT
00000750                                     ,String SERVER);
00001000 JOBSUMMARY = SUPPRESSED;
00001500 QUEUE      = 8; % mixlimit = 1 for one build at a time on windows!
00002000 String  PARAMS,ACTIE;
00002200
00002400 DISPLAY(ENVIRONMENT &" "& SERVER &" "& String(PR,2));
00002600
```

```

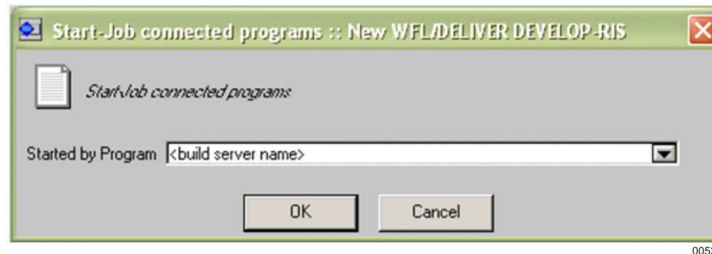
00003000 %----- Start the Build -----
00003500 ACTION := "RUN [RESULT=0] D:\Build\StartBuild.bat ";
00004000 PARAMS := ENVIRONMENT & " " & SERVER;
00004050 IP_BUILD_SERVER := "<ip-address-of-the-Windows-build-server>";
00004500
00005000 RUN *SYSTEM/NXSERVICES/PCDRIVER
00005050                (IP_BUILD_SERVER & " " & ACTION & PARAMS);
00006000
00006500 RUN OBJECT/RESPECT/SURE/FINISH(SERVER);TASKSTRING = ENVIRONMENT;
00007500 END JOB.

```

- Replace <ip-address-of-the-Window-build-server> with the ip-address of the Windows server where the SURE build server is installed.
- Save the file and load it in SURE.

Configure in SURE that the job must be started by the RESPECT/SURE/COMPILE when one or more PC-files are in the build-queue of the build-server as follows:

- Start SUREforWindows.
- Open the properties screen of start-job WFL/AT\_BUILD\_SERVER.
- From the Configuration menu, select Start-Job, click Started-By to go to the screen "start-job connected programs."
- Click New.



- Enter the name of the logical build-server and click OK. The logical build server is the build-server that is also assigned to PC-files in SURE.

The start-job works as follows:

- The start-job runs the SYSTEM/NXSERVICES/PCDRIVER program with the following parameter:  
<ip-addr> RUN [RESULT=0] D:\Build\StartBuild.bat. <environ>.<server>  
Where
  - <ip-addr> = the ip-address of the Windows-server where the SURE build server is installed.
  - <environ> = the SURE environment of the SURE-evening-batch.
  - <server> = the logical build server.
- SYSTEM/NXSERVICES/PCDRIVER passes that parameter to a Launcher program on the Windows-build-server.

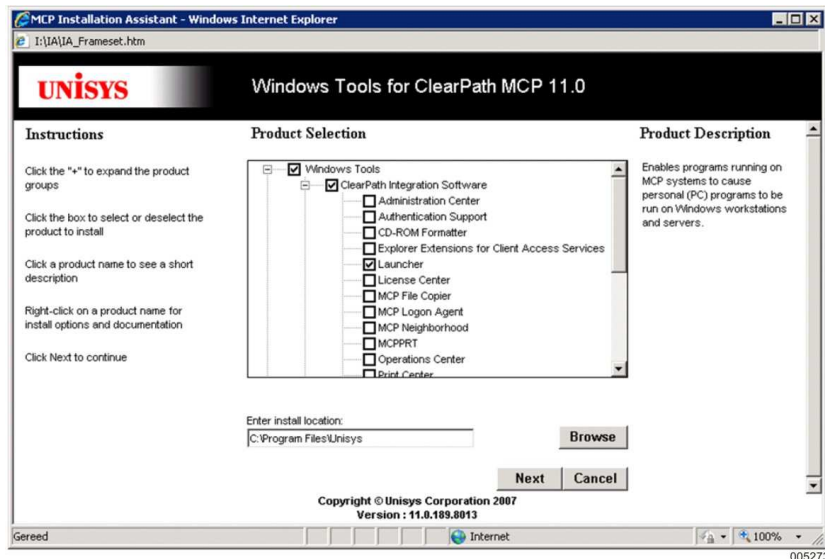
- The Launcher program executes the parameter as a DOS-command.
- The parameter instructs the Launcher to start bat-file D:\Build\StartBuild.bat on the Windows build-server, with parameters <environment>, and <SURE-build-server>.

### Configuration on the Windows Build-Server:

The first step is the installation and configuration of the Launcher on the Windows build server. The Launcher is a Windows application and it executes DOS commands.

Step 1: The installation of the Launcher:

- Start Windows Explorer and open the directory \\<MCP-IP-address>\installs.
- Start setup.exe from the installs directory.



- Mark the Launcher checkbox and click [Next] to start the installation.
- Installation has created a start-shortcut in the start-up directory. This shortcut must be deleted to prevent the Launcher from being started when you connect to the server using remove desktop:
  - Start Windows explorer.
  - Browse to the folder C:\Documents and Settings\<userid>\Start Menu\Programs\Startup.
  - Delete the shortcut that start the launcher.



- Add an entry in the Windows registry that enforces the Launcher to be started automatically at Windows startup.
  - Create registry-file launcher.reg with the following content (copy and paste):

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"Launcher"="\"C:\Program
Files\Unisys\MCP\Launcher\Launcher.exe\"<MCP-ip-address>"
```
  - Replace <MCP-ip-address> with the IP-address of your MCP machine.
  - Save registry-file launcher.reg and double click to add the command in the registry.
- Create an extra bat-file Start\_Launcher.bat that can be used to restart the Launcher when it is stopped for some reason:
  - Create bat-file Start\_launcher.bat with the following content (copy and paste):

```
"C:\Program Files\Unisys\MCP\Launcher\launcher.exe" <MCP-ip-address>
```
  - Replace <MCP-ip-address> with the IP-address of your MCP machine.
  - Save the bat-file and move it to the MCP machine.

Step 2: Create the bat-file that is started using the Launcher. The name of this file is D:\Build\StartBuild.bat and it expects two parameters: <environment> and <build-server-name>

- Create D:\Build\StartBuild.bat file for the build process as follows:

```
rem for batch mode %1 = EnvironmentName, %2 = BUILDServerNaam

if "%1" == "" goto get_env
set env=%1
goto env_ok
: get_env
echo Enter a valid environment
: env_again
set /p env="Environment name>"
if not "%env%" == "" goto env_ok
echo ERROR Environment name expected
goto env_again
: env_ok

if "%2" == "" goto get_server
set serv=%1
goto serv_ok
: get_server
echo Enter a valid server name
: serv_again
set /p serv="Server name>"
if not "%serv%" == "" goto serv_ok
```

```
echo ERROR Server name expected
goto serv_again
:serv_okecho %date% %time% begin of build %env% %serv%
>>D:\Build\ActionBuild.log
D:
cd \Build\RIS\BIN
Aw_obj.exe D:\Build\%env%\%serv%\AW_OBJ.INI
echo %date% %time% end of build %env% %serv% >>D:\Build\ActionBuild.log
```

- The bat-file will ask for an environment-name if parameter 1 is empty.
- The bat-file will ask for a server-name if parameter 2 is empty.
- The SURE-build-server is started with an AW\_OBJ.INI file in directory D:\Build\<environment>\<server>.

The entire configuration was explained using an example. You can adjust the example to your own needs:

- The name of the start-job on the MCP.
- The name of bat-file StartBuild.bat.
- The directory where the SURE-build-server is installed.
- The directory where the AW\_OBJ.INI file is placed.

## 12.9. (Windows or UNIX Files) Distribution Support

The Windows interface of the SURE software contains distribution support for Windows or UNIX files. The FILE and the FILE-TYPE definition contain an indication called "distributed." This means that a file with this indication needs to be distributed in the network. Normally, a large number of source files within a PC project are used to build an executable and only the executable is distributed in the network.

From Tools menu, PC Environment, using the Distribution Support option, a directory may be created containing all the files required for distribution. The selection for this directory may be on SYSTEM, PROJECT, or SERVER basis.

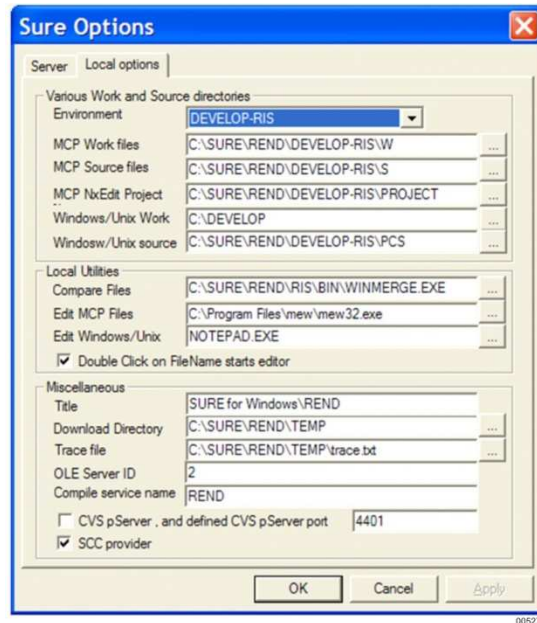
The actual distribution is not controlled by SURE. For this purpose, another tool should be used, such as Microsoft SMS.

## 12.10. SCC Provider

Microsoft has set a de-facto standard for communication with a Source Control System. This de-facto standard is named as the SCC interface and it applies to most of the Microsoft products and a vast number of third party products.

SURE can be easily configured as a SCC provider using the function Local option from the Options menu, under SURE options.

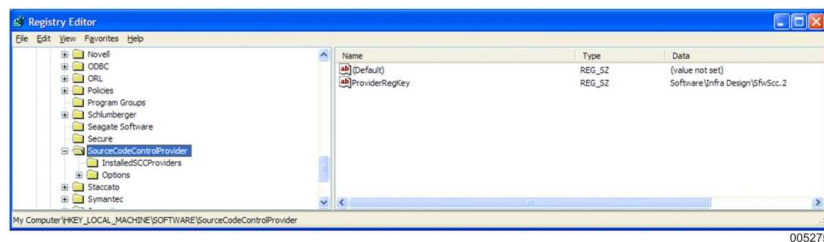
The dialog contains a checkbox named SCC provider which should be checked. Next time when SURE starts it registers itself as SCC provider.



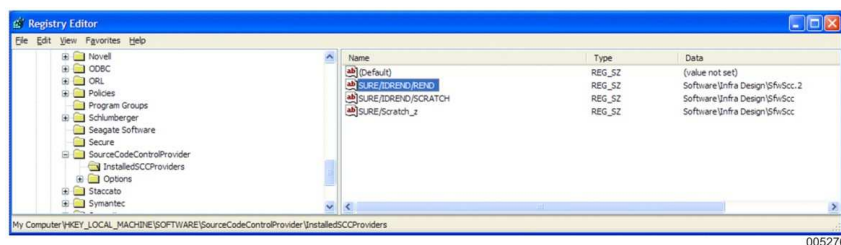
This registration is configured in the registry and this definition is not flexible if different SCC providers are used on the same machine. The default definition allows for multiple SCC providers; however, there can only be a single SCC provider active.

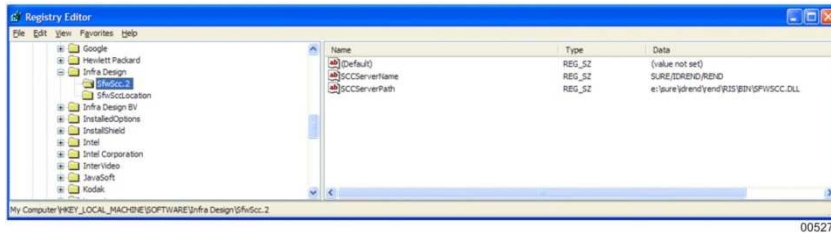
To help understanding this, the registry settings are described here.

Under HKEY\_LOCAL\_MACHINE/Software/SourceCodeControlProvider the following keys are present.



For every installed provider, a corresponding entry is present which locates the definition in the registry.





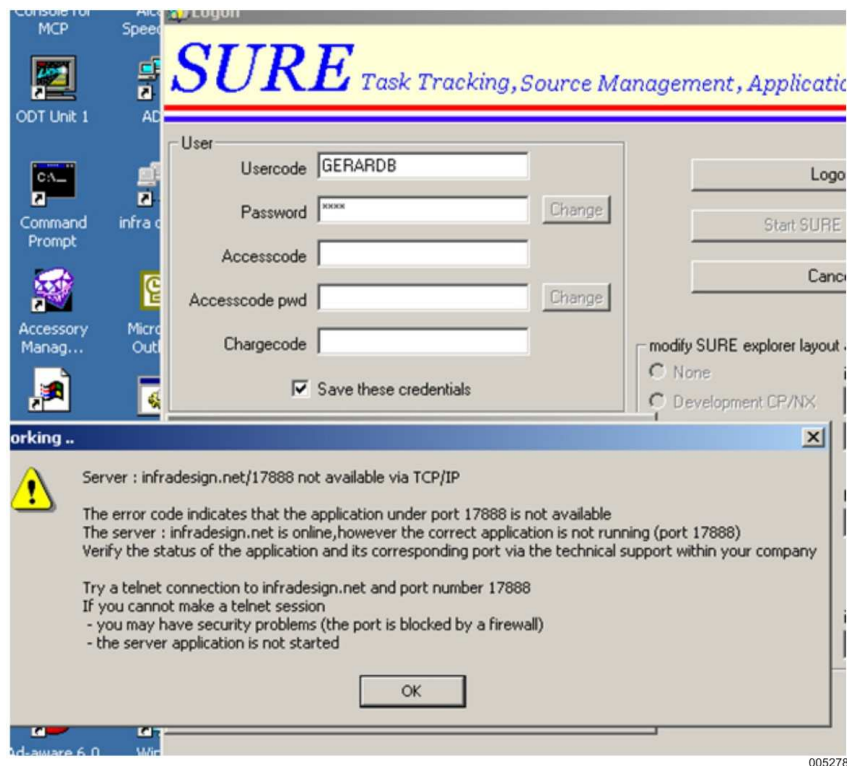
Every source control provider contains the location of the DLL servicing the requests.

### 12.10.1.Offline Processing for SCC Client

Using SURE as your source control provider requires that the SURE server is online. However, a SURE offline mode is supported when the SCC-interface is used. This offline mode enables you to check out and check in files while the SURE server is offline. This information will be kept in the INI file. As soon as the SURE server is online, the manipulated files can be synchronized using a special dialog.

This functionality will be explained using a case description.

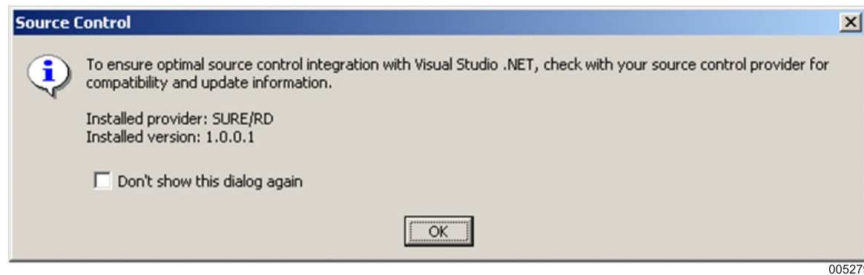
Advised for using the SCC interface is starting the SURE explorer separately. Suppose that the server is not online, then the following situation occurs.



Click OK, and leave the program running. You may click the DESKTOP icon, which minimizes this window.

Now the IDE that is using SURE as its source control provider is started. All the files are in the current defined work directory. And the IDE just opens the project file containing all the actual file references. The SURE SCC interface will respond with a message that the interface is offline and it will manipulate the check-in and check-out using the read-only indication on the files and with the INI file, where it signals all touched files. In the next example, Microsoft Visual Studio .NET is used as IDE.

The first dialog shows that SURE is being used as its source control provider.



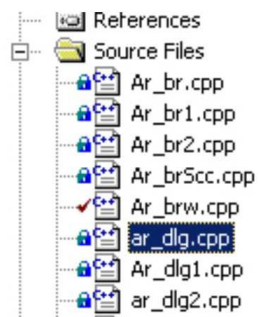
005279

The next dialog shows that the SCC interface recognizes that SURE is offline and that it will just simulate the source control commands. You will get this dialog twice in this scenario. If SURE goes offline during the session, you will get the dialog at the first action in the offline status.



005280

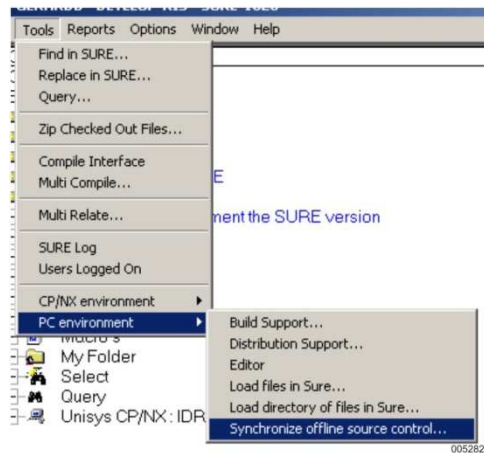
The IDE shows the status that is retrieved from the read-only bit from the file.



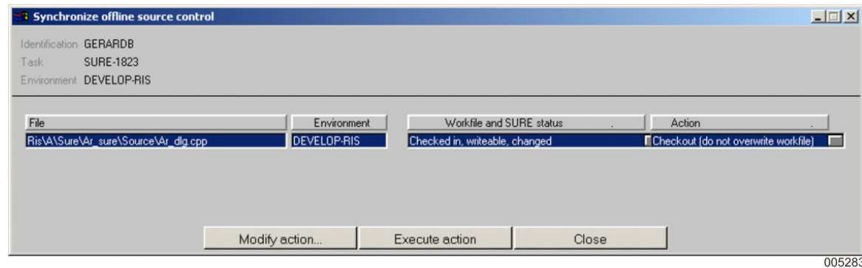
005281

Manipulating a file will just initiate the check-out dialog and modify the read-only bit.

Now, if the SURE server is online, this goes automatically, the manipulated files may be synchronized with the SURE repository. This function is initiated using the following function.



A dialog will be shown with the proposed action. You must execute an action for each file listed here. If the proposed action is not acceptable, you may modify it.



Thus, using the synchronize action; you will bring the status of the SURE repository in sync with the actual status of the files.

### 12.10.2. File Extension Associations for the SCC Interface

Some of the SCC interfaces require intelligence for associated files.

For example, Microsoft Visual Basic can contain FRM files in the project file, but each FRM file can have an FRX file associated with it, containing binary data that cannot be stored in the FRM file. This FRX file is maintained without being visible to the user from within the Visual Basic development environment. But Visual Basic does require the SCC provider to correctly handle the FRX files even if only the FRM file is specified in an SCC request.

For this reason, file extension associations can be defined, so that the SCC interface can automatically handle these "hidden" files.

In the INI file, define the section [SCCASSOCIATES], and for each extension that has one or more associated extensions, enter the main extension as a key, and the associated extensions, comma separated, as its value.

**Example**

In the INI file, the following entries must be defined for Microsoft Visual Basic:

```
[SCCASSOCIATES]
FRM=FRX
CTL=CTX
```

**12.10.3. Behavior of Get Latest Version Using the SCC Interface****Detailed Information**

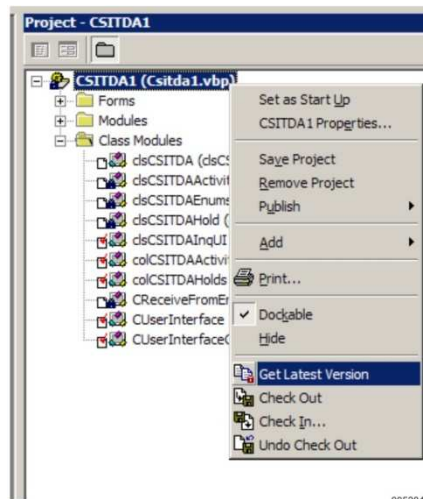
The Get Latest Version using the SCC interface behaves differently from the Get Latest Version using the SURE explorer, as explained in this paragraph.

**Note:** The IDE is in control of the list of files and the behavior of reloading the files in the IDE and that the SURE SCC acts as a server for retrieving the correct files. Any developer should carefully use the Get Latest Version function in relation to the behavior of the IDE used. There is no guideline for this because every IDE has its own characteristics.

**Example**

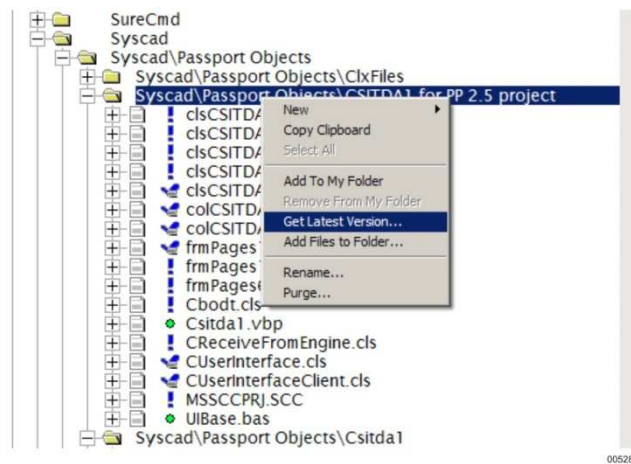
This example shows a difference in the behavior of function Get Latest Version on directory level.

Consider the following VB project.



Here the Get Latest Version function is controlled by VB and the files defined in the VBP project file. Therefore, VB controls which files are downloaded and it is not based on the file directory structure. For example, the different versions of VB do show a “strange” behavior in relation to the presence of files.

Consider the following SURE directory.



Here the Get Latest Version will download all the files within the directory. Therefore, it is not dependent on any logical project file but only on the file directory structure.

Any developer should carefully use the Get Latest Version function in relation to the behavior of the IDE used. There is no guideline for this because every IDE has its own characteristics.

### End-Example

The functional background of function GetLatestVersion is that a programmer copies all the files from the repository to the user's work directory. The SURE software will take action based on the Modify timestamp of the physical file on disk. This timestamp is compared against the timestamp stored in the repository. (Note the modify timestamp and not the created timestamp), and the following checks are done:

- The checked out status of the files is not altered when doing this function.
- The files which modify the timestamp matches the modified timestamp in the SURE repository are skipped.
- The files with the read-only indication set are overwritten without a warning. This is because the assumption is made that an older version of this file is present in the work directory.
- The files with the read-only indication reset are only overwritten after user verification. This is because the assumption is made that the programmer changed the read-only bit and changed the file without a formal checkout.



These decision steps are summarized in the following table.

<b>File is already Checked-Out</b>	<b>Compare Modify timestamp of File on disk with SURE</b>	<b>File on disk is Read-only</b>	<b>Action</b>
True	Don't care	Don't care	None
False	Equal	True	None
False	Equal	False	None
False	Not equal	True	Overwrite
False	Not equal	False	Verify

The GetLatestVersion can be initiated from the SURE explorer or from the IDE using the SCC connection, such as Visual Basic, Visual Studio .NET, or Micro Focus COBOL. The IDE is responsible for refreshing the files in the active workspace after that they have been refreshed.

**Note:** The default behavior of GetLatestVersion for a read-only file with a wrong timestamp is that the file is simply overwritten with the version from SURE. It is assumed that the file on disk is just an old version, and the version in SURE is modified using another work station by another developer. It is possible to change this default behavior to cause verification for every file using the following setting in the AW\_OBJ.INI configuration file:

```
[SURE]
GETLATESTVERSION=VERIFY
```

Overwriting a checked out file can only be done by an Undo Checkout command.

## 12.11.JAVA Eclipse CVS pServer Interface

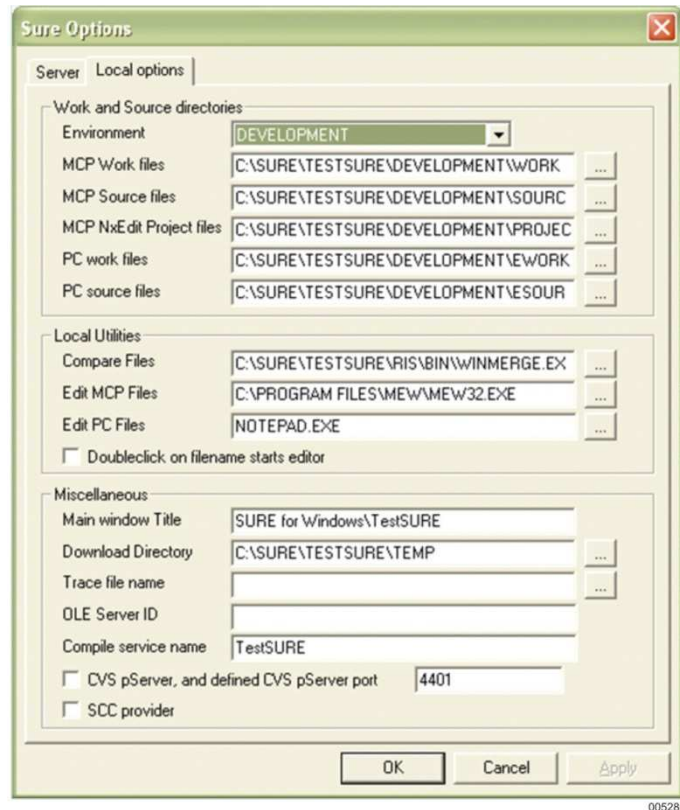
Using SURE it allows to store any PC or UNIX (through Samba software) file into the SURE repository. Using the SURE explorer, all SURE functions can be used. However, while developing software in an IDE (Interactive Development Environment), a seamless integration with an SCM like SURE provides optimal support for the developer. This integration allows the developer to check out the file from the IDE and after modification allows to check in the file. This integration allows the developer to work within the IDE without switching between the IDE and SURE.

The SURE for ECLIPSE interface uses the default ORG.ECLIPSE.TEAM.CVS plug-ins which are installed with ECLIPSE. The SURE client software, installed on the same client running ECLIPSE, emulates the CVS PSERVER protocol. For this reason, the standard ECLIPSE installation can be used without any additional plug-in. The CVS PSERVER protocol is a command line text driven interface. For this reason, various different ECLIPSE releases may not work correctly.


### Configuration

#### The SURE Configuration

The Local Options function on SURE Options Local Options contains an entry where SURE can be configured as a CVS pServer by defining the CVS (TCP/IP) port number.



The CVS server is integrated into the SURE-GUI.

In general, it is unwise to use the SURE browser interface while CVS commands are running, as they may interfere with each other. To keep the SURE browser available while safely acting as a CVS server, a special browser mode has been introduced. In this "iconized" mode the SURE browser will not be visible, but a system tray icon (  ) will be visible indicating the SURE browser is running.

The SURE-icon appears as follows in the notification area of the Windows task bar.



Clicking the tray icon will show an option to show or hide the browser.



This makes it possible to perform incidental manual actions on the SURE browser while no CVS commands are running, and afterwards to hide it again, to avoid accidental use of it while CVS commands may be executed.

If the SURE-browser interface has to be used frequently, and concurrently with CVS-commands, we recommend installing a second version of the SURE-browser on the same workstation as follows:

- The first installation is iconized and used by Eclipse to process CVS-commands.
- The second installation is not iconized and used interactively by the user.

Both installations are totally independent of each other.

This browser mode is instantiated by the following settings of the AW\_OBJ.INI file. Because of the changed implementation some old settings have to be removed from the INI-file and some new settings have to be added.

- Delete the following settings:

- In paragraph [GLOBAL]

```
PSEVER=<number>
CVS=TRUE
```

- In paragraph [CVS]

```
MAXMEM=<max-memory-usage> (deimplemented)
```

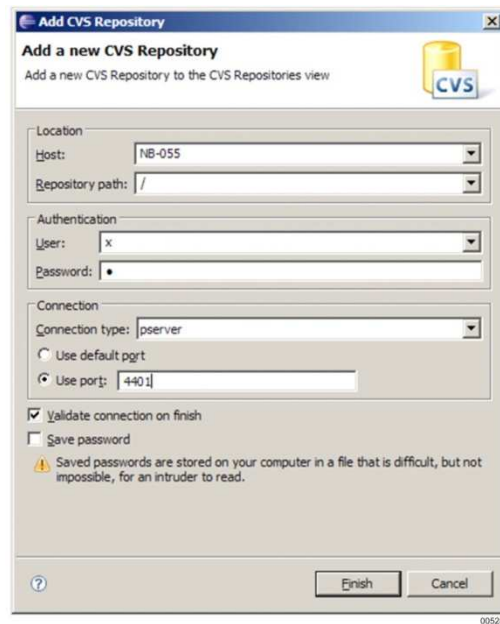
- Add the following settings:

```
[CVS]
PORT=4401
ROOT=/SURE
```

```
[SURE]
ICONIZED=TRUE
```

### The ECLIPSE Configuration

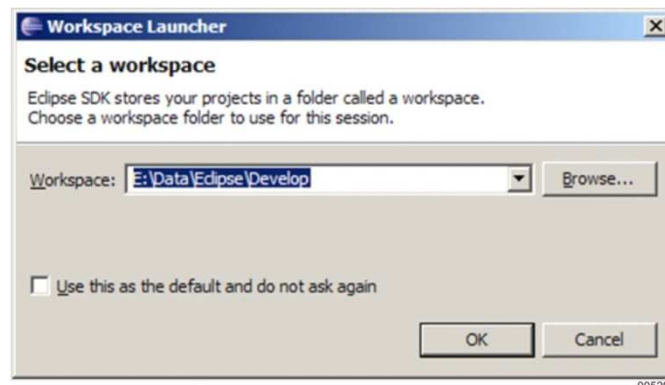
In the ECLIPSE interface, the CVS repository exploring perspective must be opened. Within this perspective, a new repository location must be defined according to the following screen example.



HOST	The computer name of this client.
Repository Path	Eclipse requires entry of a forward slash.
User	Eclipse requires entry of any string; however, not used.
Password	Eclipse requires entry of any string; however, not used.
Connection type	pServer.
Use port	Must be equal to the port number defined in the SURE configuration.

### Workspace and Work Directories

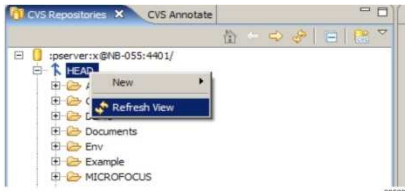
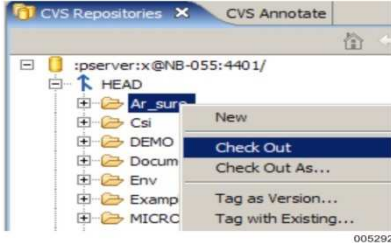
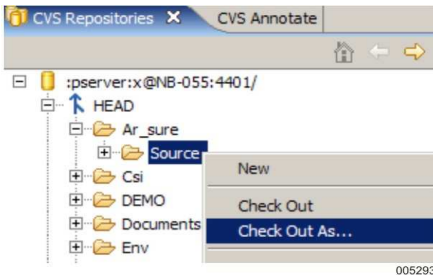

Both SURE and ECLIPSE uses a directory for the work files. Now, it is required that both directories differ. So, none of the Windows or UNIX (source and work) directories in the SURE local options may not be the same as the workspace used in Eclipse.

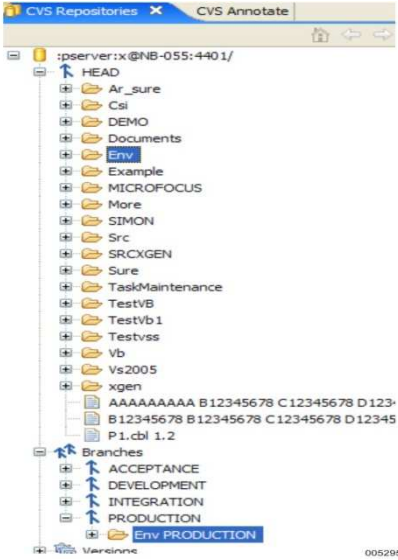


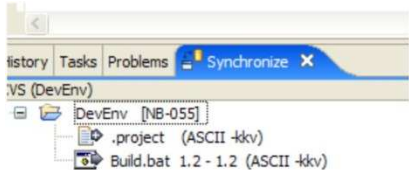


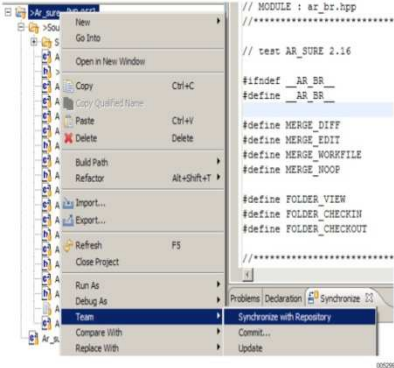
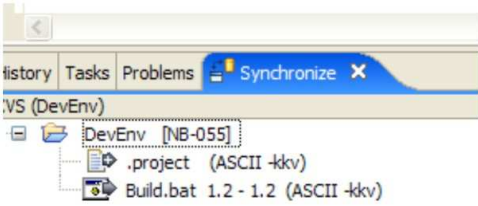
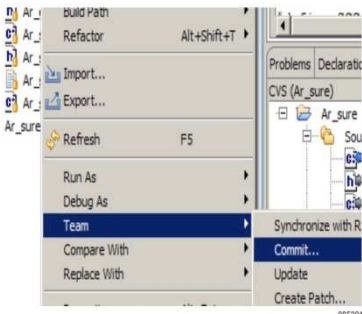
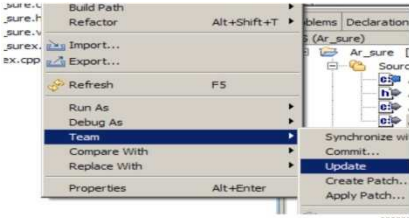
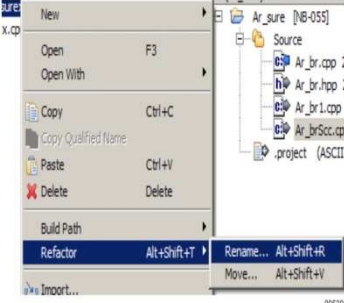
**Note:** This document only guides the specific use of eclipse. This document assumes an understanding of the eclipse interface.

## Terminology

The terminology used by CVS and Eclipse and by SURE is slightly different. For this reason, the following table defines the relation between the various commands. The following table defines the commands that are supported by the SURE interface. Therefore, using any other commands as listed here gives unpredictable results (most likely an error indication).

CVS action	SURE action
<p>Open the HEAD view from the CVS Repository exploring view</p> 	<p>List all root directories from the SURE development environment (lowest defined environment in SURE).</p>
<p>Check Out a (root) directory from the CVS Repository exploring view</p> 	<p>This function does a GetLatestVersion for all the files in the selected directory and copies these files as the directory name into the eclipse workspace. This results in the project shown in the java perspective, which allows modification and commit of the individual files.</p>
<p>Check Out As a (root) directory from the CVS Repository exploring view</p> 	<p>This function does a GetLatestVersion for all the files in the selected directory and copy these files as the given AS name into the eclipse workspace.</p>
<p>Refresh branches</p> 	<p>This function shows all root projects in the HEAD and allows viewing the different SURE environments for the selected projects. You can again check out such a project.</p> <p><b>Note:</b> If the same project is checked out twice, in different environments, you should use Check Out As.</p>

	
<p>Compare with another branch, this function in the CVS repository browsing perspective is not implemented</p> 	<p>This function is not implemented. If a project needs to be compared between environments, follow the following procedure:</p> <ul style="list-style-type: none"> <li>• Check Out the project.</li> <li>• Do a Compare with Branch in the JAVA perspective which opens the synchronize perspective.</li> </ul>
<p>Synchronize with repository of a project that was Checked Out As</p> 	<p>This inquires the status of the SURE repository and lists all the objects that differ in the eclipse workspace. If there are files listed in conflict mode they need to be merged using the compare editor. As a result, the team synchronizing perspective is shown.</p> 
<p>Synchronize with repository</p>	<p>This inquires the status of the SURE repository and lists all the objects that differ in the eclipse workspace. If there are files listed in conflict mode, they need to be merged using the compare editor. As a result the team</p>

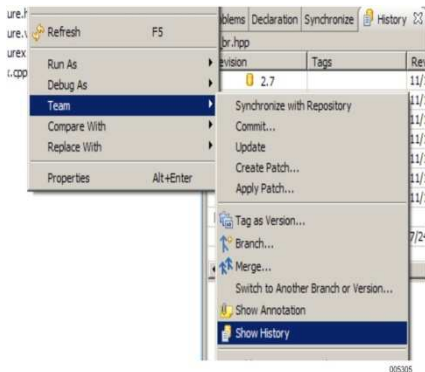
	<p>synchronizing perspective is shown.</p> 
<p>Team Commit</p> 	<p>This action does a check in of the file in the SURE repository. The current task is used to link the change.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>The Eclipse interface does not actually check out the files from SURE. This check-out is implicitly done at commit phase.</li> <li>The comment is not used in the SURE interface, the comment shown in the file history consist of the task identifier and the first line of the task description.</li> </ul>
<p>Team Update</p> 	<p>This action does a Get Latest Version of a file and updates the eclipse workspace. If the file is in conflict mode, the file is merged by the SURE software. However, the interaction with the eclipse plug-in varies from different releases and extreme caution should be taken when using this function in conflicts mode.</p>
<p>Refactor Rename</p> 	<p>This action creates a new file with the new name. The old file is not deleted and should be deleted using the SURE explorer.</p>

Delete



This action deletes the file from the eclipse project. The actual delete is done when the project is committed.

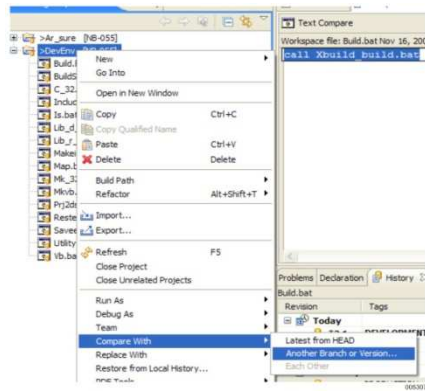
Team Show History



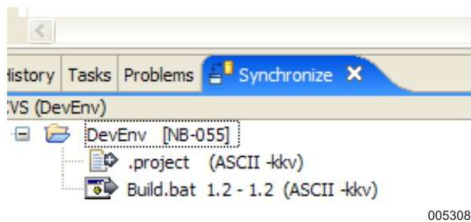
This function lists all delta files in the history perspective. Using this any version can be compared using the standard eclipse functions. The column comment is filled with the task name and the first line in the task description.

Revision	Tags	Revision Time	Author	Comment
2.2	DEVELOPMENT	11/27/06 10:02...	DEVELOPER	T_0123 : Cange by GerandB on rdef.hpp
2.1		11/27/06 9:59 AM	DEVELOPER	T_0123 : Cange by GerandB on rdef.hpp
1.2	PRODUCTION, A...	11/21/06 2:28 PM	(not logged)	no message

Compare a project with another branch



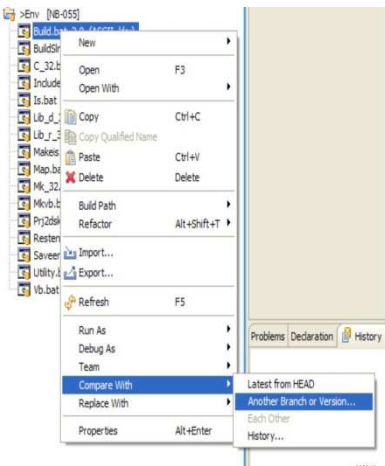
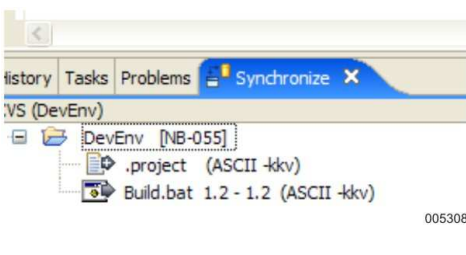
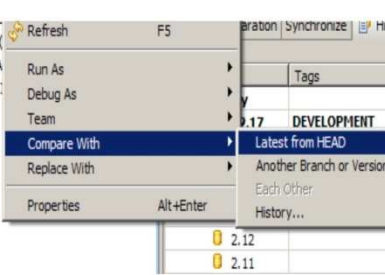
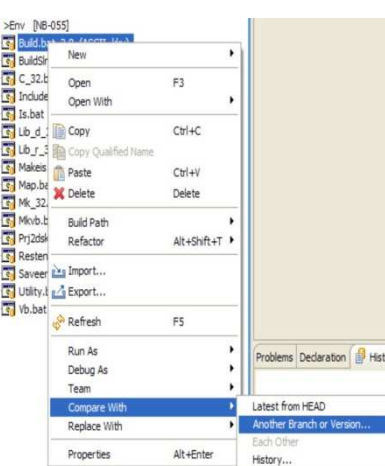
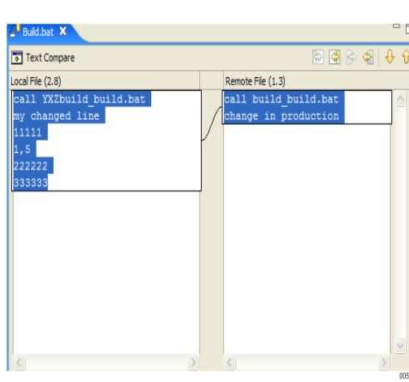
This inquires the status of the SURE repository within the selected environment (branch) and lists all the objects that differ in the eclipse workspace. If there are files listed in conflict mode they need to be merged using the compare editor. As a result, the team synchronizing perspective is shown.

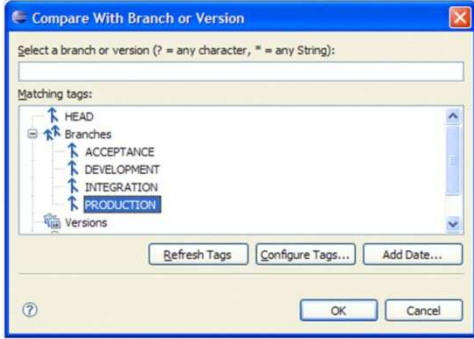
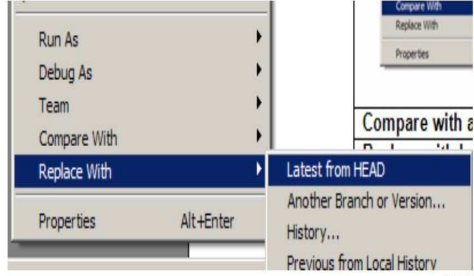
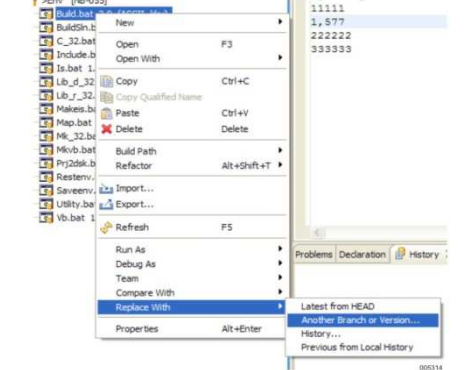


Compare a project with HEAD

This inquires the status of the SURE repository within the development environment (branch) and lists all the objects that differ in the eclipse workspace. If there are files listed in conflict mode, they need to be merged using the compare editor. As a result, the team synchronizing perspective is shown.



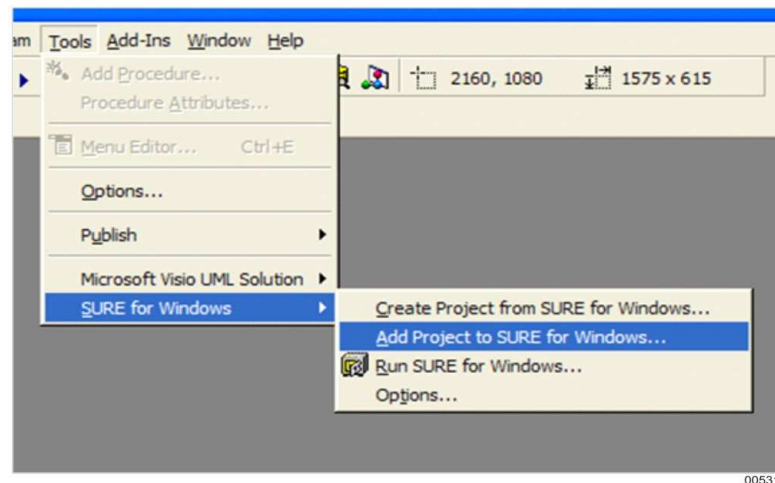
	
<p>Compare a file with latest from HEAD</p> 	<p>This compares a changed file with the latest version from development.</p>
<p>Compare a file with another branch or version</p> 	<p>This compares the version in the JAVA perspective with another version in one of the SURE environments.</p> 

	
<p>Replace with latest from HEAD</p> 	<p>This function replaces the file in the eclipse workspace with the development version from the SURE repository.</p>
<p>Replace with another branch or version</p> 	<p>This function replaces the file in the eclipse workspace with another environment version from the SURE repository.</p>
<p>Team create patch</p>	<p>NOT IMPLEMENTED</p>
<p>Team apply patch</p>	<p>NOT IMPLEMENTED</p>
<p>Team tag as version</p>	<p>NOT IMPLEMENTED</p>
<p>Team branch</p>	<p>NOT IMPLEMENTED</p>
<p>Team merge</p>	<p>NOT IMPLEMENTED</p>
<p>Team Switch to another branch or version</p>	<p>NOT IMPLEMENTED</p>
<p>Team Show annotation</p>	<p>NOT IMPLEMENTED</p>

## 12.12. Microsoft Visual Basic 6.0 Integration

For Microsoft Visual Basic 6.0, SURE should be configured as SCC provider. Now, after loading the software in the SURE repository, the situation may arise that the project file of VB does not show the source control commands like check-in and check-out. From Tools menu, under Sure for Windows, use Add Project function to Sure for Windows, to resolve this situation.

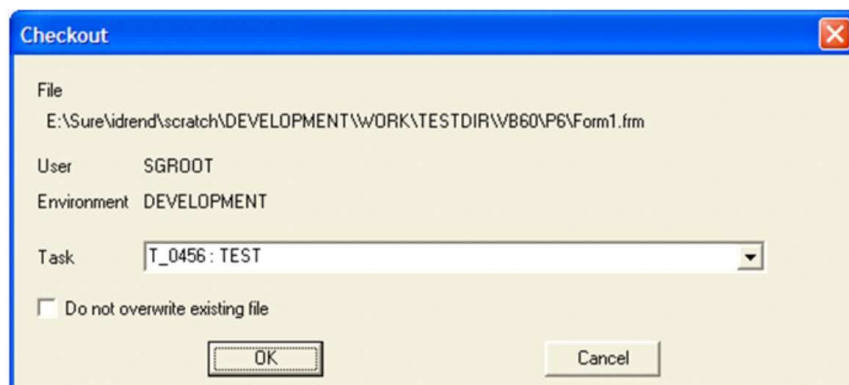
**Note:** This command can also be used when the files are already present in the SURE repository.



For new projects, the same route can be followed. The project can initially be added to SURE or it can be added later using Add Project to SURE for Windows.

The project may be in a state that VB responds "this file is not valid for this operation." From Tools menu, under Sure for Windows, use Add Files function to SURE for Windows, to repair this conflict.

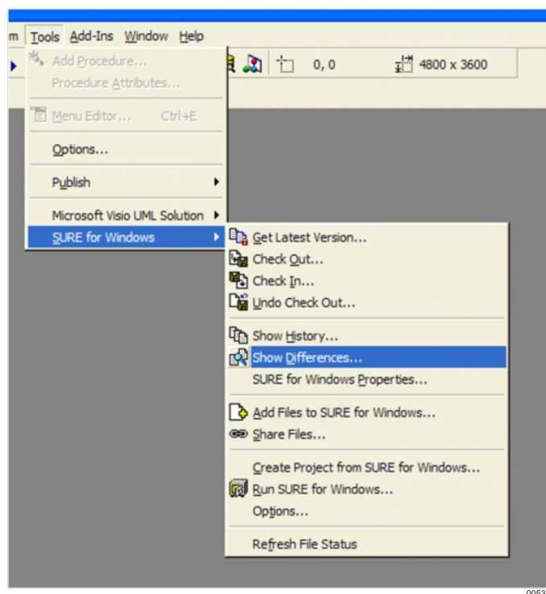
Because the SCC interface does not specifically contain the task oriented command, the SURE explorer should be used for this. When checking out a source, a dialog is given where the task for this specific change can be selected. The tasks in your To Do list are available in the drop-down list. If a new task must be entered the SURE explorer software should be used.



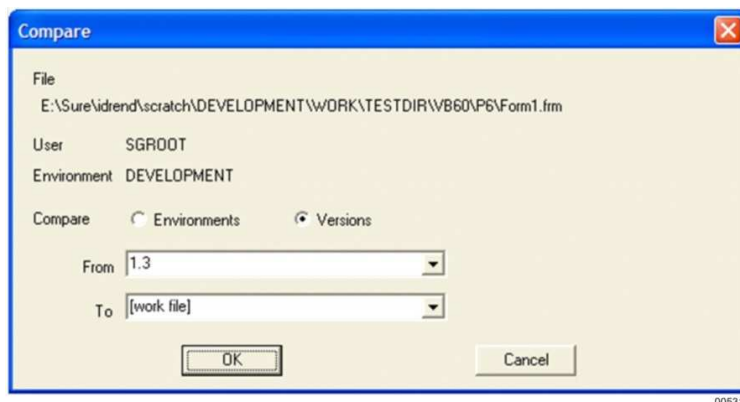
The Task field in this dialog shows all the tasks in your To Do list. Therefore, if the required task is not present in your To Do list, the SURE explorer should be used to locate the task. If you then make it your current task, it appears as the default in the previous dialog.

The SCC interface supports the basic commands, such as check-in, check-out, undo check-out, and GetLatestVersion. Furthermore, the command "show differences" is implemented, which allows showing the differences using the WinMerge tool. This command is present using the following function after selecting a specific file in the VB browse.

1. Click **Tools**.
2. Select **SURE for Windows**.
3. Select **Show Differences**.

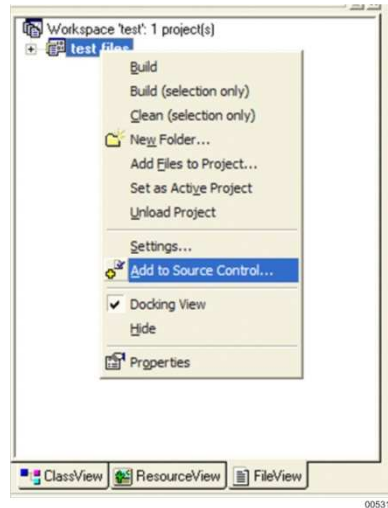


The dialog then allows comparing different versions from this file.



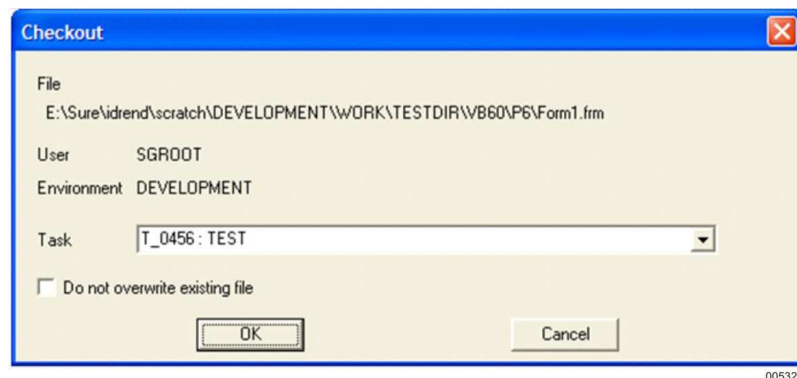
## 12.13. Microsoft Visual C++ 6.0 Integration

For Microsoft Visual C++ 6.0, SURE should be configured as SCC provider. Now, after loading the software in the SURE repository, the C++ workbench recognizes that the software is under source control and shows the check-out and check-in functionality.



For new projects, right-click the project in the FileView and select Add to Source Control. This function registers the sources into the SURE repository to make the check-out and check-in functions available.

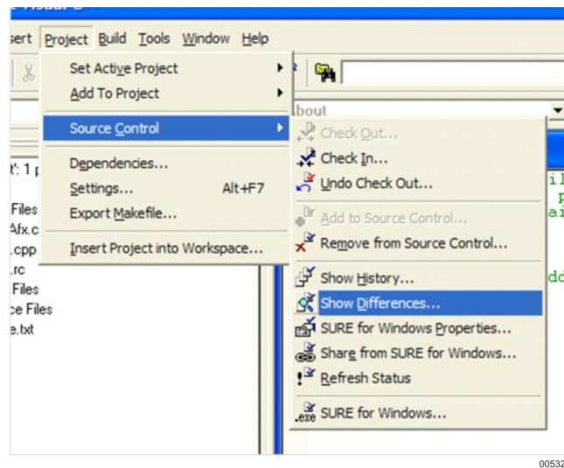
Because the SCC interface does not specifically contain the task oriented command, the SURE explorer should be used for this. When checking out a source, a dialog is given where the task for this specific change can be selected. The tasks in your To Do list are available in the drop-down list. If a new task must be entered the SURE explorer software should be used.



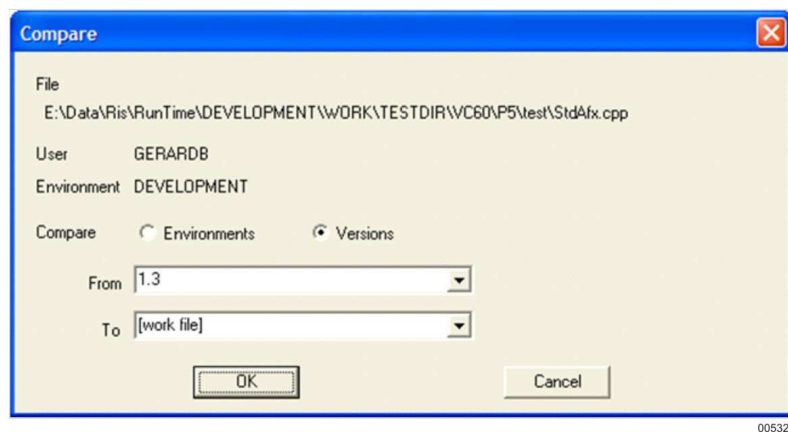
The Task field in this dialog shows all the tasks in your To Do list. Therefore, if the required task is not present in your To Do list, the SURE explorer should be used to locate the task. If you then make it your current task, it will appear as the default in the previous dialog.

The SCC interface supports the basic commands, such as check-in, check-out, undo check-out, and GetLatestVersion. Furthermore, the command “show differences” is implemented which allows showing the differences using the WinMerge tool. This command is present using the following function after selecting a specific file in the C++ browser:

1. Click **Project**.
2. Select **Source Control**.
3. Select **Show Differences**.

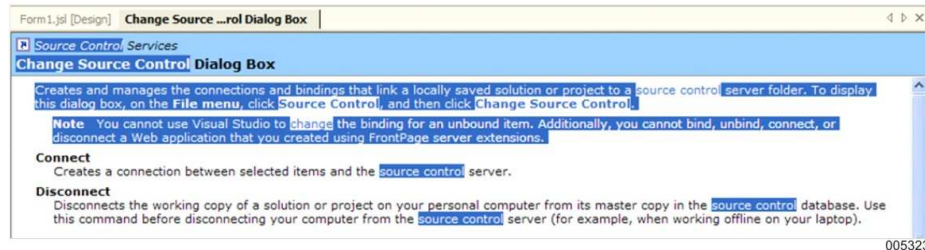


The dialog then allows comparing different versions from this file.



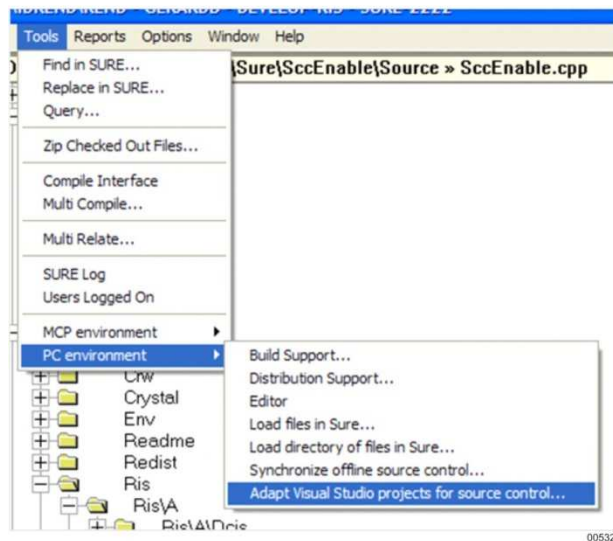
## 12.14. Microsoft Visual Studio 2003 Integration

For Microsoft Visual C++ 6.0, SURE should be configured as SCC provider. If the sources are loaded into the SURE database, and the solution (.sln) and project (.xproj) do not contain source control information, then that cannot be changed using the Microsoft Visual Studio software according to the following phrase in the Microsoft help file.



For this reason the SURE software contains a utility accessible from the Tools menu, under PC Environment, click Adapt Visual Studio projects for source control. This utility modifies the solution and project files, so that they are connected to SURE as a source control provider.

**Note:** The solution files and the project files need to be checked out because they are modified by this utility.

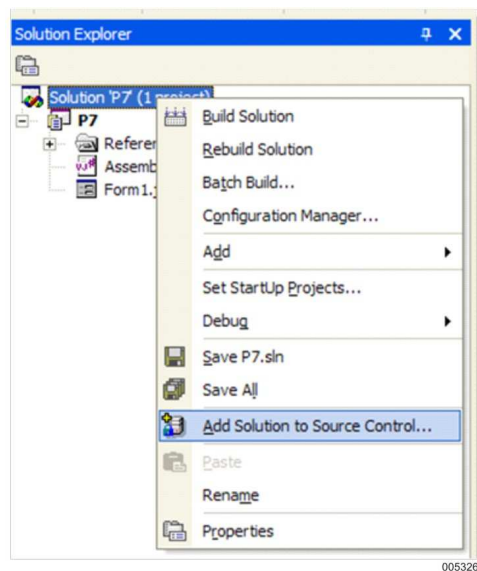


If the solution (.sln) and the project (.xproj) do contain source control information that points to another source control provider, then the Microsoft software issues a warning that describes that situation. This warning should be Ok.



Now, after loading the software in the SURE repository, the Microsoft workbench recognizes that the software is under source control and shows the check-out and check-in functionality.

New projects should be created and added to the source control using the popup menu under the solution or the project. This function registers the sources into the SURE repository and afterwards the check-out and check-in functionality are available.



Because the SCC interface does not specifically contain the task oriented command, the SURE explorer should be used for this. When checking out a source, a dialog is shown where the task for this specific change can be selected. The tasks in your ToDo list are available in the drop-down list. If a new task must be entered the SURE explorer software should be used.

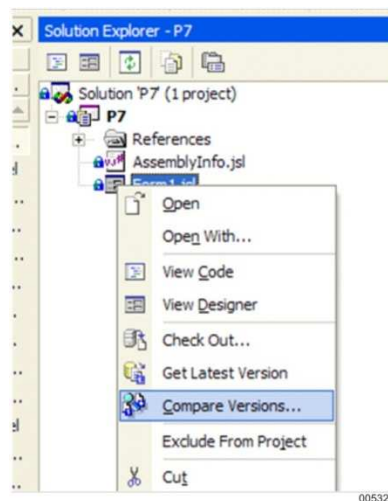




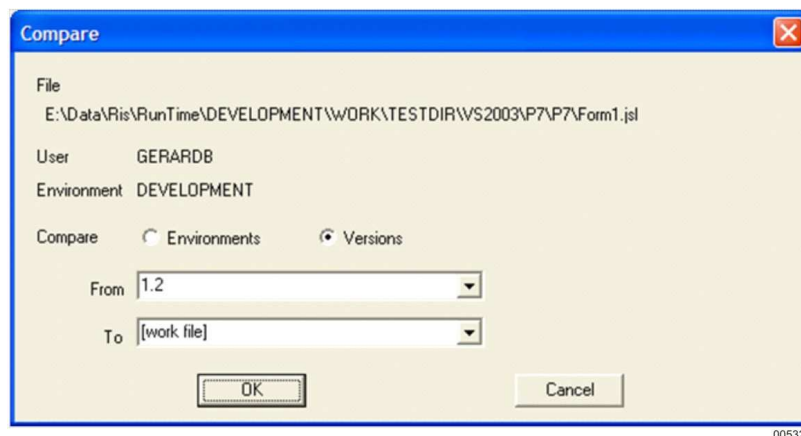
The Task field in this dialog shows all the tasks in your To Do list. Therefore, if the required task is not present in your To Do list, the SURE explorer should be used to locate the task. If you then make it your current task, it will appear as the default in the previous dialog.

The SCC interface supports basic commands, such as check-in, check-out, undo check-out, and GetLatestVersion. Furthermore, the command "Compare Versions" is implemented, which allows showing the differences using the WinMerge tool. This command is present using the following function after selecting a specific file in the file browser:

1. Right-click **file name**.
2. Select **Compare Versions**.

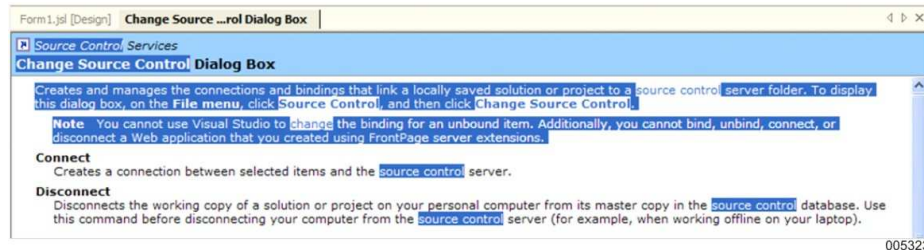


The dialog then allows comparing different versions from his file.



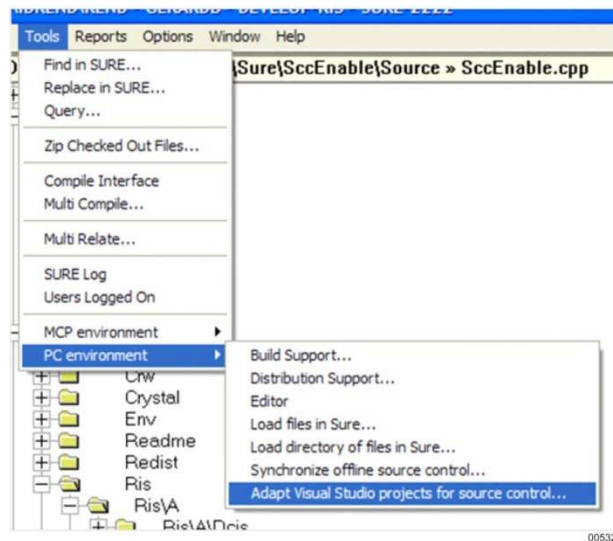
## 12.15. Microsoft Visual Studio 2005 Integration

For Microsoft Visual C++ 6.0, SURE should be configured as SCC provider. If the sources are loaded into the SURE database, and the solution (.sln) and project (.xproj) do not contain source control information, then that cannot be changed using the Microsoft Visual Studio software according to the following phrase in the Microsoft help file.



For this reason the SURE software contains a utility accessible from Tools menu, under PC Environment, click Adapt Visual Studio projects for source control. This utility modifies the solution and project files, so that they are connected to SURE as a source control provider.

**Note:** The solution files and the project files need to be checked out because they are modified by this utility.

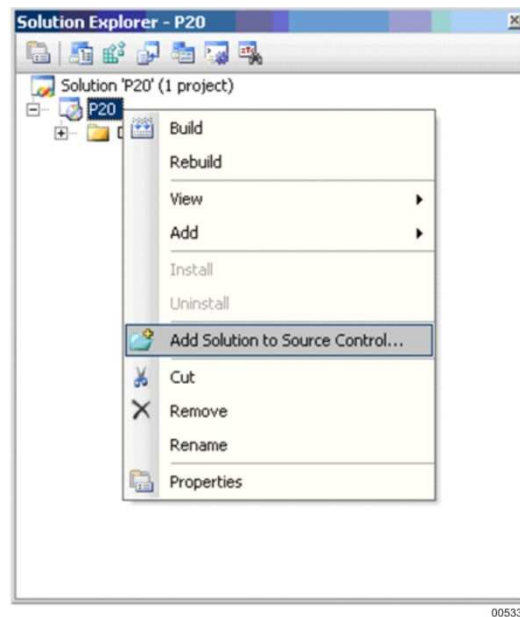


If the solution (.sln) and the project (.xproj) do contain source control information that points to another source control provider, then the Microsoft software issues a warning that describes that situation. This warning should be Ok.

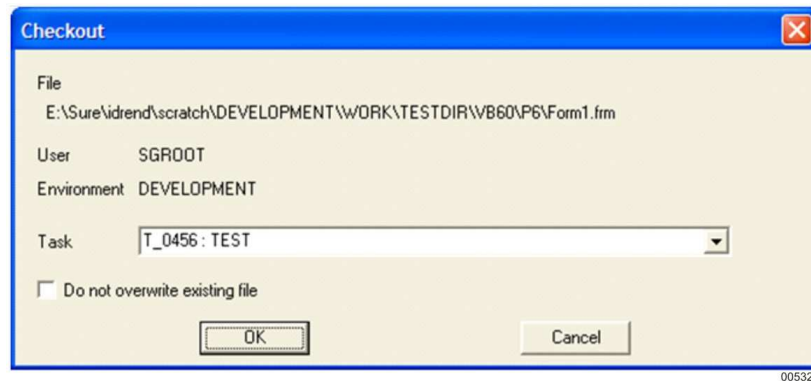


Now, after loading the software in the SURE repository, the Microsoft workbench recognizes that the software is under source control and shows the check-out and check-in functionality.

New projects should be created and added to the source control using the popup menu under the solution or the project. This function registers the sources into the SURE repository and afterwards the check-out and check-in functionality are available.



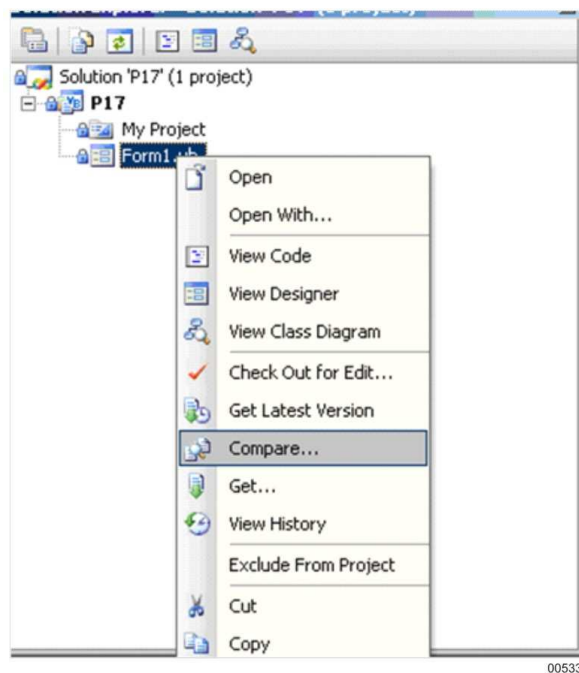
Because the SCC interface does not specifically contain the task oriented command, the SURE explorer should be used for this. When checking out a source, a dialog is given where the task for this specific change can be selected. The tasks in your To Do list are available in the drop-down list. If a new task must be entered the SURE explorer software should be used.



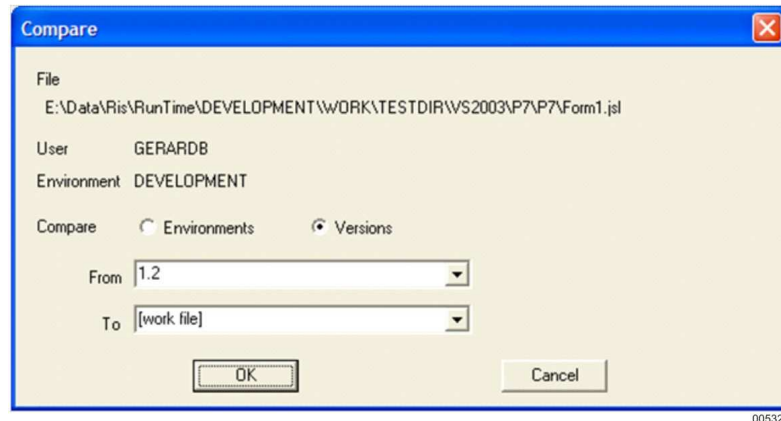
The Task field in this dialog shows all the tasks in your To Do list. Therefore, if the required task is not present in your To Do list, the SURE explorer should be used to locate the task. If you then make it your current task, it will appear as the default in the

The SCC interface supports basic commands, such as check-in, check-out, undo check-out, and GetLatestVersion. Furthermore, the command "Compare..." is implemented, which allows showing the differences using the WinMerge tool. This command is present using the following function selecting a specific file in the file browser:

1. Right-click the **file name**.
2. Select **Compare**.



The dialog then allows comparing different versions from his file.



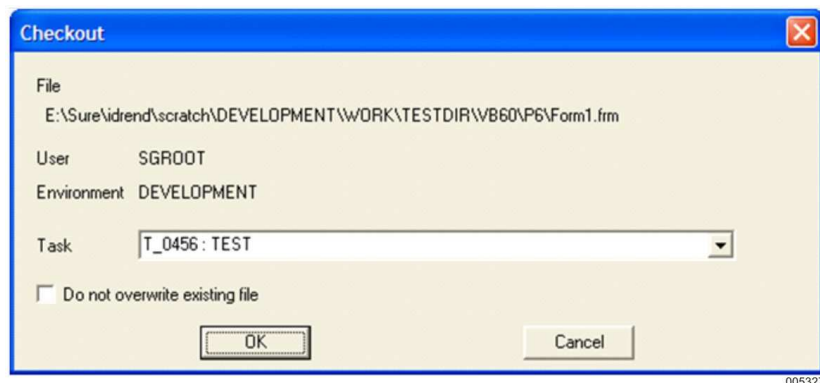
The combination Microsoft Visual Studio 2005 and SURE does contain a specific known bug. When initially opening a project that is bound to source control, the property window in the IDE does not automatically refresh. This seems to be a VS problem and therefore it cannot be solved in the SURE software. A detour for this problem is:

- First open a project not bound to source control.
- Right-click the property window title bar and select floating; this will refresh the contents of the property window.

## 12.16. MicroFocus Cobol Integration

For MicroFocus Cobol, SURE should be configured as SCC provider. MicroFocus Cobol allows Check-Out, Check-In, and Add to Source Control functionality.

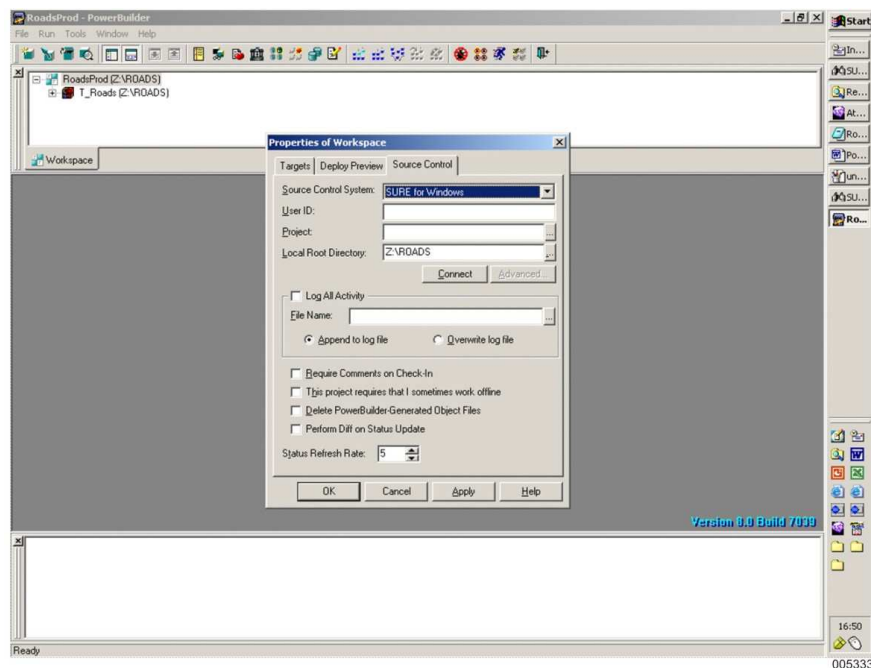
Because the SCC interface does not specifically contain the task oriented command, the SURE explorer should be used for this. When checking out a source, a dialog is given where the task for this specific change can be selected. The tasks in your To Do list are available in the drop-down list. If a new task must be entered the SURE explorer software should be used.



### 12.17. PowerBuilder Integration

Define SURE as the source control system for PowerBuilder as follows:

- Open the window “System Tree” from Window menu.
- Open the Output window for monitor purposes from Window menu, click Output.
- Open the workspace of your application from File menu, click Open Workspace.
- Open the properties of the workspace, right-click the workspace line in the system-tree and select “Properties.”
- Define SURE, select “SURE for Windows” from the drop-down box of field “Source Control System.”



The output window informs you that a connection to SURE is established.

## Technical Details and Considerations

### PowerBuilder PBG Files

PowerBuilder uses internal repository (PBL) files. A PBL contains all sources that are required to create a DLL. Each PBL has a corresponding PBG file, which contains the list of sources that are loaded in the source control system. The PBG file is automatically created and maintained by PowerBuilder when a new source is added to SURE. SURE presents an add-dialog for the source, but after that a second add-or-checkout-dialog is presented for the PBG file. The user must enter the same attributes (system, task, and file-type) twice: once for the source and once for the PBG file.



Unfortunately, the dialog for the PBG file may confuse the developer, because he might have not expected it. Therefore, it is possible to hide this extra dialog using the following setting in the AW-OBJ.INI file:

```
[PowerBuilder]
SILENT=*.PBG:<default project>:<default filetype>
```

- If no <default project> is defined, then the project of the current task is used.
- If no <default filetype> is defined, then the file-extension is used.

This enforces that all PBG files are loaded and maintained automatically and that they all get the given project and file type.

It is possible to extend the SILENT option for another file extension.

### Enable the SCC Interface of SURE

The SURE SCC interface has some impact on the behavior of various tools that run on your PC. For example, if the SURE SCC interface is enabled, then Visual Studio 6.0 tries to connect to the SURE Explorer at startup. Install Shield 7.0 has the same behavior. This behavior is not wrong, but it is unexpected when it happens for the first time. Therefore, we decided not to automatically enable the SCC interface when SURE is installed.

The SCC interface must be enabled manually using the following setting in the AW\_OBJ.INI file:

```
[GLOBAL]
SCCPROVIDER=TRUE
```

If this AW\_OBJ.INI file setting is defined, then SURE adds the following registry key at start-up:

1. Select **HKEY\_LOCAL\_MACHINE**.
2. Select **SOFTWARE**.
3. Select **SourceCodeControlProvider**.
4. Click **ProviderRegKey**.

If this AW\_OBJ.INI file setting is not defined, then SURE deletes the above-mentioned registry key at startup (when that key was available in the registry).

### Disable the SCC Interface of SURE

It is possible to disable the SCC-interface of SURE as follows:

#### Method A

Disable the following sccprovider-key in the AW\_OBJ.INI file:

```
[GLOBAL]
SCCPROVIDER=FALSE
```

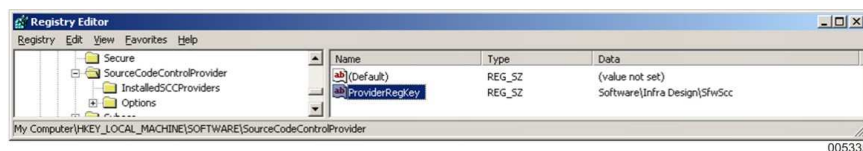
#### Method B

Step 1: Add the following setting in the AW\_OBJ.INI file in paragraph [GLOBAL]:

```
[GLOBAL]
OLESERVER=0
```

Step 2: Delete the following registry key:

1. Select **HKEY\_LOCAL\_MACHINE**.
2. Select **SOFTWARE**.
3. Select **SourceCodeControlProvider**.
4. Click **ProviderRegKey**.



### Map the Editor-Workspace to a Specific SURE Environment

The referenced SURE environment (where the source control actions are executed) is determined using the settings of the Local Work Directory for Windows or UNIX files in SURE. This local work directory must be defined for each environment using the Options menu, under SURE options, on the Local Options tab.



The source name on disk is the local-work-directory followed by the source name in SURE.

Some examples:

Source name on disk	Local work directory	Source name in SURE
C:\Develop\Abc.txt	C:\Develop	Abc.txt
C:\Develop\Roads\Abc.txt	C:\Develop	Roads\Abc.txt
C:\Develop\Roads\Sub\Abc.txt	C:\Develop	Roads\Sub\Abc.txt

Each environment must have a unique local work directory. The local work directory that matches with the first directories of the PC-source determines the SURE environment.

### Example

Consider the following local work directories in SURE.

Environment	Local Work Directory
DEVELOP	C:\Develop\PB
ACCEPTANCE	C:\Acceptance\PB
PRPDUCTION	C:\Production\PB

Consider the following workspace for the PC-editor = C:\Develop\PB\Roads\\*.\*

Source C:\Develop\PB\Roads\Autobase\d\_buyer.srd must be checked out from SURE.

The PC-source name starts with the work directory of environment DEVELOP (= C:\Develop\PB), so the source is checked out from DEVELOP. The source name in SURE = Roads\Autobase\d\_buyer.srd.

### Map a SURE Work Directory to a PowerBuilder Workspace

Consider the following workspace for the PC-editor = C:\Develop\PB\Roads\\*.\*

This workspace contains a source called C:\Develop\PB\Roads\Autobase\d\_buyer.srd

This source must be loaded in SURE as Roads\Autobase\d\_buyer.srd (because the first directory levels of the file name on disk determine only the work directory).

Therefore, in this example the work directory in SURE must be C:\Develop\PB.

### Performance Considerations (PowerBuilder)

We recommend creating a separate sub-directory for each PBL file to obtain the best performance.

### **Performance Considerations (general)**

In general, it is always better to load sources in SURE under one or more sub-directories. Loading the sources in SURE without a sub-directory, straight in the root-directory will not go wrong, but it decreases the performance for the following reason:

If you open SURE-folder "Files (PC)", then SURE returns you all files in the root-directory plus the first sub-directory-levels. Retrieving all the files in the root-directory may take some time (depending on the power of the mainframe). This is not a problem if you are the "owner" of the files in the root-directory, because in that case you want to see them. However, if you are not the "owner" of those files, and if you only want to open one of the sub-directories, then you are hindered by those "root-files" each time that you open folder "Files (PC)."

## Section 13

# Query Facility

The main function of the query facility is to select a group of files or items, so that a command can be executed on this selected group.

The repository stores the information extracted from SURE in a relation structure. This relation structure is built from the SURE commands (check-in, check-out, and other procedures). The second main input for this relation structure is provided by RESPECT/SURE/EXAMINE. This program parses files and stores structural dependency information (for example, copy files used by a source and libraries used by a source) into the relation structure.

To use the query requires some understanding of the relation structure and the possible entries stored in this structure. It is possible to pre-define query macros for common selections.

### 13.1. (Query) Relation Structure

The relation structure contains separate entries for each attribute stored for a file. Every record in the relation structure has the following fields:

ENVIRONMENT	The environment where the relation is active
ENTITY	The group of the relation
OWNER	The file owning the attributes
CLASS	The type of attribute
ASSET	The value for the attribute
TIME	The creation timestamp of the relation

See "(Repository) The Use of Relations" and "(Repository) Detailed Information about Relations" in Section 40 for more information about the relation structure.

Relations can be used in a query using the following syntax:

```
<environment>*<entity>|<owner>:<class>(<asset>{<timestamp>})
```

The fields of a relation are recognized as follows:

- The environment is terminated with a star <environment>\*.
- The entity is terminated with a pipe-sign <entity>|.

- The owner is terminated with a colon <owner>.
- The class is terminated with an open close parenthesis: <class>().
- The asset is between the parenthesis (<asset>).
- The timestamp is also inside the parenthesis between accolades directly behind the asset: (<asset>{<timestamp>}).

This relation-syntax is also used in this manual when a relation or a part of a relation is mentioned.

### Example

```
File TEST/1 requested by USER1
100 BEGIN
200 $ INCLUDE "COPY/A"
300 DATABASE DBX;
400 END.
```

```
File TEST/2 in maintenance by USER2 on pack DEV
100 BEGIN
200 $ INCLUDE "COPY/B"
300 DATASET DBX;
400 END.
```

The following relations are (amongst others) stored in the repository for these two files.

Environment*entity owner:class(asset)	Note
DEVELOP*FILE-CONTROL TEST/1:DATABASE(DBX)	1
DEVELOP*FILE-CONTROL TEST/1:COPY-FILE(COPY/A)	1
DEVELOP*FILE-CONTROL TEST/1:REQUESTED(USER1)	2
DEVELOP*FILE-CONTROL TEST/2:DATABASE(DBX)	1
DEVELOP*FILE-CONTROL TEST/2:COPY-FILE(COPY/B)	1
DEVELOP*FILE-CONTROL TEST/2:SOURCE-USERCODE(USER2)	2
DEVELOP*FILE-CONTROL TEST/2:SOURCE-PACK(DEV)	2

1—relation added during examine.

2—relation added during checkout.

It is obvious that the query syntax can only be used properly if you are familiar with the CLASS names used in the relation structure. An easy way to obtain knowledge of these class names is to look at the relations of a specific file or task (Right-click the name, select Relation).

Each relation that is available in the repository can be used in a query. A site can define its own relation-classes and relation-assets.

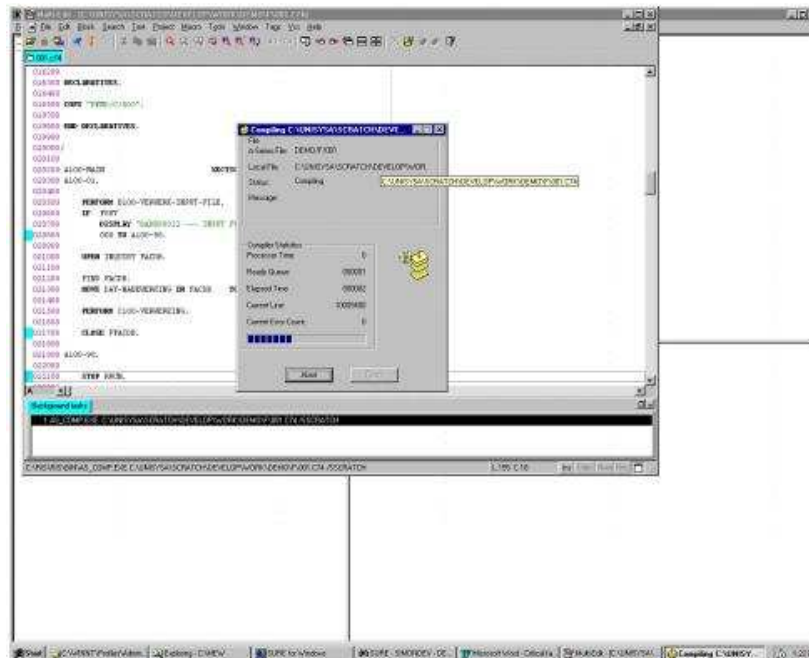
## 13.2. (Query) Names of Classes Used by the SURE Software

The following table gives an overview of the most frequently used relation classes.

Entity	owner: class(asset{timestamp})	Comment
FILE-CONTROL	<filename>:ASSIGNED(<user-id>)	Assign/Checkout
FILE-CONTROL	<filename>:REQUESTED(<user-id>)	Request/Checkout
FILE	<filename>:AUTHOR(<name>)	entry field
FILE-CONTROL	<filename>:COPY-FILE(<filename>)	EXAMINE
FILE-CONTROL	<filename>:DATABASE(<database>)	EXAMINE
FILE-CONTROL	<filename>:DATASET(<dataset>)	EXAMINE
FILE-CONTROL	<filename>:FILE(<external filename>)	EXAMINE
FILE-CONTROL	<filename>:DISK(<internal filename>)	EXAMINE
FILE-CONTROL	<filename>:STATUS(<environment>)	request/transfer
FILE-CONTROL	<filename>:PROBLEM(<task>)	verify task
FILE-CONTROL	<filename>:IMPACT(<task>)	save/transfer
FILE-CONTROL	<filename>:EXECUTE(<filename>)	EXAMINE
FILE	<filename>:FILE-TYPE(<name>)	entry field
FILE	<filename>:FILEKIND(<name>)	entry field
FILE-CONTROL	<filename>:FORMAT(<tpsformat>)	EXAMINE
FILE-CONTROL	<filename>:FOUND(<find request>)	FIND
FILE	<filename>:FUNCTION(<name>)	entry field
FILE-CONTROL	<filename>:LIBRARY(<library name>)	EXAMINE
FILE-CONTROL	<filename>:IMPORT(<procedure name>)	EXAMINE
FILE-CONTROL	<filename>:EXPORT(<procedure name>)	EXAMINE
FILE-CONTROL	<filename>:MAINTAINED(<user-id>)	checkin
FILE-CONTROL	<filename>:OBJECT-HOST(<hostname>)	ENTER/NEW/entry
FILE-CONTROL	<filename>:OBJECT-PACK(<packname>)	ENTER/NEW/entry
FILE-CONTROL	<filename>:OBJECT-USERCODE(<user-id>)	ENTER/NEW/entry
PROJECT	<projectname>:PROJECT(<filename>)	ENTER/NEW
FILE-CONTROL	<filename>:REPLACED(<replace request>)	REPLACE
FILE-CONTROL	<filename>:SOURCE-PACK(<packname>)	Checkout
FILE-CONTROL	<filename>:SOURCE-USERCODE(<user-id>)	Checkout
FILE-CONTROL	<filename>:TRBASE(<tpsbase>)	EXAMINE
FILE-CONTROL	<filename>:ADDS-Dictionary(<adds-dic>)	EXAMINE
FILE-CONTROL	<filename>:ADDS-PROGRAM(<adds-prog>)	EXAMINE
FILE-CONTROL	<filename>:ADDS-FILE(<adds-file>)	EXAMINE
FILE-CONTROL	<filename>:ADDS-GROUP(<adds-group>)	EXAMINE
FILE	<filename>:STOR({timestamp})	Check-in
FILE-CONTROL	<filename>:CHANGED({timestamp})	Check-in
FILE-CONTROL	<filename>:COMPILE-STATUS(<status>)	Check-in/COMPILE
PROBLEM	<taskname>:SETTLEMENT(<user>)	Add task
PROBLEM	<taskname>:PROBLEM-TYPE(<problem-type>)	Add task
PROBLEM	<taskname>:RECEIVED({timestamp})	Add task
PROBLEM	<taskname>:STATUS(<taskstatus>)	
PROBLEM	<taskname>:SUB-STATUS(<sub-status>)	
PROBLEM	<taskname>:READY({timestamp})	Task solved
Etcetera		

## 13.3.(Query) Query Tool Windows GUI

The query dialog.



### Object

The following objects are currently available: "File" and "Task."

- Choose "File" to select the files that meet the section criteria.
- Choose "Task" to select the tasks that meet the selection criteria.

### Macro

The Macro drop-down box shows all the macros that are defined in SURE. A macro is a pre-defined query. If a macro is selected, then the attributes of that macro are placed in the appropriate fields (Object, Wildcard, Limit, and Criteria). These fields can then be customized to the user specific needs. So, the macro is in this case used as an example-query.

### Category

This field shows the relation classes that can be selected by the user. Usually, this is a subset of the classes used by the SURE software.

Depending on the type of the object (file or task), the content of this category is retrieved from the list defined under the Configuration menu, select Query Category for Files OR Configuration menu, select Query Category for Tasks.

Frequently used categories are:

- For Task: PROJECT, PRIORITY, ANNOUNCED, STATUS, PROBLEM-TYPE, and so on.
- For File: PROJECT, FILE-TYPE, FILEKIND, STATUS, DATASET, FUNCTION, so on.

### Type

This drop-down box depends on the value of the category and it contains all the possible values of the entered category. For example, if the category = PRIORITY, then the type drop-down box contains all the possible priorities.

### Advanced

If advanced mode is off then the category and the type determine the query. For example, if category = PRIORITY and type = HIGH, then the query will select all objects (files of tasks) with a priority = high.

If the query interface is shown in advanced mode then the category and type are only for assistance in selecting values to create proper selection criteria; choose the correct category or type and press the button "add subset" to add a new selection expression to the criteria. The executed query is the one shown in the criteria edit box. The values in the category and type are not used in the query execution.

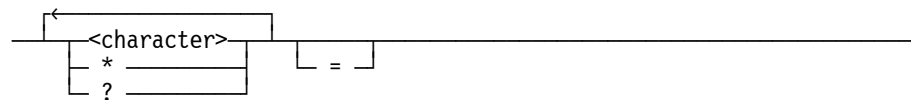
In the example screen, advanced mode is turned on. The executed query = `STATUS(SOLVED)`. The entered category and type are not used.

### Limit

This field limits the amount of selected files or tasks. If no limit is entered, then all files or tasks that meet the query expression are selected.

## 13.4.(Query) Query Syntax

### Wildcard

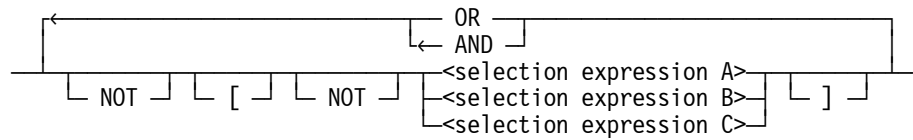


\* : the "\*" can be replaced by an undefined number of characters.

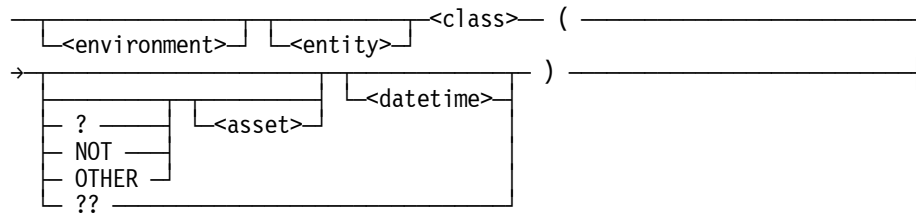
? : the "?" must be replaced by one character.

= : each name must start with the wildcard.

### Criteria

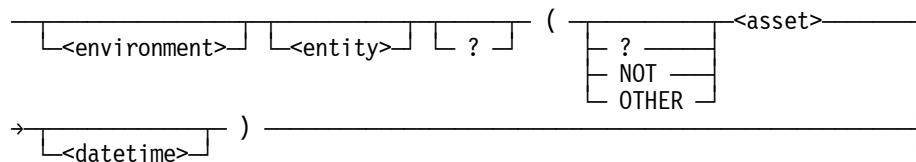


### <selection expression A>



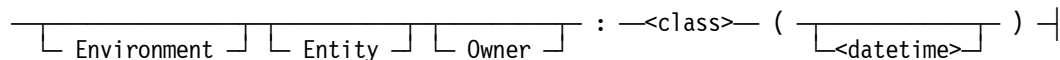
Use this construct if you want to select files or tasks with a specific class or with a specific class and asset.

### <selection expression B>



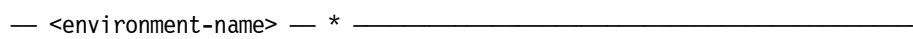
Use this construct if you want to select files or tasks that do not have the specified asset.

### <selection expression C>



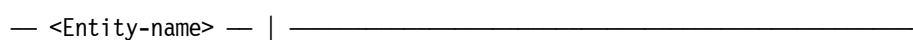
Use this construct if you want to select files or tasks that are used in a relation as an asset.

### <Environment>



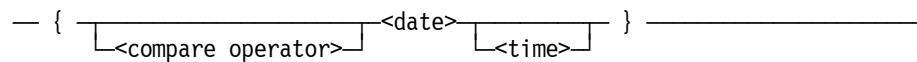
Use this construct if you want to select the relation from an environment that differs from your current environment.

### <Entity>



Use this construct if you want to select the relation from an entity that differs from the default entity (file or task).



**<Datetime>**

- <Compare operator>: GEQ, LEQ, >, <, = (GEQ = greater or equal; LEQ = less or equal)

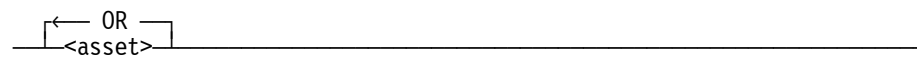
The <compare operator> clause is optional, the default is = (equal)

- <Date>: yyyyymmdd. For example, 20090408 = April 8, 2009
- <Time>: hh OR hh:mm OR hh:mm:ss. For example, 12:01:30

The <time> clause is optional, and the default depends on the <compare operator>: for operators LEQ and >, the defaults are: hours = 23, minutes = 59, seconds = 59. For the other operators the defaults are 0.

**Example**

- COMPILE-STATUS (COMPILED{LEQ 20090509 12}) means:  
select all files that are compiled before May 5, 2009 at 12:59:59.
- COMPILE-STATUS (COMPILED{GEQ 20090509 12}) means:  
select all files that are compiled after May 5, 2009 at 12:00:00.

**<Asset>****<selection expression>**

Select a file or RIS-entity that has a relation equal to <selection expression>.

**NOT <selection expression>**

Select a file or RIS-entity that does not have any relation equal to <selection expression>.

The first "?" in the selection criteria displays the asset of selected relations.

The "??" can be used to show all assets of a class, for example DATASET( ?? ).

**Criteria**

Selection expressions can be combined with the AND, OR, and NOT operators and with the use of square brackets.

### Examples with Selection Expression Type A

In all examples the selection type = FILES.

- |    |                              |  |
|----|------------------------------|--|
| 1. | CLASS(ASSET)                 | For example, STATUS(PRODUCTION)<br>Select all files that have production status.                                   |
| 2. | CLASS(ASSET1 OR<br>ASSET2)   | For example, DATASET(ACCOUNT OR<br>CUSTOMER)<br>Select all files with dataset account or dataset<br>customer       |
| 3. | CLASS()                      | For example, DATASET()<br>Select all files that use a dataset.   |
| 4. | CLASS(ASSET{DATE})           | For example, COMPILE-<br>STATUS(COMPILED{>921001})<br>Select all files that are compiled since 1 October<br>1992.  |
| 5. | ENVIRONMENT*CLASS(A<br>SSET) | For example, PRODUCTION*COMPILE-<br>STATUS(SYNTAX)<br>Select all files with compile-status syntax in<br>production |
| 6. | NOT CLASS(ASSET)             | For example, NOT DATASET(CUSTOMER)<br>Select all files that do not use the CUSTOMER<br>dataset.                    |
| 7. | CLASS(NOT ASSET)             | For example, DATASET(NOT CUSTOMER)<br>Select all files that use a dataset, and not the<br>CUSTOMER dataset.        |
| 8. | CLASS(OTHER ASSET)           | For example, DATASET(OTHER CUSTOMER)<br>Select all files that use a dataset other than the<br>CUSTOMER dataset     |
| 9. | NOT CLASS()                  | For example, NOT DATASET()<br>Select all files that do not use any dataset at all.                                 |

**Examples with Selection Expression Type B**

- |                   |  |
|-------------------|--|
| 10. (ASSET)       | For example, (DJONES)<br><br>Select all files where DJONES is the asset in one of the relations. The name DJONES can, for example, be linked to relations with class AUTHOR, ANALYST, MAINTENANCE, REQUESTED or SOURCE-USERCODE. |
| 11. (ASSET{DATE}) | For example, (PRODUCTION{>931225})<br><br>Select all files which were transferred to production after 25 December 1993   |
| 12. NOT(ASSET)    | For example, NOT(PRODUCTION)<br><br>Select all files which do not have production status and which are not in the compile or transfer queue for production.  |

**Examples with Selection Expression Type C**

- |                         |  |
|-------------------------|--|
| 13. OWNER:CLASS()       | For example, BASE:SYSTEM()<br><br>Select all files which are part of system BASE<br><br>For example, RESPECT/SURE:COPYFILE()<br><br>Select all copy files of program RESPECT/SURE. |
| 14. OWNER:CLASS({DATE}) | For example, ACCOUNT:PROJECT({<940101})<br><br>Select all files that were already part of project ACCOUNT before 1 January 1994.   |
| 15. NOT OWNER:CLASS()   | For example, NOT BASE:SYSTEM()<br><br>Select all files that belong to a system not equal to BASE.  |

All the above examples show the selection criteria where only one selection expression is used. It is also possible to define selection criteria with multiple selection expressions. The AND and OR operators and the square brackets (instead of parenthesis), can be used to combine a multiple selection expression into one selection criterion.

### Example

- |   |   |
|---|---|
| 16. CLASS-1(ASSET-1) AND<br>CLASS-2 (ASSET-2)                           | For example, DATASET(CUSTOMER) and<br>DATASET (ACCOUNT)<br><br>Select all files that use datasets customer and<br>account.  |
| 17. OWNER-1:CLASS-1() AND<br>[CLASS-2(ASSET-2) OR<br>CLASS-3 (ASSET-3)] | For example, BASE:SYSTEM() AND<br>[SOURCE-USERCODE(SG) OR REQUESTED(SG)]<br><br>Select all files in system BASE that are in<br>maintenance or requested by user SG. |

It is allowed to combine AND and OR operators in one selection criterion, but an OR operator cannot be followed by an initial open bracket. AND operators have a higher priority than OR operators.

### Example of Incorrect Usage

- |  |  |
|--|--|
| 18. SELECTION-EXPRESSION-1<br>or [SELECTION-<br>EXPRESSION-2 AND<br>SELECTION-EXPRESSION-3 ] | In this example an initial bracket is preceded<br>by the OR operator. The square brackets are<br>meaningless, and therefore the selection<br>criterion is rejected |
|--|--|

### Examples of Correct Usage

- |   |  |
|---|--|
| 19. SELECTION-EXPRESSION-1<br>OR SELECTION-<br>EXPRESSION-2 AND<br>SELECTION-EXPRESSION-3   | This is the correct expression for example 15.   |
| 20. SELECTION-EXPRESSION-1<br>AND [SELECTION-<br>EXPRESSION-2 OR<br>[SELECTION-EXPRESSION-3<br>AND SELECTION-<br>EXPRESSION-4 ] ] | In this example an open bracket is preceded<br>by the OR operation, but it is not an initial<br>open bracket (it is a nested one). |

### Dynamic Values in the Selection Expression

Dynamic values are extremely handy in macros, but they can also be used in an ad hoc query. A dynamic value is replaced by an actual value, just before the macro or query is executed. The actual value depends on the user who executes the macro or query, on his work schedule, and on the current date.

The following dynamic values are available.

MY-TASK	<p>MY-TASK is replaced by &lt;my current task&gt; when the macro or query is executed.</p> <p>For example, if my current task is TSK0025 and the expression is PROBLEM (MY-TASK), then all files linked to task TSK0025 are selected.</p>
MY-USERID	<p>MY-USERID is replaced by &lt;my logon usercode&gt; when the macro or query is executed.</p> <p>For example, if my logon usercode is SIMON and the expression is REQUESTED (MY-USERID), then all files requested by SIMON (=myself) are selected.</p>
MY-PROJECT	<p>MY-PROJECT is replaced by &lt;one of the projects of my-project-list&gt; when the macro or query is executed.</p> <p>For example, if PROJ-A and PROJ-B are the projects in my project list, and the expression is PROJECT (MY-PROJECT) AND STATUS (DEVELOP), then all files with STATUS = DEVELOP and (PROJECT = PROJ-A or PROJECT = PROJ-B) are selected.</p>
MY-TEAM	<p>MY-TEAM is replaced by &lt;one of the teams of my-team-list&gt; when the macro or query is executed.</p> <p>For example, if TEAM1 and TEAM2 are the teams in my-team-list, and the expression is SETTLEMENT (MY-TEAM), then all tasks with (SETTLEMENT = TEAM1 or SETTLEMENT = TEAM2) are selected.</p>
MY-TEAMMEMBERS	<p>MY-TEAMMEMBER is replaced by &lt;one of the members of a team from my-team-list&gt; when the macro or query is executed.</p> <p>For example, if TEAM1 and TEAM2 are the teams in my-team-list, and users SIMON, JAN, GERARD, FRANK, and MARCO are members of one of those team, and the expression is SETTLEMENT(MY-TEAMMEMBER), then all tasks with (SETTLEMENT = GERARD or SIMON or JAN or FRANK or MARCO) are selected.</p>
MY-TEAMMEMBER	
MY-EMPLOYEE-FUNCTION	<p>MY-EMP-FUNC is replaced by &lt;one of my employee function&gt; when the macro or query is executed.</p> <p>For example: if SYS1-DEVELOP and SYS2-DEVELOP are my employee-functions and the selection expression is SETTLEMENT (MY-EMP-FUNC), the all tasks with (SETTLEMENT = SYS1-DEVELOP or SYS2-DEVELOP) are selected.</p>
MY-EMP-FUNCTION	
MY-EMP-FUNC	

TODAY	<p>TODAY can be used in the date expression, and will be replaced by today's date when the macro or query is executed.</p> <p>For example, the expression CHANGED ({TODAY}) selects all files that are changed today.</p>
TODAY - <days>	<p>TODAY can be used in the date expression, and will be replaced by today's date when the macro or query is executed.</p> <p>For example, the expression CHANGED ({&gt;TODAY-2}) selects all files that are changed since 2 days ago.</p>
THIS-WEEK	<p>THIS-WEEK can be used in the date expression, and will be replaced by the first day of this week (Sunday) when the macro or query is executed. If THIS-WEEK is not preceded by a &gt;, &lt; or = then &gt; is used as default.</p> <p>For example, CHANGED ({THIS-WEEK}) selects all files that are changed since the beginning of this week.</p>
THIS-MONTH	<p>THIS-MONTH can be used in the date expression, and will be replaced by the first day of this month when the macro or query is executed. If THIS-MONTH is not preceded by a &gt;, &lt; or = then &gt; is used as default.</p> <p>For example, CHANGED ({THIS-MONTH}) selects all files that are changed since the beginning of this month.</p>
THIS-YEAR	<p>THIS-YEAR can be used in the date expression, and will be replaced by the first day of this year when the macro or query is executed. If THIS-YEAR is not preceded by a &gt;, &lt; or = then &gt; is used as default.</p> <p>For example, CHANGED ({THIS-YEAR}) selects all files that are changed since the beginning of this year.</p>

### 13.5. (Query) Macro Facility for the Select Function

With the MACRO facility, it is possible to declare common or frequently used selection expressions that can be used later on by inexperienced users (From the Macro folder click New).

It is also possible to define site-specific workflow macros.

The Windows GUI shows a Macro folder with all possible macros. Using properties, these macros may be maintained. Opening the macro (by pressing the + sign) executes the macro.

Macros can be defined for an individual user, for a team, or for a role, (employee-function), to limit the visibility of a macro.

### 13.5.1. (Query) #SINCE in the MACRO Definition

The #SINCE construction is valid within a DATE expression. This #SINCE construct allows creation of a query that selects all relations with a timestamp that changed since the last time that the macro was executed. Therefore, it can give a new list of changed files since the last time that this macro was executed.

#SINCE is only applicable in Macros and not in ad hoc Queries.

#### Example

Macro definition: "files CHANGED ({#SINCE})" will give all the files changed since the last time that the macro was executed.



#### Technical Details and Considerations

The information about the #SINCE information is stored in the INI file according to the following example:

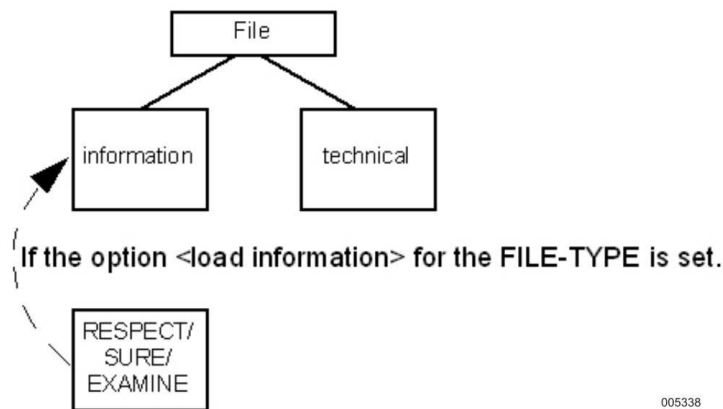
```
[ SINCE ]
SELECT-MACRO-0011=20050202112037
```





## Section 14

# Text as Information/Documentation



### 14.1. Overview

The SURE repository contains various information storage mechanisms, which allow storing textual information attached to a file. Currently, this information is divided into three separate types, INFORMATION, REMINDER, and ATTACHMENT. Using the Windows GUI, it is also possible storing PC files such as Word files or Visio drawings for a project. This means that the documentation is stored along with the actual sources.

#### INFORMATION

This is often programmer or operation information describing the program functions or flow. This information is versioned and promoted to other environments if the task is promoted.

Information will be loaded automatically when a source is checked in, if the option <load information> is set for the FILE-TYPE of that source. Setting this option causes the comment lines of a ClearPath server source to be loaded in the repository as INFORMATION.

#### REMINDER

This information is informal information, which is used as communication between programmers or as a yellow memory paper. The RIS generators also create this kind of information, but then it contains generated syntax errors or warnings. This

information is not promoted to other environments. The information is versioned; the contents may differ in different environments.

### **ATTACHMENT**

This information can contain detailed analysis in the form of a Windows file. This attachment is not versioned and therefore equal in every environment. If you require versioned information, a separate PC file must be loaded for the project.

## **14.2. Information, Reminder, and Attachment**

Select from the file properties, or from the speed-menu (a right-click on the file name), the choices Information, Reminder, and Attachment.

From the file properties maintenance menu, or from the speed-menu, the choices Modify Information, Modify Reminder, and Modify Attachment are present. Select one of these to modify the information.

The notepad function allows creating stream files that are stored in the repository structure. These stream files are converted to the original text format (72 characters each line) if they are updated using the Terminal Emulation interface. If one decides to update and view this information using the Windows interface only, there are no restrictions in using the notepad. Otherwise, it is advised to stick to 72 characters each line.

Closing the notepad will verify if the modification needs to be stored. Acknowledgement will store the information in the repository.

The user object allows changing the language in which the information is updated.

## Section 15

# Copy Files

For Unisys ClearPath servers, the use of copy files and other resources is integrated with the file location algorithm and the security implementation on the ClearPath server.

There are two distinct ways in which SURE can approach resource handling, depending on the option <Use resource versions> in the SYSTEM definition for the source file:

- If option "Use resource versions" is not set, the source file will retrieve its copy files from a single central location.
- If option "Use resource versions" is set, a number of different resource types are distinguished that can be stored in several locations, depending on task, project, system, and source file dependent settings.

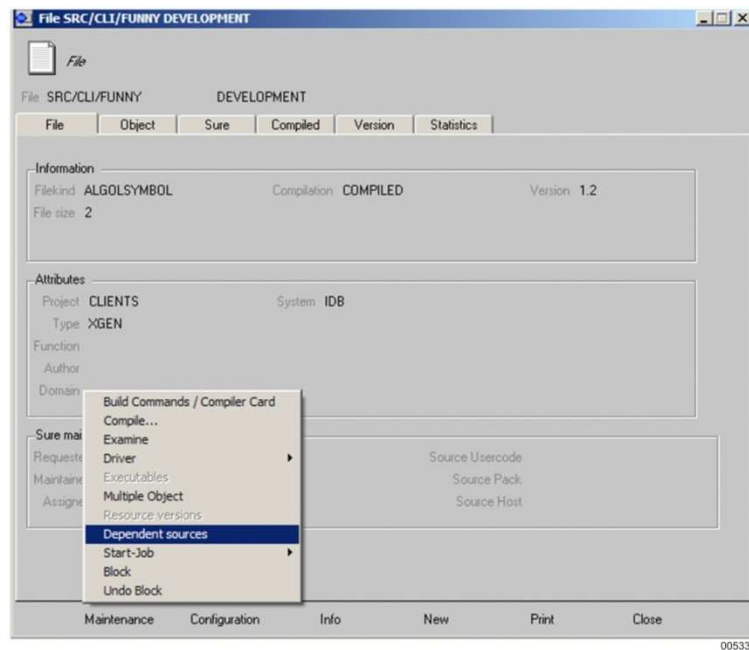
These two ways of working can hardly be combined in a controllable manner, so the <Use resource versions> option should only be set for a system when it is newly set up. In addition, links between sources from different systems, of which only one has the resource option set, should be avoided.

### 15.1. Dependent Files

Copy files are automatically recognized by SURE. The examine process scans sources for COPY and INCLUDE statements and adds a copy file relation between the source and each of its copy files.

It is also possible to define dependencies between files manually using the SUREforWindows option, from file-properties select configuration, and click Dependent sources.

Files that are marked as dependent are called dependent files. Effectively, these files are treated by SURE exactly the same as copy files. The only difference is that the relation between the source and the dependent file is added manually. Copy files are linked to a source using a COPY FILE relation. Dependent files are linked to the source using a DEPENDENT relation.



The effect for the dependent relation is:

### MCP files

When the SURE compile runs, the dependent files are extracted from the SURE repository and placed in the SURE batch environment where the compile runs, similar to what SURE does for copy files.

### PC files

For PC files this relation is used in the build process.

If a BUILD.BAT file has DEPENDENT relations to other BUILD.BAT files, these DEPENDENT builds are executed first.

If a BUILD.BAT file has DEPENDENT relations to another file which is not a BUILD.BAT file, then this file is added in the compile queue.

## 15.2. (Copy files) Normal Mode

If copy files are used in a program, the appropriate syntax for each language will be parsed by the RESPECT/SURE/EXAMINE program. The used copy files will be linked to the source using a COPY FILE relation. These relations are used by the SURE software.

The examine program will run asynchronously from the end user interface; therefore, the relations may not match the contents of the source at a given time. The RESPECT/SURE/EXAMINE program is triggered by a file change and is initiated automatically by SURE. The program examines the file twice, the first time immediately after the file change where it just extracts the copy file relations. The second run is performed in the SURE batch environment. At that time, a total examine is performed.

Above mentioned mechanism uses the capacity of the computer optimally, since the total examine is done in evening hours, while the copy file relations needed to correctly perform, some SURE functions are updated as soon as possible after a source is changed.

### Example

```
File TEST/1
  100 BEGIN
  200 $ INCLUDE "COPY/A"
  300 $ INCLUDE "COPY/B"
  400 END.
```

The RESPECT/SURE/EXAMINE program creates the following relation when this file is scanned:

```
FILE-CONTROL|TEST/1: COPY-FILE(COPY/A):
FILE-CONTROL|TEST/1: COPY-FILE(COPY/A):
```

The use of copy files from a central location requires some special handling to make it possible that different programmers can access the same files. For this reason, the SYSTEM definition contains a number of options that control the handling of copy files.

## 15.2.1. (Copy files) Work-Environment

This option identifies the CANDE directory where a user has to be logged on when he wants to work on a task (updating sources that are loaded in SURE). The work-environment defines also the location of source files that are retrieved from SURE and the location of copy files.

It is important that the work-environments are correctly defined for all repository environments; otherwise, it will almost be impossible for the programmers to do their jobs correctly. One of the functions of the work-environment is that it identifies the disk location where changed copy files are placed. These copy files on disk will be used by programmers when they do a test compilation of a source that they have in maintenance. It is obvious that the version of a copy file can change per SURE environment. Nevertheless, the programmer expects that the correct versions of all copy files are available on disk; otherwise, he cannot compile his source correctly.

Option <put copy files in the work-environment after a save or transfer> enforces that copy files that change in an environment (by a save or by a transfer) are copied to the corresponding CANDE work-directory of that environment.

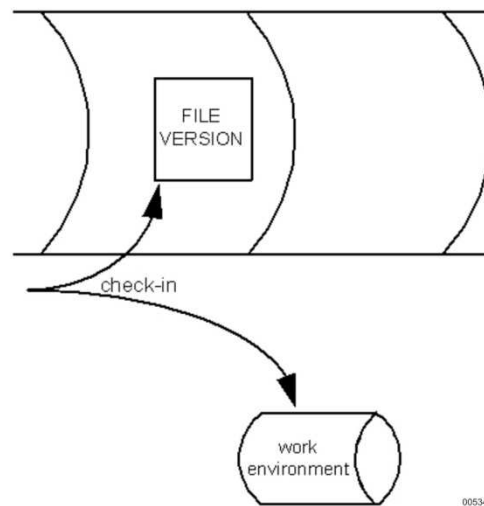
A user can only retrieve a source from SURE (function GET) if he is linked to a current task. This current task identifies the SURE environment from where the source must be retrieved. A work-directory must be defined for this SURE environment; otherwise, SURE will give an error message (work-environment not correctly defined for this system). SURE will place the file on disk in the defined work-directory. This procedure enforces that the source is placed on disk in a directory where the correct copy files are visible. When the programmer wants to edit this source using CANDE, he has to be logged on in that directory too.

A work-environment has to be defined for each application system and for every environment.

See “Work-Environment” in Section 36.

The Windows interface connects the users to the API running against a repository. This implies that the API is running for all users and there is no individual copy of the user interface program for each user. For this reason, the files are created in the defined work-environment for a system.

### 15.2.2. (Copy files) <Put Copy Files in the Work Directory>



This system option instructs the SURE software to copy a file to the defined work directory after the file has been changed, if this file is known by SURE as a copy file.

A file is known by SURE as a copy file if this file is currently in use as a copy file by one or more programs or if this file has a “copy file” file type.

This option is often used in the development environment, where the work directory contains all copy files used by the programmers. If a copy file is in maintenance or checked out, it is resident under the usercode maintaining the file. Under that usercode, the file is privately accessible by the maintenance user and cannot be seen by other users. After completing maintenance and checking in of the file, it is copied to the work directory as a public accessible file. In the work directory, the file can be used by all programmers.

The integrity of the development environment cannot be guaranteed if this option is not set. Side effects, like copies of possible different versions of copy files in different user directories, are a result of resetting this option. For this reason, it is advised to set this option for the development environment.

This option is usually not set for non-maintenance environments. The reason for this is obvious: programmers do not adapt sources in these non-maintenance environments, and therefore it is not necessary to keep the copy files on disk. (This saves some processor power and disk space).

It may happen that a source is adapted in a non-maintenance environment for quick fix purposes. SURE checks if the correct versions of copy files are available on CANDE if the programmer retrieves a source from the repository. Copy files that are not resident on disk will be copied to the usercode of the programmer, so that he can compile his program correctly. When the programmer saves the source back into SURE, then the downloaded copy files will be removed from his usercode (if they are not in use by another source that is in maintenance by the programmer).

### **15.2.3. <Use Never Object Relation>**

The FILE-TYPE definition contains the option <Use never object-relations>, which instructs SURE not to link any object-location to a file of that file type. As a result, these files will never be compiled by SURE compile procedure. This attribute should be set for copy files.

### **15.2.4. (Copy files) <Allow Copy Files From "\*">**

The "SURE compile options" definition contains the option <Allow use of copy files from star (\*)>.

The RESPECT/SURE/COMPILE program checks during its initialization which copy files are needed for the compilation process. If a copy file is not yet resident on disk, then the RESPECT/SURE/COMPILE program will copy it temporarily from SURE to disk. When all compilations are done, the downloaded copy files will be removed from disk.

If the above-mentioned option is set, a copy file may be resident under directory "\*" or it may be resident under the usercode where the RESPECT/SURE/COMPILE program is running.

If the above-mentioned option is reset, the \* directory is ignored. A copy file will then be downloaded from SURE if it is not resident under the usercode where the compile program is running.

### **15.2.5. (Copy files) Copy or Include Statements**

The copy file statements in the programs should not contain usercode or family syntax in the file name. If these statements contain family or usercode syntax, versioning of these copy files is complicated or impossible.

A number of mechanisms play a role in the handling of copy files:

- The MCP will search for a copy file using the rule, first search in the USERCODE directory on the PRIMARY and ALTERNATE FAMILY and secondly without usercode "\*" on the PRIMARY and ALTERNATE FAMILY.
- The batch SURE compile process will copy the copy files that are to be used under its own USERCODE and on its PRIMARY FAMILY if required.
- The SURE software may copy a checked in copy file to the work directory, if the system option <Put copy files in the work directory after save or load> is set.

These three mechanisms create some limitations on the syntax and use of copy files, which can best be explained by some examples.

### Example

File TEST/1

```
100 BEGIN
200 $ INCLUDE "COPY/A"
300 $ INCLUDE "COPY/B"
400 END.
```

File COPY/A

```
100 ARRAY A[0:100]
```

File COPY/B

```
100 ARRAY B[0:100]
```

Environment Development	Environment Production
WORK-USERCODE = *	WORK-USERCODE = QUICKFIX
WORK-PACK = DEV	WORK-PACK = DEV
WORK-HOST = DEVHST	WORK-HOST = DEVHST
<Put the copy files ....> = SET	<Put the copy files ....> = RESET
<Do not use copy files from *> = RESET	<Do not use copy files from *> = SET

- Checkout TEST/1 in development will verify the presence of COPY/A and COPY/B in the work directory. If these files are not present, they are copied into the work directory (or the option copy include files is set in the CHECK OUT command).
- The programmer working for TEST/1 uses the files COPY/A and COPY/B from the work directory.
- Checkout COPY/A places the maintenance copy for COPY/A in the usercode directory for the programmer. After checking in of this file, SURE copies it to the work directory and makes it available for all programmers. The SURE batch compile program will compile all programs that have a reference to this file.



- The SURE compile batch for development will use the copy files in the work directory that is those that match the last checked in versions by the programmers.
- The SURE compile batch for production will copy the copy files in its running directory, so that they can be used by the compiler.

### Example

```
File TEST/1
  100 BEGIN
  200 $ INCLUDE "(COPYLIB)COPY/A ON ICLPACK"
  300 $ INCLUDE "(COPYLIB)COPY/B ON ICLPACK"
  400 END.
```

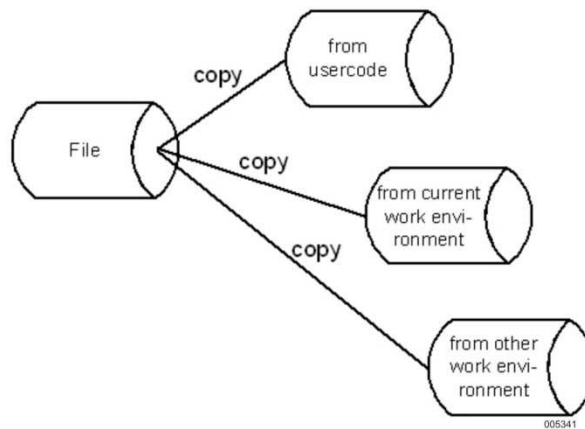
```
File COPY/A
  100 ARRAY A[0:100]
```

```
File COPY/B
  100 ARRAY B[0:100]
```

Environment Development	Environment Production
WORK-USERCODE = * WORK-PACK = DEV WORK-HOST = DEVHST <Put the copy files ....> = SET <Do not use copy files from *> = RESET	WORK-USERCODE = QUICKFIX WORK-PACK = DEV WORK-HOST = DEVHST <Put the copy files ....> = RESET <Do not use copy files from *> = SET

- The include-statements in the program do not match the SURE definitions, because the include-statement refers to usercode COPYLIB on ICLPACK, while the work-directory in SURE is defines as \* on DEV. For this reason the organization must create its own mechanism to install the proper versions for the include files in the directory (COPYLIB) = ON ICLPACK.
- The production environment must contain a compile replace table that replaces (INCLUDE) and ICLPACK. Otherwise, the same versions for this copy file are used in development and in production.

### 15.3. (Copy files) <Use Resource Versions>



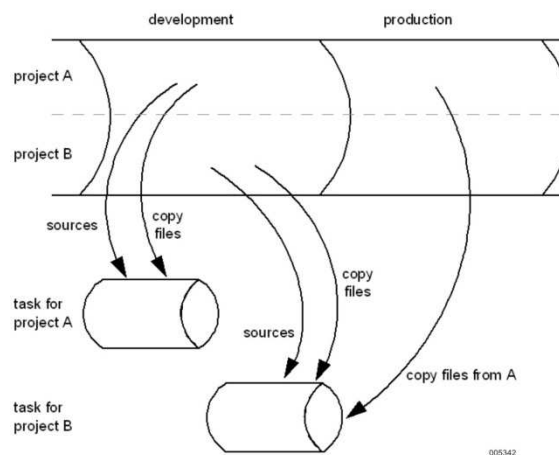
If multiple projects are developed simultaneously, the common location of copy files under the "\*" directory may not be satisfactory. As is explained, this directory is refreshed if a file is checked in. For this reason, the integrity of this development environment is not guaranteed between the check-in and the batch-compile. The second complexity is introduced if different projects are using the functions of each other. If so, they are using the newest changes while the last promoted versions are the stable ones.

#### Example

Consider the following case, Project A is scheduled for a new release in one year. Project B is to be released in one month. Project B uses some sources from project A.

In this case, project B should retrieve the sources, belonging to project A, from the production environment of project A, while it can use its own sources from the development environment.

Project A, however, should use all its sources from its development environment.



Using the <Use resource versions> option allows for such a flexible approach of locating resources. In this approach, each system has a separate resource location defined for each repository environment. Copy files and other resources will be retrieved from one of these locations, depending on a number of definitions.

### 15.3.1. (Copy files) Resource Statements, Syntax, and Qualification

There are three types of resources that can be handled by SURE: copy file, library, and ADDS dictionary. Each of these resources is recognized only when it appears in a statement, or set of statements, satisfying certain syntax, which may be more limited than the syntax recognized by the compiler.

When a file is copied from the repository, statements with a supported syntax will be "qualified:" the original resource name will be replaced according to the rules defined for that particular type of resource. If qualification of a statement causes line overflow, SURE tries to fit the record by shifting it left as far as possible. If there is no way to fit it, no replaces will be done, since splitting the source line could cause unwanted effects, like the resequencing of large parts of the source.

Therefore, it is a good habit to isolate resource statements as much as possible.

When a resourced file is stored into the repository, the same syntax rules are applied to find the resource statements, and the qualifications are undone before the file is actually saved.

The resource qualification algorithm only recognizes resource statements if their significant parts are contained in a single source line. The RESPECT/SURE/EXAMINE program, however, will also recognize statements spanning multiple lines. To avoid unwanted side effects in this respect, resource statements should always be kept on a single source line.

#### COPY FILE

COBOL source in database	<u>COPY</u> " <u>copy-file-name</u> "
ALGOL of JOB source in database:	<u>\$ INCLUDE</u> " <u>copy-file-name</u> "
source on disk	COPY "(usercode)copy-file-name ON pack"
or	\$ INCLUDE "(usercode)copy-file-name ON pack"
resource relation	<source-file> - COPY-FILE - <copy-file-name>

### LIBRARY

COBOL source in database	<u>CALL</u> " <u>identifier</u> <u>IN</u> object-name"
	<u>CALL</u> " <u>identifier</u> <u>OF</u> object-name"
source on disk	<u>CALL</u> "identifier <u>IN</u> (usercode)object-name ON pack" <u>CALL</u> "identifier <u>OF</u> (usercode)object-name ON pack"
resource relation	FILE-CONTROL <source-file>:LIBRARY(<source of object-name>)

### DICTIONARY

COBOL source in database:	<u>SPECIAL-NAMES.</u> <u>DICTIONARY</u> IS " <u>dictionary-name</u> " <u>PROGRAM-NAME</u> IS " <u>program-name</u> " <u>PROGRAM-VERSION</u> IS program-version ] <u>PROGRAM-DIRECTORY</u> IS " <u>directory-name</u> " (in this statement order only)
source on disk:	PROGRAM-DIRECTORY IS "resource-prefix dictionary-name"

example	database	DICTIONARY "DICT1"  PROGRAM-DIRECTORY "DIR2"  DICT1 is dictionary for system SYS1  DIR2 is resource from environment DEVELOP  RESOURCE-PREFIX for system SYS1 on DEVELOP = "DEV1"
	disk	PROGRAM-DIRECTORY "DEV1DIR2"
resource relations		FILE-CONTROL <source-file>:ADDS-DICTIONARY(<dictionary-name>)  FILE-CONTROL <source-file>:ADDS-PROGRAM(<program-name>)  FILE-CONTROL <source-file>:ADDS-PROG-DIR(<directory-name>)

**Note:** The program-name must be related (manually) to a project; directory-name will be automatically related to the same project.

### 15.3.2. (Copy Files) Resource Environment

If the <Use resource versions> option is set for a system, the Resource Environment should contain a location (usercode, family, hostname) where copy files and other resources can be found.

When a file is copied from the repository, its resources will be qualified according to the resource environment of the resource's SYSTEM for the repository environment that the resource is to be retrieved from.

At check in of a copy file it will automatically be copied to the resource environment of its system in the current repository environment. If this copy file contains resources they will all be qualified to point to that same environment, thus ensuring that the copy file itself uses only stable resources, so that any file using it as a copy file can consider it to be a single and stable resource.

### 15.3.3. (Copy Files) Resource Prefix

The ADDS-DICTIONARY resource type is not qualified using a usercode, family, and hostname.

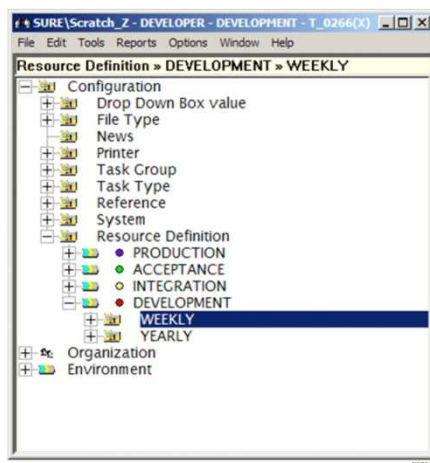
Instead, the original resource name is prefixed with a string, defined in the Resource Prefix field. This allows the use of a distinct set of dictionaries, one for each SYSTEM on each repository environment.

### 15.3.4. (Copy Files) Resource Definition

A resource definition is a table that defines a set of default resource versions.

In a resource definition, a default resource version can be assigned per project, per repository environment. It can be linked to a task or a project, making all sources, maintained through this task or project, assume the default destinations stated in the definition table.

Resource definitions are maintained using a folder in the Resource definition under Configuration.



### 15.3.5. (Copy Files) Default Resource Version

A default resource version can be assigned to a project. This version will be used for locating all resources that are not of the same project as the source and were not handled by a resource definition (either for the TASK or for the PROJECT).

### 15.3.6. (Copy files) Resource Resolution Algorithm

Resource relations can be divided in two classes.

- The relations that SURE determines at check in. These relations will be used to determine what resource statements are to be considered when the file is processed by SURE (check-out or copy) or the COMPILE batch.
- Once type-1 relations are established for a source and this source is checked out, copied, or compiled, the resources in this file should be qualified. For every type-1 relation encountered for a source file, the resource version to be used for qualification is determined through a set of rules, applied in a particular order, until one is satisfied. The resulting resource version is stored in a new relation between source file and resource, with class RESOURCE-VERSION. This relation will subsequently be used to qualify the resource. Some on-line SURE commands allow these type-2 relations to be changed by the user, thus creating an accurate means to determine the origin of each individual resource.

A special resource "version" is USERCODE, which implies that the resource will not be qualified, and should be retrieved according to the default visibility rules.

Case	Resource Version
Relation exists: <file>:RESOURCE-VERSION(<resource>)	from relation
The resource belongs to the same PROJECT as the source and it is currently maintained by me (relation: <resource>:SOURCE-USERCODE(<my-userid>))	USERCODE
There is a resource definition linked to my current TASK, that contains an entry for the resource's PROJECT, stating a default version for this project	the stated version
There is a resource definition linked to the source's PROJECT, that contains an entry for the resource's PROJECT, stating a default version for this project	the stated version
The resource belongs to the same PROJECT as the source (but is not in maintenance by me)	my-version
A default RESOURCE-VERSION is defined for the source's PROJECT (relation: <project>:RESOURCE-VERSION(<default-version>))	default-version
my-version = highest level (PRODUCTION)	my-version
else	first higher level with status SOLVED

**Example**

```

File EXAMPLE/SOURCE
000000 ...
005000 ENVIRONMENT DIVISION.
010000 SPECIAL-NAMES.
011000     DICTIONARY "XMPLDICT"
012000     PROGRAM-NAME IS "XMPLPROGRAM"
013000     PROGRAM-DIRECTORY "XMPL".
020000 DATA DIVISION.
025000 WORKING-STORAGE SECTION.
026000     COPY "MY/COPY/1".
027000     COPY "MY/COPY/2".
028000     COPY "YOUR/COPY".
040000 PROCEDURE DIVISION.
050000     CALL "SOME-ID IN OBJECT/MY/LIB".

```

The first check in of this file adds the following relations:

```

FILE-CONTROL|EXAMPLE/SOURCE:ADDS-PROGRAM(XMPLPROGRAM)
FILE-CONTROL|EXAMPLE/SOURCE:ADDS-PROG-DIR(XMPL)
FILE-CONTROL|EXAMPLE/SOURCE:COPY-FILE(MY/COPY/1)
FILE-CONTROL|EXAMPLE/SOURCE:COPY-FILE(MY/COPY/2)
FILE-CONTROL|EXAMPLE/SOURCE:COPY-FILE(YOUR/COPY)
FILE-CONTROL|EXAMPLE/SOURCE:LIBRARY(MY/LIB)

```

The resources are part of the following systems.

Resource	Project	System	
XMPLPROGRAM	BASE	BASE	
XMPL	BASE	BASE	(inherited from XMPLPROGRAM)
MY/COPY/1	MYPRJ	MYSYS	
MY/COPY/2	MYPRJ	MYSYS	Checked out by me
YOUR/COPY	YOURPRJ	YOURSYS	
MY/LIB	3RDPRJ	3RDSYS	

The following definitions have been entered in the system options.

<b>System</b>	<b>Version</b>	<b>Resource Environment</b>	<b>Resource Prefix</b>
BASE	DEVELOPMENT	(DEV) ON BASEPK	BADE
	ACCEPTANCE	(ACC) ON BASEPK	BAAC
	PRODUCTION	(PROD) ON BASEPK	BAPR
MYSYS	DEVELOPMENT	(ME) ON DVLP	MYDE
	ACCEPTANCE	(ME) ON ACC	MYAC
YOURSYS	DEVELOPMENT	(YOU) ON DVLP	YRDE
	ACCEPTANCE	(YOU) ON ACC	YRAC
3RDSYS	DEVELOPMENT	(YOU) ON DVLP	3RDE
	ACCEPTANCE	(YOU) ON ACC	3RAC

Resource definition MYRES0002 for project MYPRJ contains.

<b>Project</b>	<b>Default Version</b>
BASE	PRODUCTION
MYSYS	DEVELOPMENT
YOURSYS	ACCEPTANCE

Resource definition MYRES0001 for task MYTASK contains.

<b>Project</b>	<b>Default version</b>
3RDSYS	ACCEPTANCE

The first checkout of EXAMPLE/SOURCE for MYTASK on environment DEVELOPMENT will determine the following resource versions.

<b>Resource</b>	<b>Version</b>	<b>Rule</b>	<b>Qualified name</b>
XMPL	PRODUCTION	4	BAPRXMPL
MY/COPY/1	DEVELOPMENT	5	(ME)MY/COPY/1 ON DVLP
MY/COPY/2	USERCODE	2	MY/COPY/2
YOUR/COPY	ACCEPTANCE	4	(YOU)YOUR/COPY ON ACC
MY/LIB	ACCEPTANCE	3	(YOU)OBJECT/MY/LIB ON ACC



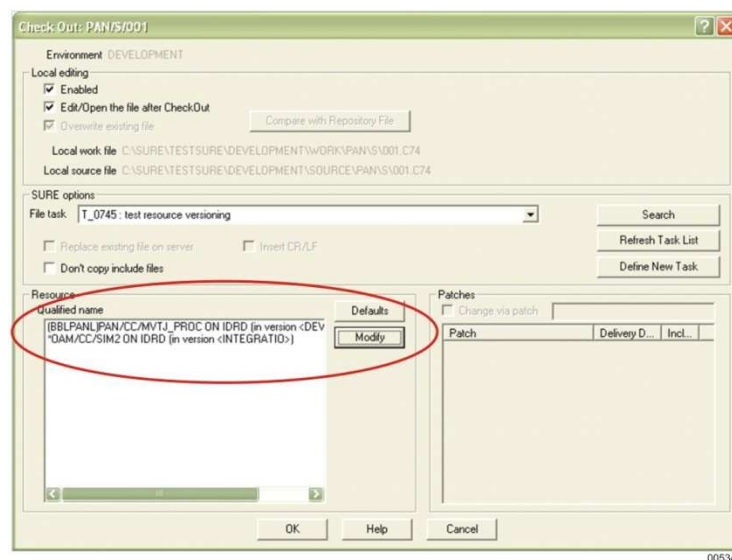
### 15.3.7. (Copy files) Changing Resource Versions

Some on-line SURE commands allow the user to directly choose versions for the resources of a file.

The manual specification of a resource location on the check-out screen covers four possible locations as follows:

- My usercode.
- The resource location of my current environment.
- The resource location of the environment next to my current environment.
- The resource location of the environment where the task gets status solved.

If a file that belongs to a system with resource-versioning-enabled is checked out a special version of the check-out dialog is shown, where you can override the default resource locations.



Each copy file that is used in the source is mentioned in the list of used resources.

- The [defaults] button sets the resource-locations of all resources to their defaults.
- The [Modify] button works only for the resource that is selected in the list and it toggles through four possible resource locations:
  - My usercode.
  - The resource location of my current environment.
  - The resource location of the environment next to my current environment.
  - The resource location of the environment where the task gets status solved.
  - Back to number one.

At check-out, the resources in the source-file are qualified with the resource location. For example, the include-files in COPY statements obtain a temporary usercode and a pack-name that refer to a resource-location. A compilation started by the developer can now successfully find the resources in the correct locations.

At check in the resource qualifications are removed from the COPY statements, so the source-file that is stored in SURE never contains these extra qualifications.

### 15.3.8. (Copy Files) Resource Commands

Activating the "Resource versions" options changes the behavior of some SURE commands. In addition, some commands are only valid if resources are active.

#### **Check-out** **GET**

New resource table allows the user to change these specifications. Then, copies the file from the repository; qualifying the resource names in the resource statements.

#### **Check-out** **GETIT**

This function always uses the existing resource table. If this table is incomplete, it is completed according to the resource resolution algorithm.

#### **Check-In** **SAVE**

This function scans the file to be checked in for resource statements. For any such statement, the qualifications are removed; for new resources, the appropriate relation is added. The resulting file is stored in the repository. If it is a copy file, it is copied to its appropriate resource environment, as is done by the COPY RESOURCE command, described below.

#### **Copy** **COPY**

This function copies file from the repository, qualifying the resource names according to the existing resource table.

#### **Copy, <skip resource replaces>** **COPY NOREPLACE**

This function copies file from the repository, leaving the resource statements unaltered.

#### **-----** **COPY RESOURCE**

This function copies a copy file to its appropriate resource environment, qualifying any resource statements to point to that same environment.

### **Properties/Configuration/Resource versions    RESOURCE**

This function shows the current settings for the resources of a file and allows the user to change them.

#### **-----                                    RESOURCE VERIFY**

This command verifies the resource table against the resource relations.

#### **-----                                    RESOURCE DELETE**

This command deletes the resource table, thus resetting all resources to their default location.

### **15.3.9. Overview: Resource Locations, Definitions, and Usage**

It is possible to create an overview with the definitions of all resource-locations and how the resource definitions are used in a specific file. This overview is handy for debugging purposes.

```
RUN RESPECT/PRINT ("RESOURCE-CONFIGURATION <optional input file>");
```

The following is listed for each environment:

- For each system
  - Option "Use resource versions" Yes/No.
  - The resource location (usercode/pack).
  - For each project of the system the resource table and the default resource environment.
- For each defined resource table
  - The target projects + corresponding resource environments.
- For the (optional) input file
  - All copy files + the corresponding project and resource parameters.

### 15.3.10. Nested Resources in Copy files

SURE offers various methods to qualify “nested” resources. Nested resources are, for example, copy files that are declared within other copy files. This is possible in C85 and ALGOL.

In the case of nested copy statements in a copy file, the copy file itself is placed in the resource-location, and the copy statements inside that copy file are qualified with the usercodes and pack names of their resource-locations. There can be a conflict with those nested qualifications if a nested copy file belongs to a different system than the copy file itself.

The following methods are available, depending on an option:

#### Method 1

- Resources in a copy file in the resource-location are all qualified with that same resource location. This mode is enabled using the following relation:

```
Environment = 0
Entity = OPTION
Owner = RESOURCE-VERSION
Class = OPTION
Asset = PREVIOUS
```

#### Method 2

- Resources in a copy file in the resource-location are qualified according to the same resource-rules as when that copy file is checked out. Resources of a different system are qualified with the resource-location of that other system on the applicable resource environment. This mode is enabled using the following relation:

```
Environment = 0
Entity = OPTION
Owner = RESOURCE-VERSION
Class = OPTION
Asset = RESOURCE-VERSION
```

#### Method 3

- A resource in a copy file in the resource-location is qualified with the default resource location of its own system. This mode is by default enabled when the two other modes are not enabled.

Each method has advantages and disadvantages, but Method 2 has the least disadvantages, as explained in the following example.

**Example**

Consider the following case:

- There are two systems
  - System FO with program FO/PS/00 and copy files FO/CC/03.
  - System FP with program FP/PS/00, and copy files FP/CC/01 and FP/CC/02.

- Development is done in environment DEVELOP.
- INTEGRATION is an environment with stable resources.
- The resource-locations of FP are:

```
DEVELOP :    (FP) ON DEVPK
INTEGRATION: * ON INTPK
```

- The resource locations of FO are

```
DEVELOP:    (FO) ON DEVPK
INTEGRATION: * ON INTPK
```

- Source FP/PS/00 uses copy file FP/CC/01.
- Source FO/PS/00 uses copy file FP/CC/01.
- Copy file FP/CC/01 uses copy files FP/CC/02 and FO/CC/03.

The issue of this example is the way how the nested resources in copy file FP/CC/01 are qualified. Notice that FP/CC/01 is a resource by itself, and therefore resident in the resource location of its system, where it is visible for the compilers.

The resource-location of FP in DEVELOP is only used by sources and copy files in DEVELOP with system = FP.

The resource-location of FP in INTEGRATION is only used by sources and copy files in DEVELOP with system not equal to FP.

**Resource Qualification Using Method 1**

In this case, the nested resources in a copy file in the resource-location are all qualified with that same resource location.

Copy file FP/CC/01 in the resource-location of DEVELOP has the following content:

```
COPY "(FP)FP/CC/02 ON DEVPK"
COPY "(FP)FO/CC/03 ON DEVPK"
```

Both copy statements are qualified with the resource-location of FP in DEVELOP because copy file FP/CC/01 itself belongs to system FP.

**Note:** *The resource-locations of FO and FP in DEVELOP are not equal and that is why copy file FO/CC/03 (which belongs to FO) must be placed manually into this FP-resource-location.*

Copy file FP/CC/01 in the resource-location of INTEGRATION has the following content:

```
COPY ``*FP/CC/02 ON INTPK``  
COPY ``*FO/CC/03 ON INTPK``
```

Both copy statements are qualified with the resource-location of FP in INTEGRATION because copy file FP/CC/01 itself belongs to system FP.

**Note:** *The resource-locations of FO and FP in DEVELOP are equal and that is why copy file FO/CC/03 (which belongs to FO) is automatically available for FP.*

If source FP/PS/00 is checked out then its copy statements are qualified as follows:

```
COPY ``(FP)FP/CC/01 ON DEVPK`` (used from DEVELOP because the system of the copy  
file matches the system of the source (both FP))
```

and that copy file contains the following nested copy statements:

```
COPY ``(FP)FP/CC/02 ON DEVPK``  
COPY ``(FP)FO/CC/03 ON DEVPK``
```

### Disadvantage

The FP-source uses an FO copy file from an invalid location. If the copy statement was used in the source itself, this copy file would have been qualified as COPY ``\*FO/CC/03 ON INTPK.`` This must be solved by copying that copy file manually from the resource location of FO in INTEGRATION to the resource location of FP in DEVELOP.

If source FO/PS/00 is checked out then its copy statements are qualified as follows:

```
COPY ``*FP/CC/01 ON DEVPK`` (used from INTEGRATION because the of the copy file  
differs from the system of the source (FP versus FO))
```

and that copy file contains the following nested copy statements:

```
COPY ``*FP/CC/02 ON INTPK``  
COPY ``*FO/CC/03 ON INTPK``
```

### Disadvantage

The FO source uses an FO copy file from INTEGRATION instead of DEVELOP. If the copy statement was used in the source itself, this copy file would have been qualified as COPY ``(FO) FO/CC/03 ON DEVPK.``

### Resource Qualification Using Method 2

In this case, the nested resources in a copy file in the resource-location are qualified according to the same resource-rules as when that copy file is checked out. Resources of a different system are qualified with the resource-location of that other system on the applicable resource environment.

Copy file FP/CC/01 in the resource-location of DEVELOP has the following content:

```
COPY "(FP)FP/CC/02 ON DEVPK"
COPY "*FO/CC/03 ON INTPK"
```

- The FP copy statement is qualified with the resource location of FP in DEVELOP, because it belongs to the same system as copy file FP/CC/01.
- The FO copy statement is qualified with the resource location of FO in INTEGRATION, because it belongs to another system as copy file FP/CC/01.

**Note:** Each copy statement is qualified with a valid resource location, so there are no manual actions required.

Copy file FP/CC/01 in the resource-location of INTEGRATION has the following content:

```
COPY "*FP/CC/02 ON INTPK"
COPY "*FO/CC/03 ON INTPK"
```

Each copy statement is qualified with the resource-location of its own system in INTEGRATION.

**Note:** Each copy statement is qualified with a valid resource location, so there are no manual actions required.

If source FP/PS/00 is checked out then its copy statements are qualified as follows:

```
COPY "(FP)FP/CC/01 ON DEVPK" (used from DEVELOP because the system of the copy
file matches the system of the source (both FP))
```

and that copy file contains the following nested copy statements:

```
COPY "(FP)FP/CC/02 ON DEVPK"
COPY "*FO/CC/03 ON INTPK"
```

If source FO/PS/00 is checked out then its copy statements are qualified as follows:

```
COPY "*FP/CC/01 ON INTPK" (used from INTEGRATION because the copy file differs
from the system of the source (FP versus FO))
```

and that copy file contains the following nested copy statements:

```
COPY "*FP/CC/02 ON INTPK"
COPY "*FO/CC/03 ON INTPK"
```

### Disadvantage

The FO source uses an FO copy file from INTEGRATION instead of DEVELOP. If the copy statement was used in the source itself, this copy file would have been qualified as COPY "(FO) FO/CC/03 ON DEVPK."

### Resource Qualification Using Method 3

In this case, a resource in a copy file in the resource-location is qualified with the default resource location of its own system.

Copy file FP/CC/01 in the resource-location of DEVELOP has the following content:

```
COPY "(FP)FP/CC/02 ON DEVPK"
COPY "(FO)FO/CC/03 ON DEVPK"
```

Each copy statement is qualified with the resource-location of its own system in DEVELOP.

**Note:** *Each copy statement is qualified with a valid resource location, so there are no manual actions required.*

Copy file FP/CC/01 in the resource-location of INTEGRATION has the following content:

```
COPY "*FP/CC/02 ON INTPK"
COPY "*FO/CC/03 ON INTPK"
```

Each copy statement is qualified with the resource-location of its own system in INTEGRATION.

**Note:** *Each copy statement is qualified with a valid resource location, so there are no manual actions required.*

If source FP/PS/00 is checked out then its copy statements are qualified as follows:

```
COPY "(FP)FP/CC/01 ON DEVPK" (used from DEVELOP because the system of the
copy file matches the system of the source (both FP))
```

and that copy file contains the following nested copy statements:

```
COPY "(FP)FP/CC/02 ON DEVPK"
COPY "(FO)FO/CC/03 ON DEVPK"
```

### Disadvantage

The FP-sources uses an FO copy file from DEVELOP instead of INTEGRATION. If the copy statement was used in the source itself, this copy file would have been qualified as COPY "\*FO/CC/03 ON INTPK."

If source FO/PS/00 is checked out then its copy statements are qualified as follows:

```
COPY "*FP/CC/01 ON INTPK" (used from INTEGRATION because the copy file differs
from the system of the source (FP versus FO)).
```

and that copy file contains the following nested copy statements:

```
COPY "*FP/CC/02 ON INTPK"
COPY "*FO/CC/03 ON INTPK"
```



### **Disadvantage**

The FO source uses an FO copy file from INTEGRATION instead of DEVELOP. If the copy statement was used in the source itself, this copy file would have been qualified as COPY "(FO) FO/CC/03 ON DEVPK."

### **Work Around for the Invalid Nested Resource Qualification in Methods 2 and 3**

This must be solved as follows:

- First do a check out of copy file FP/CC/01 and choose on the check out screen the correct resource-location for the nested copy files. SURE will lead you in this.
- Then do a check out of the source itself, and that will qualify copy file FP/CC/01 with its check-out location.
- When the changes in the source are done
  - Do a check in the source.
  - Do an undo check out for copy file FP/CC/01.



## Section 16

# Life Cycle Support (Internals)

### 16.1. (Internals) Check-In

The following relations are modified when a file is checked in.

#### Check-In

Action	Entity owner:class (asset)
Del	FILE-CONTROL <filename>:SOURCE-USERCODE(<work-usercode>)
del	
del	FILE-CONTROL <filename>:SOURCE-PACK(<work-pack>)
del (1)	FILE-CONTROL <filename>:SOURCE-HOST(<work-host>)
del	FILE-CONTROL <filename>:SOURCE-COPY(<PC file>)
del	FILE-CONTROL <filename>:USER-ID(<user-id>)
upd (2)	FILE-CONTROL <filename>:REQUESTED(<user-id>)
upd	FILE <filename>:MAINTAINED(<user-id>)
upd	FILE <filename>:FILEKIND(<filekind of diskfile>)
upd	FILE <filename>:STOR(NULL)
add	FILE-CONTROL <filename>:CHANGED(NULL)
add	FILE-CONTROL <filename>:IMPACT(<task>)
add	FILE-CONTROL <filename>:COMPILE-STATUS(TO-COMPILE)
add (3)	FILE-CONTROL <filename>:EXAMINE-STATUS(TO-EXAMINE)
add (4)	FILE-CONTROL <filename>:PREVIOUS(NULL)
add	FILE-CONTROL <filename>:MATCH(NULL)
	FILE-CONTROL <filename>:CHECK(STOR)

- (1) This relation is deleted if local edit is not used.
- (2) The user-id is copied from the USER-ID relation.
- (3) This relation is the previous STOR relation. It points to the previous version of the file.
- (4) This relation is only added if it is an existing file, not for a first check in of a new file.

#### 16.1.1. Compile as a Result of File Change

Every file that is changed in an environment (using check-in or transfer) is added to the SURE compile queue of that environment. This is done with relation `COMPILE-STATUS(TO-COMPILE)` or using relation `IMPACT(<task>)`. The actual compilation is done by the RESPECT/SURE/COMPILE program that runs in the SURE batch job.

### **Notes:**

- *Copy or include files are also placed into the compile queue. In this case, it results in the compilation of the files using this copy or include file.*
- *The attributes object usercode and object pack will determine the location of the object file.*

### **16.1.2. Examine as a Result of a File Change**

Every file that is changed in an environment will be placed into the examine queue of that environment. This is done through the `EXAMINE-STATUS( TO-EXAMINE )` relation. The actual examination of the sources is done by the `RESPECT/SURE/EXAMINE` program that runs in the SURE batch environment. This program is a syntax parser that extracts structural information from the source and stores it as relations in the repository (for example, copy files and datasets).

### **Notes:**

- *Examining of sources is an essential process within SURE, because this process creates the interrelationships between sources that are used during the compile process.*
- *The FILE-TYPE declaration may indicate to omit examining for a specific FILE-TYPE.*

### **16.1.3. Match as a Result of a File Change**

Every file that is changed in an environment is placed into the match queue of that environment. This is done through the `MATCH( <NULL> )` relation. The match process creates a delta file with the differences between the new FileVersion and the previous FileVersion. The actual creation of the delta files is done by the `RESPECT/SURE/MATCH` program that runs in the SURE batch environment.

Delta files are used by SURE to construct previous file versions. The programmer may benefit from these files by examining changes using the delta files.

## **16.2. (Internals) Check-Out**

The following relations are modified when a file is checked out:

- The `ASSIGNIT` command is a combination of `REQUEST` and `ASSIGN` commands.
- The `CHECKOUT` command is a combination of `REQUEST`, `ASSIGN`, and `GET` commands.

### **Request**

Action	Entity owner:class(asset)
Add	FILE-CONTROL <filename>:REQUESTED(<userid>)
Add	FILE-CONTROL <filename>:PROBLEM(<task>)

**Assign**

Action	Entity owner:class(asset)
Add	FILE-CONTROL <filename>:ASSIGNED(<userid>)

**Get**

Action	Entity owner:class(asset)
Add	FILE-CONTROL <filename>:SOURCE-USERCODE(<work-usercode>)
Add	FILE-CONTROL <filename>:SOURCE-PACK(<work-pack>)
Add (1)	FILE-CONTROL <filename>:SOURCE-HOST(<PC name>)
Add (1)	FILE-CONTROL <filename>:SOURCE-COPY(<PC filename>)
Add	FILE-CONTROL <filename>:USER-ID(<userid>)

(1) Only if the file is checked out with option "local edit."

## 16.3.(Internals) Transfer

**Relation Modification**

The transfer of a file causes all relations with RIS-entity = FILE to be copied to the destination environments. The relations with RIS-entity = FILE-CONTROL are not copied, but updated according to the following scheme.

**Transfer**

Action	Environment*Entity Owner:class(asset)
add (1)	<destination env>*FILE-CONTROL <filename>:PREVIOUS(NULL)
add (2)	<destination env>*FILE-CONTROL <filename>:OBJECT-USERCODE(<object usercode>)
add (2)	<destination env>*FILE-CONTROL <filename>:OBJECT-PACK(<object pack>)
add (2)	<destination env>*FILE-CONTROL <filename>:OBJECT-HOST(<object host>)
add	<destination env>*FILE-CONTROL <filename>:MATCH(NULL)
add	<destination env>*FILE-CONTROL <filename>:EXAMINE-STATUS(TO-EXAMINE)
add	<destination env>*FILE-CONTROL <filename>:COMPILE-STATUS(TO-COMPILE)
add	<destination env>*FILE-CONTROL <filename>:CHANGED(NULL)
del	<source env>*FILE-CONTROL <filename>:ASSIGNED(<user-id>)
upd	<source env>*FILE-CONTROL <filename>:STATUS(<destination env>)
upd	<destination env>*FILE-CONTROL <filename>:STATUS(<destination env>)
add	<destination env>*FILE-CONTROL <filename>:PROBLEM(<task>)
del	<source env>*FILE-CONTROL <filename>:PROBLEM(<task>)
add	<destination env>*FILE-CONTROL <filename>:IMPACT(<task>)
add (3)	<destination env>*FILE-CONTROL <filename>:HISTORY(<task>)

1. This relation is copied from the previous STOR relation in the destination environment.
2. See the discussion about "object location" below.
3. This relation is only added if the delivery option is set for the system of the task.

### **16.3.1. Object Location as a Result of Transfer**

The object attributes `OBJECT-USER`, `OBJECT-PACK` and `OBJECT-HOST` are inherited from the system defined options in combination with the `FILE-TYPE` definition.

The system definition contains entries for object locations. The `FILE-TYPE` definition contains four indications referring to these attributes:

- Option `Inherit object-location-1` that specifies to use the system defined `object-location-1` definitions as object attributes (`object-user/pack`).
- Option `Inherit object-location-2` that specifies to use the system defined `object-location-2` definitions as object attributes (`object-user/pack`).
- Option `Inherit object-location-3` that specifies to use the system defined `object-location-3` definitions as object attributes (`object-user/pack`).
- Option `Use never object-relations` that specifies to leave the object attributes empty.

These four options define which fields from the system definitions are to be used as object attributes for the file.

The three system-object-locations are treated by SURE in the same way. These are just three different predefined object-locations.

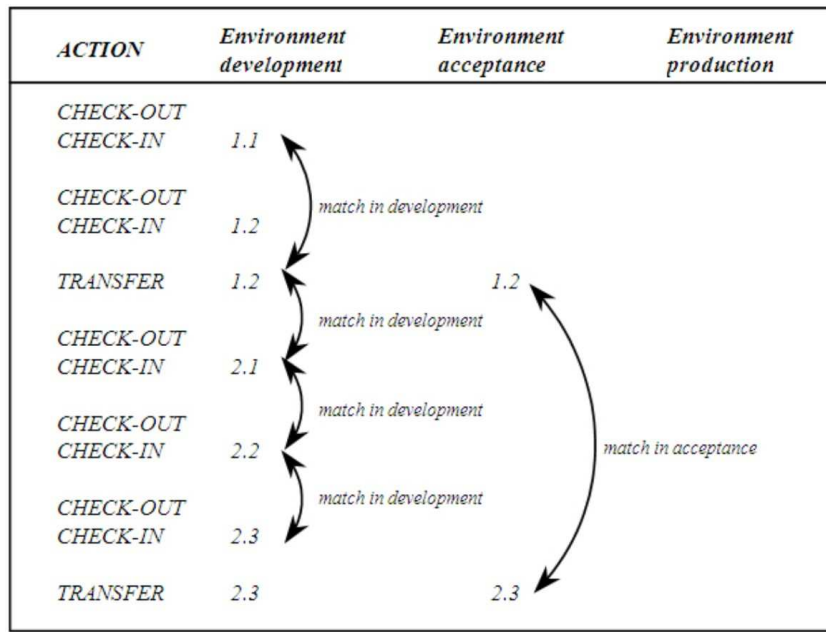
The object-location of a file is re-evaluated when

- The system or the file type of the file is changed.
- The object-locations of the system attribute of the file are changed.
- The inherit-object-location of the file type of the file is changed.

### **16.3.2. Delta Files as a Result of a Transfer**

Each file that is transferred to an environment will be placed into the SURE match queue of that destination environment.

The `RESPECT/SURE/MATCH` program creates a delta file with differences between the new `FileVersion` and the previous `FileVersion`. It is obvious that the delta files in one environment may be different and have another range than the delta files in another environment.

**Example**

For PC files, the delta files are created at the PC when the source is checked in. At transfer time, the appropriated delta files from the environment where the file is transferred from are copied to the destination environment. Therefore, the sequence of delta files in the various environments is usually equal. Special commands like quick fix and recover may cause differences in this sequence.

Refer to Section 17, "Delta Files," for more information on Delta files.

**16.3.3. Examine as a Result of Transfer**

Each file that is transferred to an environment will be placed into the SURE examine queue of that destination environment.

Examining a file is done after applying all compiler control cards that are available for the source using the merged card file as a primary input file. Because the compiler control card options in this file may differ for every environment, the actual compiler input, and therefore the examine input may differ for every environment.

Additionally, the compilation translate tables defined for every environment may alter the source used as input for the examine process.

Due to these reasons, the examine process may result in different output for every environment.

### Example

Consider source PROG/AA that belongs to system SYS1:

```
Source PROG/AA      100 BEGIN
                    200$ SET OMIT = NOT TEST
                    300 $ INCLUDE "TEST/LAYOUT"
                    400$ POP OMIT
                    500$ SET OMIT = NOT PROD
                    600 $ INCLUDE "FUNNY/PRODUCTION/FILE"
                    700$ POP OMIT
                    750
                    800 DATABASE DBTEST;
                    900 END.
```

### Environment DEVELOP

Compiler card input

```
$ SET MERGE TEST
```

The examined relations are

#### Environment\*entity|owner:class(asset)

```
DEVELOPMENT*FILE-CONTROL | PROG/AA: COPY-FILE (TEST/LAYOUT)
DEVELOPMENT*FILE-CONTROL | PROG/AA: DATABASE (DBTEST)
```

### Environment PRODUCTION

Compiler card input:

```
$ SET MERGE PROD
```

Translate table for environment production

```
DBTEST@
DBPROD@
```

The examined relations are:

#### Environment\*entity|owner:class(asset)

```
PRODUCTION*FILE-CONTROL | PROG/AA: COPY-FILE (FUNNY/PRODUCTION/FILE)
PRODUCTION*FILE-CONTROL | PROG/AA: DATABASE (DBPROD)
```



## 16.4.(Internals) Enter a New File

### Relation Modification

The NEW or FILE command results in the following relation modifications:

#### New

Action	Entity owner:class(asset)
add	FILE <filename>:AUTHOR(<entry on screen>)
add	FILE <filename>:FILE-TYPE(<entry on screen>)
add	FILE <filename>:FUNCTION(<entry on screen>)
add	FILE <filename>:MAINTENANCE(<user-id>)
add	FILE-CONTROL <filename>:ASSIGNED(<user-id>)
add	FILE-CONTROL <filename>:CHANGED(NULL)
add (1)	FILE-CONTROL <filename>:OBJECT-PACK(<object usercode>)
add (1)	FILE-CONTROL <filename>:OBJECT-PACK(<object pack>)
add (1)	FILE-CONTROL <filename>:OBJECT-HOST(<object host>)
add	FILE-CONTROL <filename>:REQUESTED(<user-id>)
add	FILE-CONTROL <filename>:PROBLEM(<current task>)
add (2)	FILE-CONTROL <filename>:SOURCE-USERCODE(<work usercode>)
add (2)	FILE-CONTROL <filename>:SOURCE-PACK(<work pack>)
add (2)	FILE-CONTROL <filename>:SOURCE-HOST(<work host>)
add (2)	FILE-CONTROL <filename>:USER-ID(<user-id>)
add	FILE-CONTROL <filename>:STATUS(<environment>)

1. These relations define together the object-location. They are inherited from the system's object definition in combination with the FILE-TYPE definition (see below).
2. These relations define together the source location, and are compared with the system's work-environment definition.

### Object Attributes

Object attributes such as OBJECT-USER, OBJECT-PACK, and OBJECT-HOST are derived from the SYSTEM definitions in combination with the FILE-TYPE definition. The FILE-TYPE definition defines the object-location that must be inherited from the SYSTEM definition: object-location-1, -2, or -3. Depending on these definitions, the object attributes are set for the file.

**Note:** Transferring the file will derive the object definitions from the SYSTEM definitions defined for the transferred environments. For this reason, the object-location will be different in every environment.

## 16.5.(Internals) Logical Delete a File

### Relation Modification

The “Logical Delete” command hides all relations in groups FILE and FILE-CONTROL. The other relations that are modified are:

### Remove

Action	Entity owner:class(asset)
--------	---------------------------

Add	FILE-CONTROL <filename>:PROBLEM(<current task>)
Upd	FILE-CONTROL <filename>:STATUS(REMOVE)

## 16.6.(Internals) Undo Logical Delete a File

### Relation Modification

The “Undo Logical Delete” command re-establishes all relations in groups FILE and FILE-CONTROL. The other relations that are modified are:

### Undo remove ( = Activate)

Action	Entity owner:class(asset)
--------	---------------------------

Add	FILE-CONTROL <filename>:PROBLEM(<current task>)
Upd	FILE-CONTROL <filename>:STATUS(<environment>)

## 16.7.(Internals) Purge a File in this Environment

### Relation Modification

The “Purge a file in this environment” command deletes all relations in groups FILE and FILE-CONTROL. The other relations that are modified are:

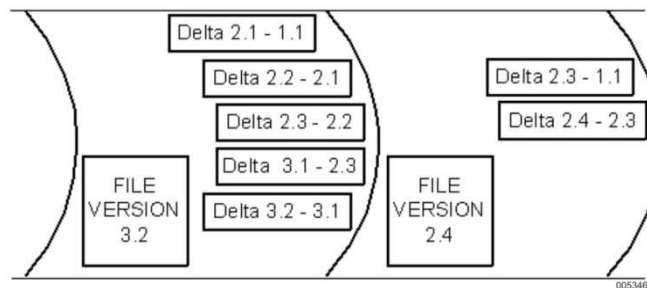
### Purge

Action	Entity owner:class(asset)
--------	---------------------------

Add	FILE-CONTROL <filename>:CHECK(STOR)
-----	-------------------------------------

## Section 17

# Delta Files



Delta files are the files that contain differences between consecutive versions of a file in the repository.

Within SURE, these files are used to provide enhanced support for programmers and they are used to rebuild any requested previous version of a file (= FileVersion). SURE does not use delta files for its main processing mechanism.

Delta files are available for binary files and for source files.

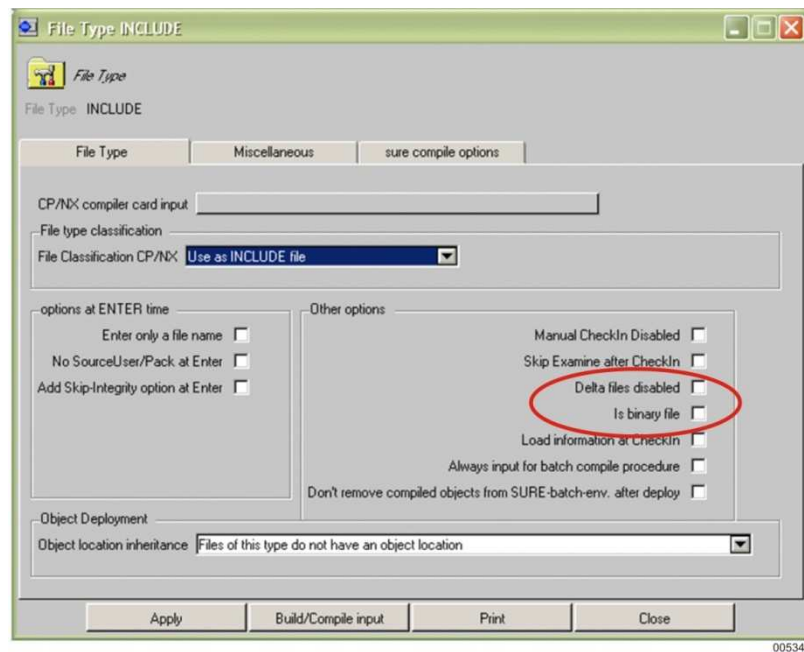
### 17.1. Delta Files for Source Files

Examples of source files are:

- MCP source files: All files with sequence numbers, such as COBOL85SYMBOL, ALGOLSYMBOL, JOBSYMBOL, SEQDATA, and TEXTDATA.
- PC or UNIX source files, such as C++, Java, XML, MicroFocus-COBOL, Delphi, and PowerBuilder

SURE treats a file as a "Source file" if its FILE-TYPE has option "Is binary file" reset.

A source file has delta file if its FILE-TYPE has option "Delta files disabled" reset.



The support provided by these delta files are:

- Rebuild a previous version of the source (for example, to compare it with the current version of the source)
- MCP sources only:
  - Create Unisys patch files that may be used for software updates.
  - Search (within a sequence range) for modifications on the source.

Delta files are the differences between two consecutive FileVersions that are checked in, in a particular environment. Therefore, these delta files can be used to recreate any previous version in an environment, but they cannot be used to recreate a FileVersion in another environment.

**Example**

Action	Environment Development	Delta	Environment Production	Delta
Check-In	1.1			
Transfer	1.1		1.1	
Check-In	2.1	2.1 - 1.1		
Check-In	2.2	2.2 - 2.1		
Check-In	2.3	2.3 - 2.2		
Transfer	2.3		2.3	2.3 - 1.1
Check-In	3.1	3.1 - 2.3		
Check-In Quick Fix			2.4	2.4 - 2.3
Check-In	3.2	3.2 - 3.1		

After the listed actions, the repository contains two complete files (3.2 in development and 2.4 in production) and seven delta files (five in development and two in production). As the example shows, the delta files can be used to rebuild any version in a particular environment.

Notice that FileVersion 2.4 cannot be reconstructed using the delta files from environment "development."

The content of an MCP delta file is similar to the output of the CANDE match command.

**Example**

FileVersion 2.2:

```
00000000% VERSION    2.2;    CHECK IND 95-12-27 09:44:50
00000100 BEGIN
00000200     DISPLAY("5");
00000300 END.
```

FileVersion 2.3:

```
00000000% VERSION    2.3;    CHECK IND 95-12-27 09:54:50
00000100 BEGIN
00000200     DISPLAY("6");
00000220     DISPLAY("new line");
00000300 END.
```

Delta file between 2.3 and 2.2:

```
+00000000% VERSION    2.3;    CHECK IND 95-12-27 09:54:50
-^^^^^^^^% VERSION    2.2;    CHECK IND 95-12-27 09:44:50
+00000200    DISPLAY("6");
-^^^^^^^^    DISPLAY("5");
+00000220    DISPLAY("new line");
```

SURE provides commands to create Unisys ClearPath server patch files from these delta files. These patch files can be used as input for the SYSTEM/PATCH or SYSTEM/EDITOR (U ED).

Because ClearPath server delta files rely on sequence numbers, a resequence of a file would create delta files containing the complete file. For this reason, a SURE resequence command is available.

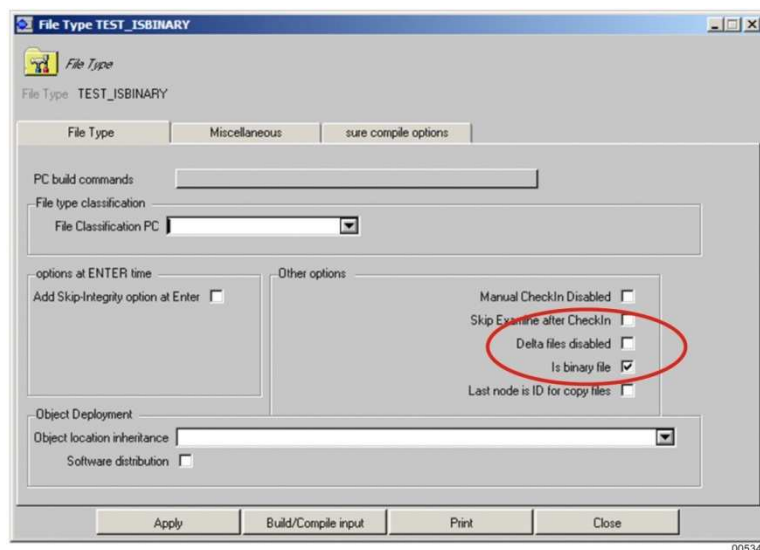
## 17.2. Delta Files for Binary Files

Examples of binary files are:

- MCP binary files: All files without sequence numbers such as data files and object files.
- PC or UNIX binary files: Word, Excel, or PowerPoint documents, data files, and so on.

SURE treats a file as a "Binary file" if its FILE-TYPE has option "Is binary file" set.

A binary file has delta file if its FILE-TYPE has option "Delta files disabled" reset.

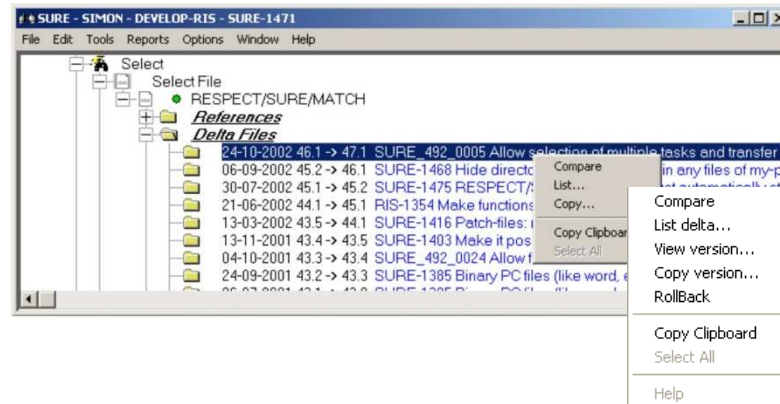


If a binary file contains delta files, then a complete file version is saved as a delta file. You cannot compare the delta files; however, you can copy, view, and rollback to any previous version.

## 17.3. (Delta Files) Windows

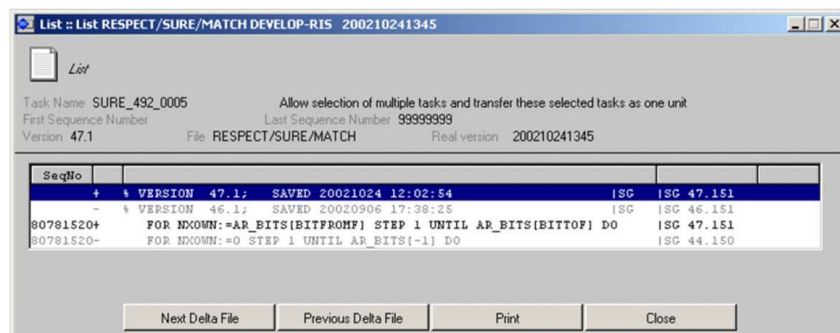
The DELTA files can be accessed using different ways in the Windows interface. Opening the file folder shows a sub folder called Delta files. Opening this sub folder will show all the delta files for the file.

By default, a delta file is created during the evening batch. However, if this list with delta files is opened, the pending delta file is created immediately.



Selecting a delta file in the SURE explorer allows simple functions like list and compare:

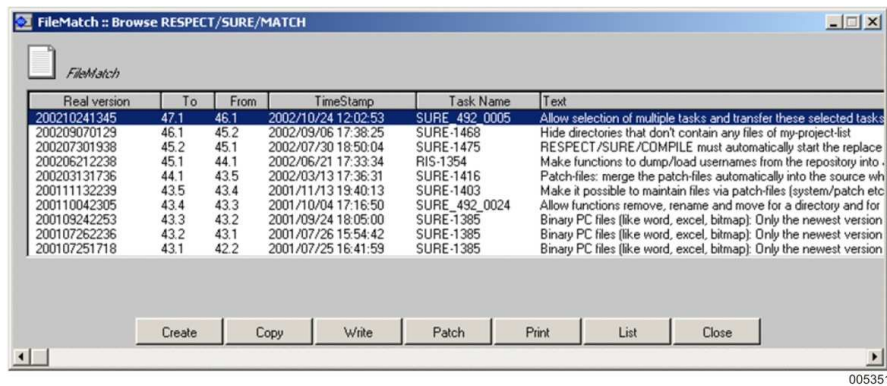
- Right-click the delta file; a popup menu with different commands are displayed.
- Clicking the object bitmap lists the delta file and allows browsing to the next or previous delta file.



005350

Browse through the list with delta files with buttons "Next delta file" and "previous delta file."

The popup folder under Delta files gives the detail function, which gives a tabular format with the delta files and a command menu bar. This same format can be accessed through the file properties Info menu through the DELTA function.



Selecting one of these delta files in the detail function allows performing one of the next listed commands.

**Note:** The first delta file is selected if the form is showed. Select another delta file by clicking it with the mouse. The color of the selected version differs from the other delta files.

### 17.3.1. (Delta files) Create

If a source is checked in, a match-relation is added [`<filename>:MATCH(<NULL>)`], which instructs the RESPECT/SURE/MATCH batch program to create delta files for that source. By default, the delta files are not created during the on-line commands, but by RESPECT/SURE/MATCH during the SURE batch, because the creation of a delta file may take some time and may cause some performance degradation of the on-line command. Since the delta files are created during the SURE batch, they will be available on the day after saving a file.

If a delta file is required in some on-line action, but that delta file is not yet created by the batch program, then it will be created immediately when the on-line command is executed.

### 17.3.2. (Delta files) Copy

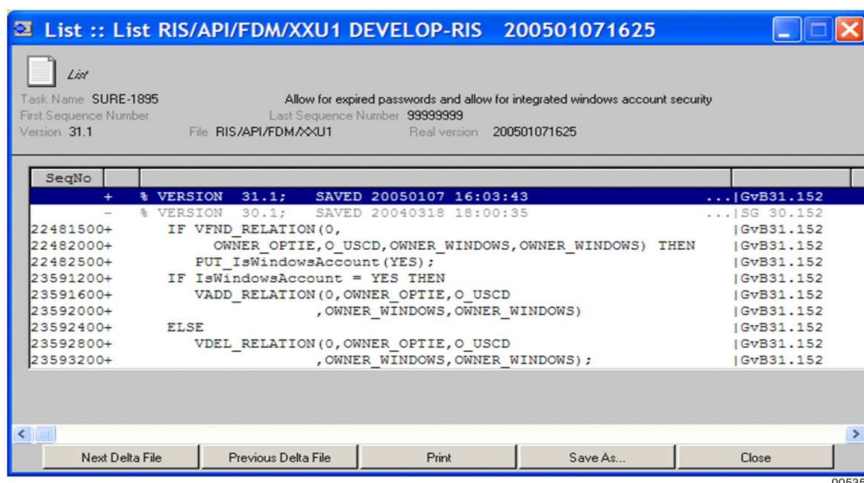
The COPY function copies the current selected delta file to the ClearPath server disk and optionally to a PC destination.

If the FileName field is not filled then a name is suggested by SURE.

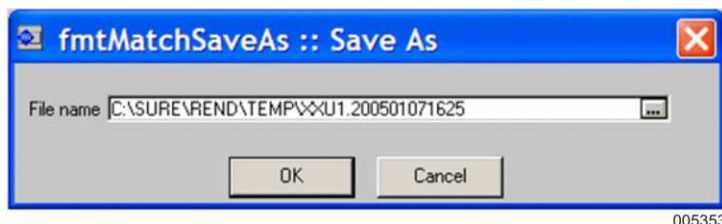
### 17.3.3. (Delta files) Save as

The "list delta file" screen contains now a new button "save as." With this button it is possible to copy the content of the delta file to a text file in the temporary directory. This text file can then be used to copy paste pieces of the source.





Click <Save as>.



You can overwrite the proposed name with your own choice.

### 17.3.4. (Delta files) Write

The WRITE function writes the current selected delta file to the user-defined printer.

**Note:** The user-defined printer is controlled by the ClearPath server. Copying the delta file to a PC destination and using a local word processor to print the contents causes printing without ClearPath server printer definitions.

### 17.3.5. (Delta files) Patch

The PATCH function creates a Unisys ClearPath server patch file from the delta file, to be used by the SYSTEM/EDITOR (U ED) or by SYSTEM/PATCH.

If the FileName field is not filled then a name is suggested by SURE.

The patch menu contains two choices: FORWARD and BACKWARD.

#### FORWARD

Create a patch file to roll the file forward from the FileVersion preceding the selected delta file up to the selected delta file.

### Example

FileVersion 2.2

```
00000000% VERSION    2.2;    CHECK IND 95-12-27 09:44:50
00000100 BEGIN
00000200     DISPLAY("5");
00000300 END.
```

FileVersion 2.3

```
00000000% VERSION    2.3;    CHECK IND 95-12-27 09:54:50
00000100 BEGIN
00000200     DISPLAY("6");
00000220     DISPLAY("new line");
00000300 END.
```

The FORWARD option creates the following patch file, which can be used to create version 2.3 if it is applied to version 2.2.

```
00000000% VERSION    2.3;    CHECK IND 95-12-27 09:54:50
00000200     DISPLAY("6");
00000220     DISPLAY("new line");
```

### BACKWARD

Create a patch file to roll the file backward from the selected delta file to the preceding FileVersion.

### Example

FileVersion 2.2

```
00000000% VERSION    2.2;    CHECK IND 95-12-27 09:44:50
00000100 BEGIN
00000200     DISPLAY("5");
00000300 END.
```

FileVersion 2.3

```
00000000% VERSION    2.3;    CHECK IND 95-12-27 09:54:50
00000100 BEGIN
00000200     DISPLAY("6");
00000220     DISPLAY("new line");
00000300 END.
```

The BACKWARD option creates the following patch file, which can be used to create version 2.2 if it is applied to version 2.3.

```
00000000% VERSION    2.2;    CHECK IND 95-12-27 09:44:50
00000200     DISPLAY("5");
00000220$
```

### 17.3.6. (Delta files) List

This function lists the contents of the selected delta file.

### 17.3.7. (Delta files) Roll back

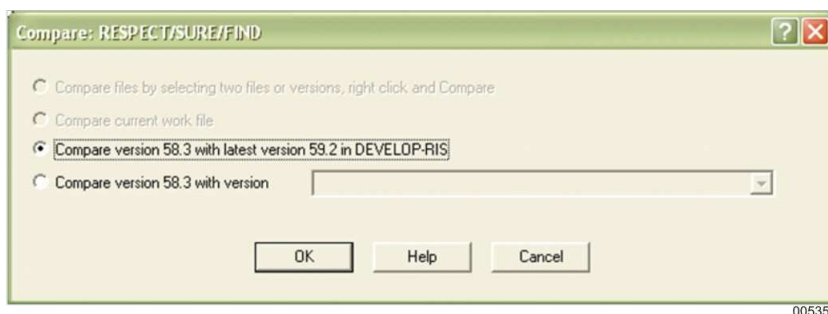
This function recreates a previous version of the source in SURE. For MCP files it is possible to undo the rollback. For PC-files that is not possible because a roll-back of a PC-file deletes all the delta-file that are rolled-back.

### 17.3.8. (Delta files) Compare

This function compares a previous version of the source with the current version or with another previous version.

Compare a previous version with the current version as follows:

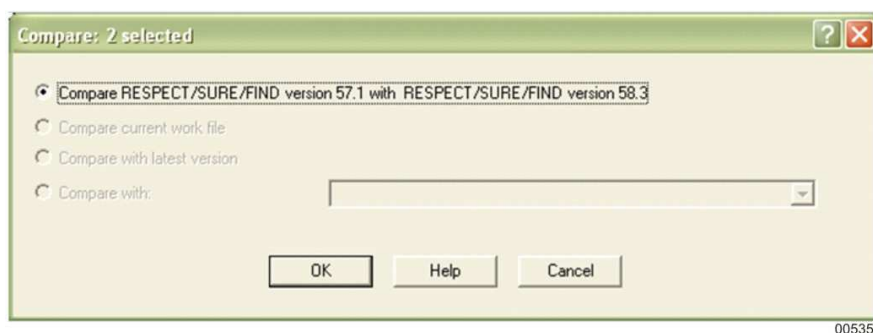
- Right-click the delta file that changed the previous version and choose compare



- Click OK to start the compare.

Compare two previous versions as follows:

- Select the two delta-files that changed the versions that you want to compare.



- Click OK to start the compare.

The previous versions are temporarily recreated and loaded in the compare tool that is defined in the local-options. The default compare tool is WinMerge.

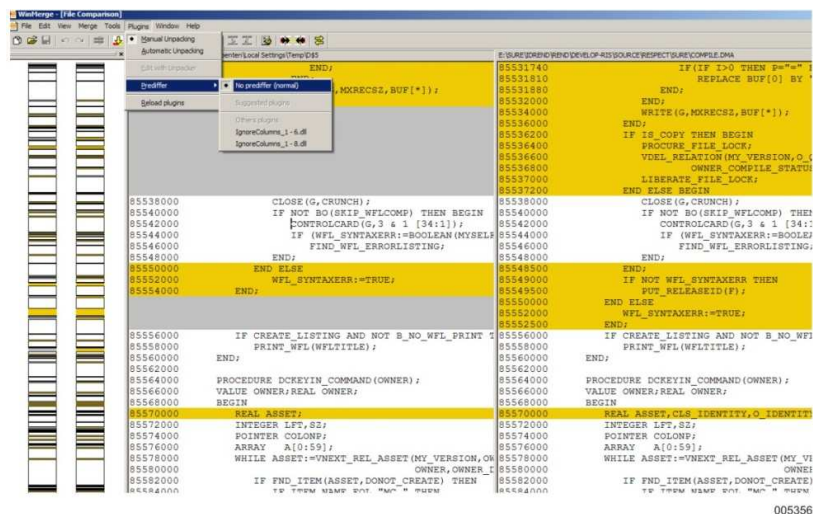
SURE co-installs the WinMerge 2.2 application, a tool to compare different versions of a source.

The advantage of WinMerge 2.2 is that it offers a plug-in to ignore differences in certain columns.

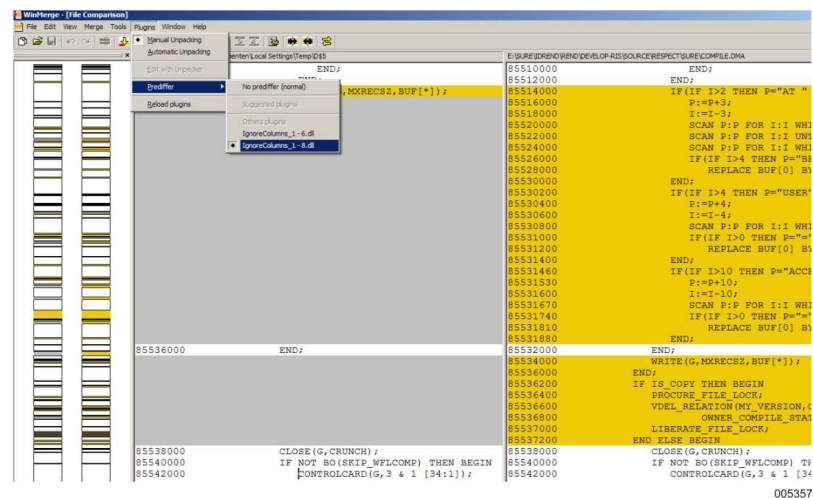
Mainframe files contain sequence numbers and a resequence of a file causes large delta files. WinMerge 2.2 with the installed plug-in ignores the sequence range differences and shows only the differences in the source code.

### Example

The following screen shot shows the WinMerge application with the PlugIns/Prediffer set to no pre differ, meaning the sequence numbers included.

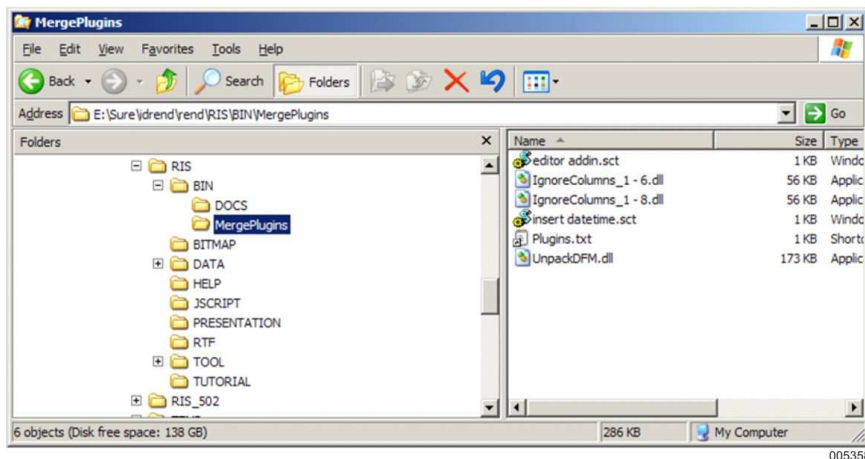


The following screen shot shows the WinMerge application with the PlugIns/Prediffer set to column 1-8 ignored.

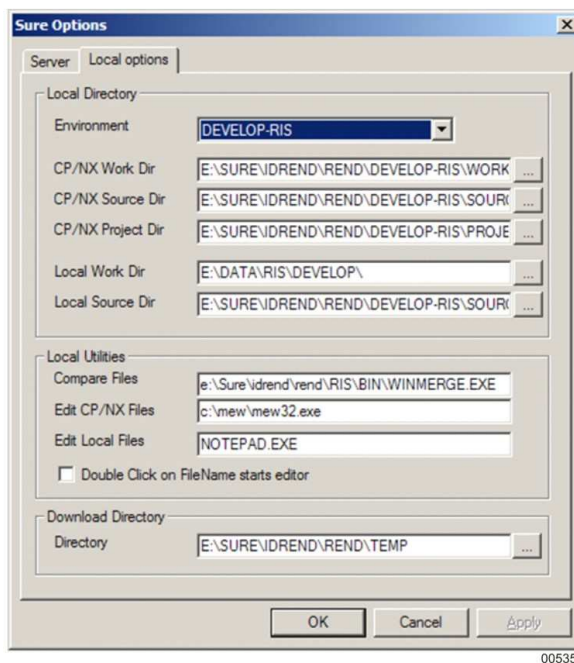


## Technical Details and Considerations

The SURE installation installs the WinMerge application by default in the BIN directory. This directory contains a sub directory MergePlugins that contain the installed plug-ins.



You need to set the WinMerge application as the application used to show the differences using the Option/SURE options and the local options tab sheet as the next example shows.



### 17.4. (Delta files) Resequencing

The ClearPath server delta files rely on sequence numbers. This implies that a resequence of a file would create a huge delta file containing the original file two times: each line with the old sequence number and the new sequence number. This makes the delta file useless and therefore it is recommended not to resequence a source using CANDE or another editor.

On the other hand, there is often the need to resequence a source, because the sequence numbers do not fit anymore.

For this reason, a SURE resequence command is available.

The resequence (file properties maintenance) is used to resequence a file and retain the MATCH LIST ability. The file must be assigned and in maintenance.

### 17.5. (Delta files) Relation Modification

The creation of delta files is controlled with the following relation:

This relation is added when a file is changed in an environment (Checked in or transferred). The FileVersions to be matched are stored in the relation. Therefore, every file change will add a MATCH relation:

```
FILE-CONTROL | <source> :MATCH ( <null> )
```

This relation is removed by the RESPECT/SURE/MATCH program when the match is done.

### 17.6. (Delta files) Storage of Files and Delta Files

The files in the repository are stored in a common structure called DSTOR. This structure contains all files for all environments and its access is based on:

#### **STOR-OWNER**

The owner of the file, which equals to FIL-OWNER from the DFIL structure.

#### **STOR-CLASS**

An indication of the type of file like TOTAL file or DELTA file.

#### **STOR-VERSION**

A presentation for the value of FileVersion (Version \* 1000000 + Cycle) or timestamp.

#### **STOR-INDEX**

An ascending index used to access the information blocks.

The contents of the files, including the delta files, are stored in an internal compressed format. For this reason, the files can only be retrieved using SURE software.

When a file is checked in, this total source will be added to the repository with a unique and new store-version. (For example, ten times a check out or check in of a file directly after each other will result in ten different versions of that source loaded in the repository). Each time when a file is changed in an environment (by a check in or by a transfer) a `MATCH` relation is added which is used as a trigger for the RESPECT/SURE/MATCH program. This `MATCH` relation contains the two FileVersions to be matched.

A version of a file remains in the repository as long as that version is still active in one or more environments or as long as that version is still needed for a match.

Action	STOR CLASS	STOR VERSION	RELATION	V[1]	V[2]	V[3]	V[4]
Check In 1	TOTAL	1.1					
Check In 2	TOTAL	1.2	MATCH	1	2	1	1
Check In 3	TOTAL	1.3	MATCH	1	3	1	2

As the previous table shows, the three check in actions result in three total files in dataset DSTOR and two `MATCH` relations indicating the FileVersions to be matched. This situation is used in describing the options for the RESPECT/SURE/MATCH program.

Action "Check-in 1" does not add a match relation, because this is the first version of the file.

Action "Check-in 2" adds a match relation. This match relation contains FileVersions 1.1 and 1.2 as the two versions that have to be matched.

Action "Check-in 3" adds a match relation. This match relation contains FileVersions 1.2 and 1.3 as the two versions that have to be matched.

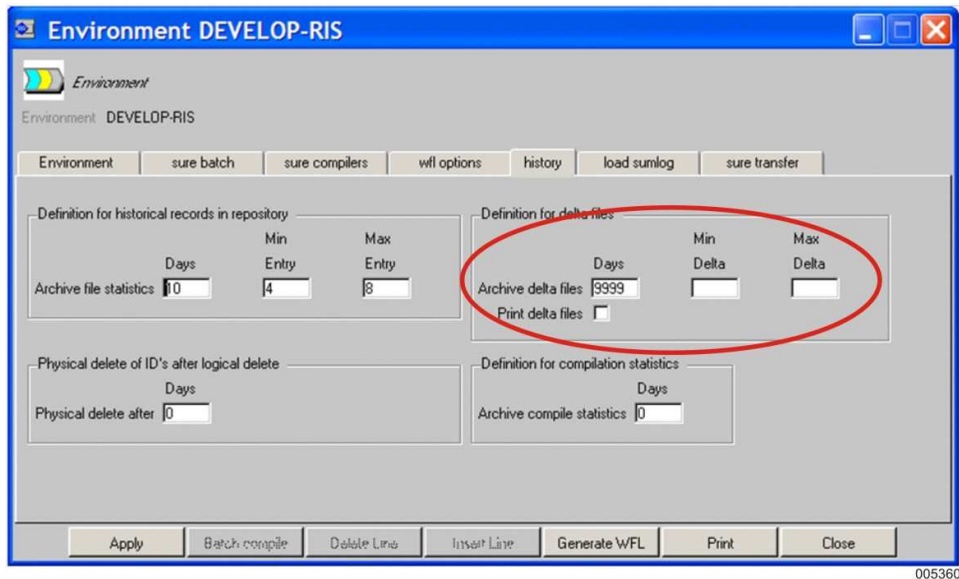
When the matches are completed, then the match relations are removed. If FileVersions 1.1 and 1.2 are not current in any environment, and if these FileVersions are not needed for a match in any environment, then they will be removed from the repository.

## 17.7. (Delta files) Cleaning Old Delta Files

The amount of days that a delta file must be kept in the repository must be defined for each environment in the repository. The match program will abort if this option is not set for one or more environments.

This option must be defined using the <environment> dialog, from Properties, select History tab.

It is possible to load delta files in each repository environment, and therefore the "delta file cleaning criteria" have to be defined for each environment too.



The following options must be defined for each environment.

Days	The number of days that a delta file of a source is kept in SURE (in this environment).
Min Delta	Binary files only: The minimum number of delta files that are kept in SURE for a source. Expired delta files are not removed if the number of delta files is less or equal than the minimum number.
Max Delta	Binary files only: The maximum number of delta files that are kept in SURE. If a source has more delta files, than the oldest ones above the maximum are removed, even if they are not yet expired. Remember that binary delta files are full copies of previous versions of a file. These delta files can consume very much space, so the maximum number of binary delta files should be kept low.
Print delta files	Source files only: The delta file will also be printed.

Expired delta files will be removed by the match program.

## 17.8. (Delta files) RESPECT/SURE/MATCH

The RESPECT/SURE/MATCH program runs for every environment in the SURE batch job and it creates and deletes delta files. The file change procedure triggers action for this program by the MATCH relations. This trigger is then used to process a file.

The RESPECT/SURE/MATCH program has a single parameter that determines the action. Use this parameter to control the creation of delta files, printing of delta files, or cleaning of delta files.



**Note:** Running this program for each environment is essential because it does the housekeeping for various datasets. If the RESPECT/SURE/MATCH program does not run, then the physical size of the repository will grow dramatically.

Therefore, this program must be present in every SURE batch environment. The RESPECT/SURE/MATCH program has the following functions:

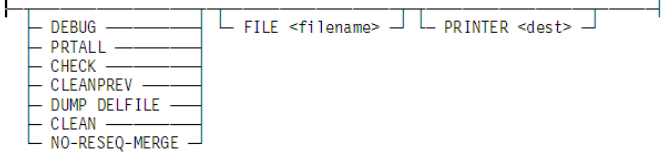
- Select all MATCH relations to match the FileVersions that are mentioned in these relations.
- Delete FileVersions that are not used anymore in any environment.
- Delete old delta files. The period that a delta file must be kept in the repository can be defined each environment. A delta file will be deleted if this period has expired.
- Delete FormatVersions that are not used anymore in any environment.
- Delete old tasks. The period that a solved task must be kept in the repository can be defined. A task will be deleted if this period has expired. This clean action will be done on every fifth day.
- Delete old log-info. The period that log-information must be kept in the repository can be defined. Log-info will be deleted when this period has expired. This clean action will be done on every fifth day.
- Delete old history-info. The period that history information must be kept in the repository can be defined. History info will be deleted when this period has expired. This clean action will be done on every fifth day.
- Delete old compilation statistics. The period that cumulated compilation statistics must be kept in the repository can be defined. These statistics will be deleted when this period has expired. This clean action will be done on every fifth day.
- Patch files are automatically merged into the main-source (and removed afterwards) if the main source itself and all its patch files are equal in all environments. The new (merged) main source is also resequenced automatically.

DSTOR structure after running the RESPECT/SURE/MATCH("")

STOR-CLASS	STOR VERSION	RELATION	V[1]	V[2]	V[3]	V[4]
MATCH	1.2					
MATCH	1.3					
TOTAL	1.3					

Run the RESPECT/SURE/MATCH program as follows:

```
RUN OBJECT/RESPECT/SURE/MATCH(" <run parameter> ");
```

Input	parameter	
		
		See "RESPECT/SURE/MATCH" in Section 47.
	Task string	: <environment>.
Output	overview	: Debug overview (optional). Listing of each created delta file (optional).
Example	RUN RESPECT/SURE/MATCH(" "); TASKSTRING = "DEVELOPMENT"	
Where	This program runs in the SURE evening batch.	
Security	MP +PU (the program accesses files in other directories)	
	<runparameter> = empty	The default, which creates delta files in the repository and deletes the old delta files.
	CLEANPREV	This option is used if the previous FileVersions are to be deleted. If an environment does not need objects, one may choose not to run the RESPECT/SURE/COMPILE program. Not running the compile program introduces a conflict situation within SURE: the previous FileVersion (= the last good compiled version of the file) is deleted by this compile process. If the compile program never runs, then these previous FileVersions are never deleted. Therefore, omitting the compile process must be combined with an additional run of the RESPECT/SURE/MATCH program providing the CLEANPREV option.
	CHECK	The option enforces the match program to check all versions of all files that are stored in the repository. If a FileVersion is not in use anymore (= not current in any environment, not previous in any environment, and not needed for a match in any environment), then it will be removed. It should not be necessary to use this option, because the regular match process does a check of all FileVersions for the files that are changed.
	DEBUG	This option is used for debugging purposes. An overview of available delta files is made.
	PRTALL	This option is used for debugging purposes. It makes an extended overview of available delta files.
	CLEAN	The RESPECT/SURE/MATCH program automatically does the housekeeping for the entire repository on each fifth day, according to the history-options that are defined for each environment and on the global level. Expired records are removed during this clean action. This option enforces an immediate clean action on the repository.

DUMP_DELFILE	This option enforces the match program to dump the FileVersions files that are deleted from the repository into a directory SUREDELETE/<filename>.
NO-RESEQ-MERGE	The RESPECT/SURE/MATCH program does an automatic merge of patch files into the main source if the source itself and all patch files of that source are equal in all environments. The merged file is also automatically resequenced. With this option the resequence is skipped.
FILE <filename>	This parameter can be used to match a single file instead of all files.
PRINTER <dest>	This parameter can be used to route the output to a print destination. If this parameter is not used, then the output will be routed to the default print destination of the usercode that started the match program.

**Note:** The “print” option, can be defined for each environment (Dialog <environment>/properties/tab sheet history). If this print option is set, then the delta files will also be printed when they are created.



## Section 18

# Repository Maintenance

Repository maintenance is performed by the MODIFY command, where manually defined entries, like AUTHOR and FUNCTION, can be updated.

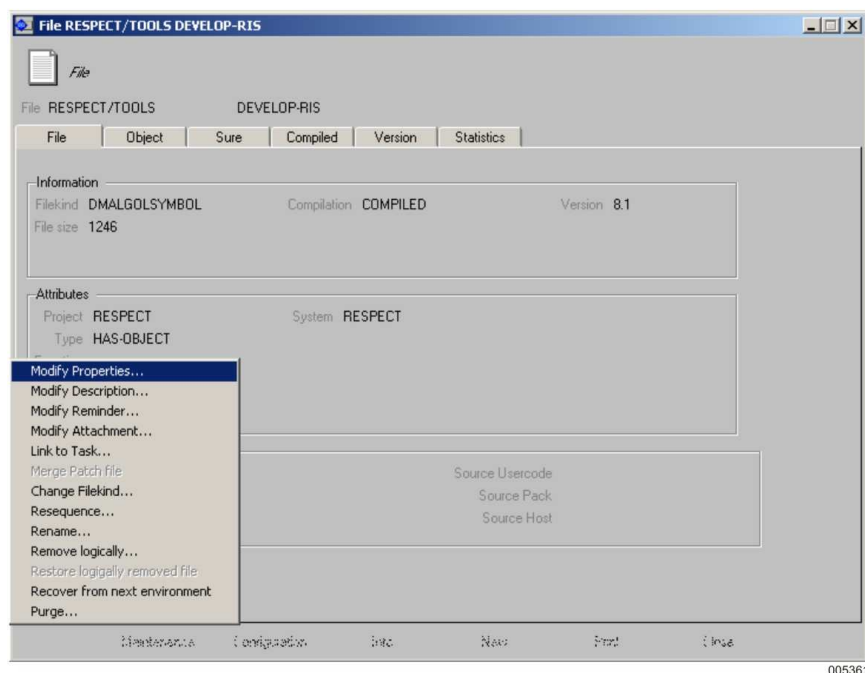
In practical situations, these updates are rarely used. Most of the attributes are derived from the system definition and are not entered manually.

A more powerful mechanism to update multiple files with a single command is the RELATE function. This function allows adding, deleting, or changing attributes or relations for multiple files in a single command.

A file name can be changed with function RENAME. This function is listed here although it could also be listed in the life cycle support functions.

### 18.1. Modify

Modify the file properties using the function Modify Properties under <file-properties>, under Maintenance menu.



The Modify command allows you to change the user defined SURE attributes for a file. This implies static attributes like FUNCTION or AUTHOR and dynamic attributes such as object attributes.

### Entityowner:class(asset)

```
<global>*FILE|<filename>:AUTHOR(<author name>)
<global>*PROJECT|<systemname>:SYSTEM(<filename>)
<global>*PROJECT|<projectname>:PROJECT(<filename>)
<global>*FILE|<filename>:FUNCTION(<text>)
FILE|<filename>:FILE-TYPE(<file type>)
FILE|<filename>:FREQUENCY(<text>)
FILE|<filename>:PRIVILEGE(<usercode name>)
FILE-CONTROL|<filename>:OBJECT(<filename>)
FILE-CONTROL|<filename>:OBJECT-USERCODE(<usercode name>)
FILE-CONTROL|<filename>:OBJECT-PACK(<family name>)
FILE-CONTROL|<filename>:OBJECT-HOST(<host name>)
FILE-CONTROL|<filename>:COPYFILE-USERCODE(<usercode>)
FILE-CONTROL|<filename>:COPYFILE-PACK(<family name>)
FILE-CONTROL|<filename>:LOGUSER(<usercode>)
FILE-CONTROL|<filename>:LOGPACK(<family name>)
FILE|<filename>:<filekind related compiler>(<compiler filename>)
FILE|<filename>:GUARD(<guard filename>)
FILE|<filename>:DCKEYIN(MP +IDENTITY <identity marker>)
FILE|<filename>:SECURITY(<security type and security use>)
FILE-CONTROL|<filename>:NOT(TO-EXAMINE)
FILE|<filename>:DCKEYIN(MP +PU)
FILE|<filename>:DCKEYIN(MP +PU TR)
FILE|<filename>:DCKEYIN(MP +TASKING)
FILE|<filename>:DCKEYIN(MP +TASKING TR)
FILE|<filename>:DCKEYIN(MP +SECADMIN)
FILE|<filename>:DCKEYIN(MP +SECADMIN TR)
FILE|<filename>:DCKEYIN(MP +CONTROL)
FILE|<filename>:DCKEYIN(MP +SUPPRESSED)
FILE|<filename>:DCKEYIN(MP +EXECUTABLE)
FILE|<filename>:DCKEYIN(MP +LOCKED)
FILE|<filename>:DCKEYIN(PP:UP)
FILE|<filename>:DCKEYIN(UP,SECADMIN)
FILE|<filename>:DCKEYIN(CP)
FILE|<filename>:DCKEYIN(SUPPRESS)
FILE|<filename>:DCKEYIN(XP)
FILE|<filename>:DCKEYIN(LP)
```

SURE verifies the entry of names by asking for user verification if a new name is entered.

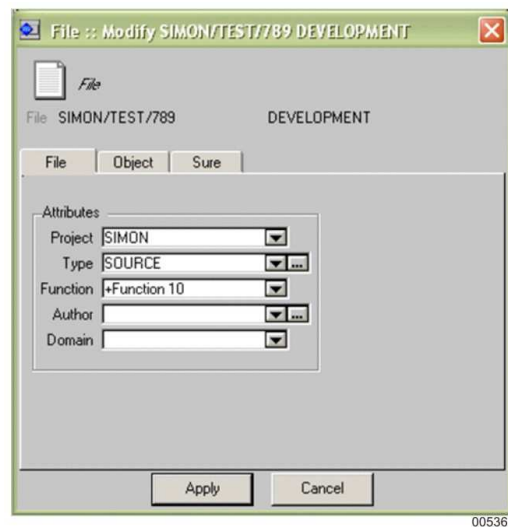
SURE treats these updates as updates to the file. If the relation-entity = FILE and the environment is not <global>, then the file is assigned to a task and the changes take effect in the next environment after promoting the task to that environment.

### 18.1.1. Allow to Define Multiple Functions for a File

It is possible to define multiple functions for a single file.

The function attribute is only used for documentation purposes. SURE does not contain any automatic procedures based on the function attribute. Sites can choose any function-name they like and link it to a file name.

It is possible to link multiple functions to a file using the modify properties screen.



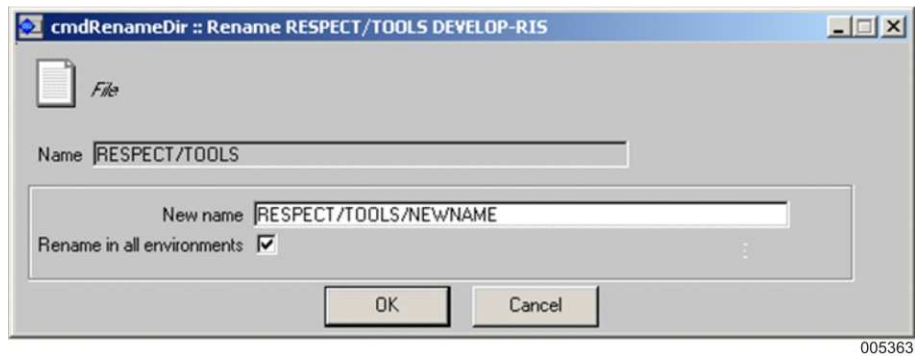
- Add an extra function by placing the "+" sign immediately in front of the function value.
- Remove an existing function by placing the "-" sign in front of the value.
- If you change the function-name but you do not use the "+" or "-" sign, then the value on the screen will be replaced by the new value.

Some examples of how to use the function in queries and macros:

- FUNCTION(<value>)
- FUNCTION(<value1>) OR FUNCTION(<value2>)
- FUNCTION(<value1> OR <value2>)
- FUNCTION(<value1>) AND FUNCTION(<value2>)

### 18.2. Rename (Popup Menu)

File names can be changed with the function RENAME (speed menu).



The RENAME is performed for the current environment and the hierarchical lower environments. The rename requires a task and promoting the task will rename the file in the higher environments, unless the option "Rename in all environments" is checked. In that case, the rename is done immediately in all environments and the command will not be linked to a task.

**Note:** The rename command will rename the FileName in the repository, but it does not modify sources. If a name of a copy file is renamed, then the sources using that copy file should be replaced manually.

### 18.3. Renamed PC Files are Removed on the Build Server

PC files that are deleted or renamed in SURE are also deleted from the build server. The build process reads in its initialization a deleted-or-renamed queue, and removes all the sources from the build-directories that are mentioned in that queue.

SURE keeps a deleted-or-renamed queue for each build-server on each environment.

A source is added to the deleted-or-renamed queue of an environment at the moment that it is deleted or renamed in that environment. That is

- With functions "Remove logically," Purge or Rename for a specific environment when option "for all environments" is disabled.
- With a transfer of a task that contains the "Logical delete" or Rename function.
- With functions "Remove logically," Purge, or Rename for all environments when option "for all environments" is enabled.



The build process performs the following actions:

- It reads the deleted-or-renamed queue of the environment; therefore, the build is started.
- For each source in that queue:
  - Delete the source from the build directories on the build server.
  - Remove the source from the deleted-or-renamed queue.

In case of a rename, the old file name is removed from the build server. The source is placed with its new name on the build server during the next phase of the build process.

## 18.4.Relate (Popup Menu)

It is possible to show all relations that point to a file or task, and modify these relations with the RELATE and MULTI RELATE commands.

### Function

### How

Show all relations of a file

Right-click on file name, select Miscellaneous, click Relation.

Show all relations of a task

Right-click on taskname, select Relation.

Relate: modify a relation of a file or task

Select a file or task from Tools menu, select Multi Relate.

or

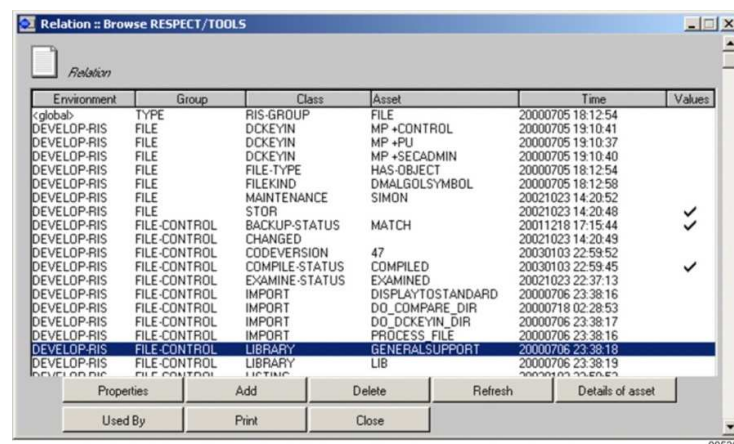
Show relations of file or task from Properties menu, select Modify.

Multi Relate: modify relations of a group of files or tasks

Select a group of files or tasks from Tools menu, select Multi Relate.

### Example

Show all relations of a file or task.



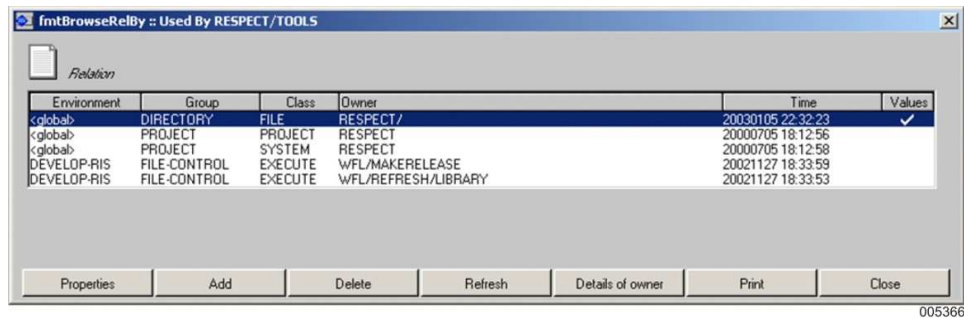
Column	Description
Environment	The environment where this relation is used. <global> = visible for all environment.
Group	Relations are grouped. Relations with the FILE group will be transferred to another environment. Relations with the FILE-CONTROL group will not be transferred.
Class	What kind of information is kept in the relation.
Asset	The value of the info.
Time	Each relation has its own timestamp.
Values	Four fields with variable info.

Button	Function
Properties	Show all details of the selected line and modify the properties of that relation.
Add	Add a new relation.
Delete	Delete the relation on the selected line.
Refresh	Refresh the screen.
Details of Asset	Show the relations of the "asset" in the selected line (in this case, library GENERALSUPPORT).
Used by	Show how the file or task in the title bar (in this case, RESPECT/TOOLS) is referenced.

The "Properties" button gives the properties of the current selected relation. The diagram shows four variables "relation values."



The “Used By” button shows how the file or task in the title-bar is referenced.



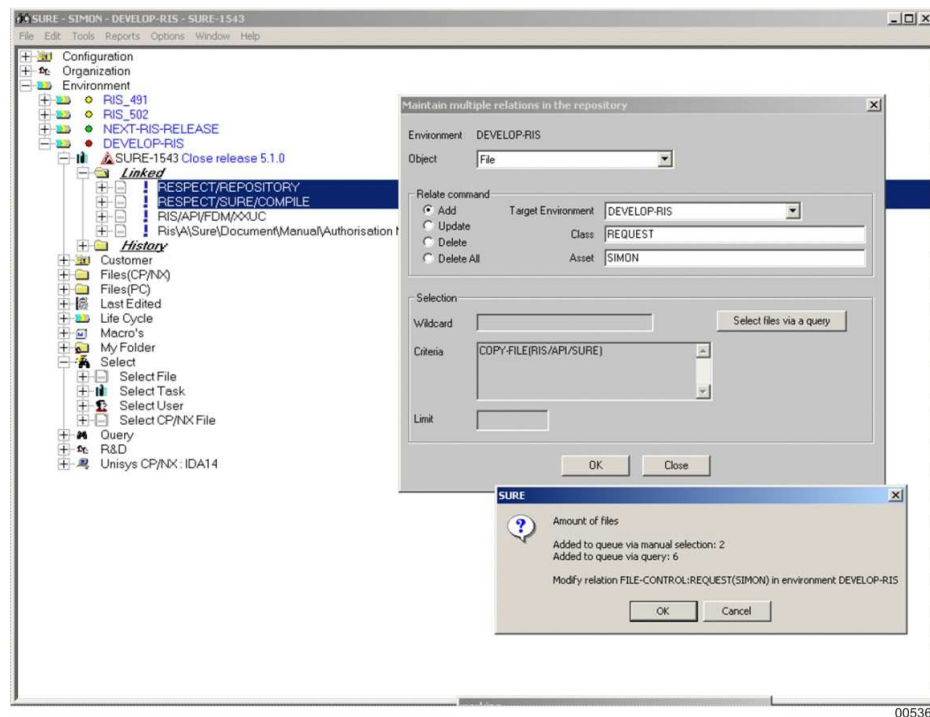
005366

### Column Description

**Owner** The file, item, or task that references the ID in the title bar (in this case, RESPECT/TOOLS).

### Modify Relations using “Multi Relate”

From Tools menu, select Multi relate.



005367

This example adds a relation with eight files in the DEVELOP-RIS environment.

Group	=	FILE-CONTROL (automatically determined by SURE)
Class	=	REQUEST
Asset	=	SIMON

Two of the files were selected in the browser; the other six files were selected using the "files with `COPY-FILE(RIS/API/TASK)`" query.

Field "Object" on the MultiRelate dialog identifies the group: files or tasks. If one or more tasks are selected in the SURE browser, then field "Object" is pre-filled with "Task"; otherwise, the field is pre-filled with "File."

It is possible to switch between "File" and "Task." The selected tasks in the browser are also added to the relate-queue if the object is "Task." The selected files in the browser are also added to the relate-queue if the object is "File."

Field "Environment" is pre-filled with the environment of the current selected line in the browser. It is possible to change the environment, but not if files or tasks are selected in the browser.

### Technical Details and Considerations

Most of the information on all screens is stored in the repository using relations. Modifying a relation will change the properties of a file or task on a screen in an uncontrolled way (this is similar to the case that a user is changing information in a dataset using ERGO or DMS-inquiry, instead of doing that using a regular application program). Therefore, it is not recommended to add, delete, or modify relations if you are not familiar with the concept of relations.

Function "Multi Relate" is secured with the "Multi Relate" authorization bit.

Modify a relation through the "Relations" screen is secured with the "Relate" authorization bit.

## Section 19

# Find and Replace Utilities

The FIND function is used to scan a group of sources for one or more target strings, or to scan a group of tasks for one or more target strings in the task-descriptions and solutions. The FIND function is not available for binary PC files.

The REPLACE function is used to scan a group of sources for one or more target strings and to replace these strings by corresponding destination strings (replace). The REPLACE function is only applicable for ClearPath server files.

Both functions scan only sources or tasks that are present in the repository. The replace function can only scan sources.

The total FIND or REPLACE function consists of the following steps:

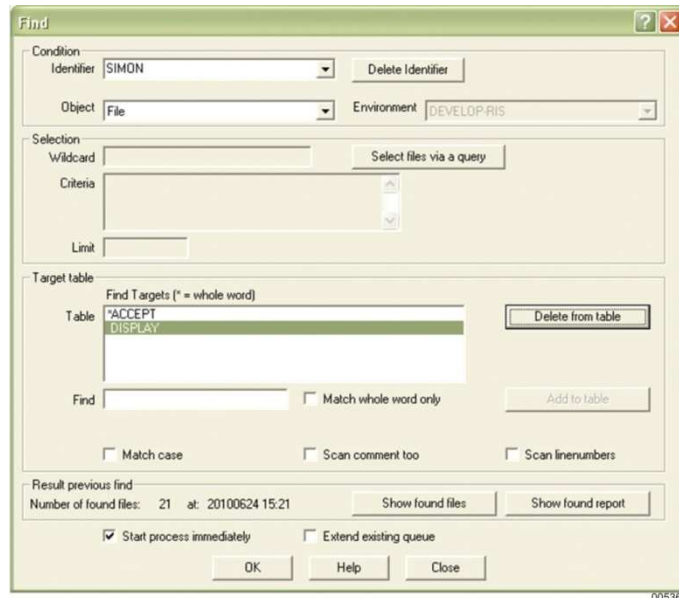
- Optional: Select the sources or tasks (that have to be scanned) in the browser.
- Start the procedure using the Find or Replace function on the Tools menu.
- Optional: Enter a query to select an additional group of sources or tasks that have to be scanned.
- All selected sources or tasks (step 1 and 3) are placed in a batch find or replace queue.
- Start the batch program that runs in the background and performs the actual find or replace function.

The output is present through the following mechanisms:

- printer output
- repository relations
- modified files

### 19.1. Find

The FIND dialog is opened using the Find function on the Tools menu.



Add a find-target as follows:

Enter a new find-target in the "Find" field and add it to the find-table using "Add to Table" button.

Delete a find-target from the table as follows:

Click a line in the find-table and use the "Delete from table" button.

It is possible to define multiple find-targets. In this example, the find table consists of two targets.

ACCEPT	The * sign at the beginning of the line identifies that this is a find "whole word"
DISPLAY	A find "literal"

If one of the targets is found in a file then that file is added to the find-output list.

#### Match whole word only

The "match whole word only" option is used for individual find-target. If this option is checked, then a star is placed before the target.

### Match case

"Match Case" is by default off, to enforce that the scan is not case-sensitive.

The "Match Case" option caters for all find-targets in the list.

### Scan comment too

Scan also comment lines (by default the comment lines are skipped).

### Scan linenumbers

Scan also the sequence numbers (by default the sequence numbers are skipped). This makes it possible to search for a specific sequence number.

### Intermediate spaces are compressed to one space.

For example, consider the find target "AAA BBB". If the source contains a line with "AAA BBB", then it will be selected because of the space compression.

### Search for a combination of targets

It is possible to scan for a combination of targets where all targets must appear in a given order in the source.

For example, consider the find target "CALL USING FOR 4 LINES."

A match will be found if a source contains the targets CALL and USING in this order in a range of four lines.

```
052900      CALL "CGET_APPLICATION_TITLE OF SER"
053000          USING APPL-DATAFILE-KY
053100              ,APPL-DATAFILE-PACK.
```

### Extra options

The first lines of the find table can be used to define extra options for the batch find program.

Statement @@DISKFILE <filename>	Route the output to a file.
Statement @@DESTINATION <name>	Route the output to a printer destination.
Statement @@-DESTINATION	Do not print the output automatically.
Statement @@CASE ON	Make the find case-sensitive.
Statement @@CASE OFF	Make the find not case-sensitive (default).
Statement @<char>	The "@" sign is used for technical purposes in the scanning routine, and therefore it is by default not possible to use this character in a find target. However, in the first line of the find-table it is possible to change this special character.

### 19.1.1. Start the FIND Process

The FIND command is automatically linked to a find-id. The find-id in the example dialog is SIMON. A user can define his find-ids. The results of a specific find action are linked to the find-id and remain available for the user until he uses that same find-id for another find action.

#### Start process immediately

If this option is set, then the RESPECT/SURE/FIND batch program is immediately started with the find-id as parameter when the find dialog is completed.

If this option is reset then the batch program is not started. In that case, the find-request is scheduled.

#### Extend existing queue

If this option is set then the existing (and scheduled) find-request (indicated by the find-id) is extended. If this option is not set and the find-request is scheduled or busy, then an error is given.

#### Start

Completing the FIND entry format with the strings to be found will result in the following actions:

- The selected files are added to the find queue using the relation FIND(<find-id>).
- If option "Start find process immediately" is set.
- Start the RESPECT/SURE/FIND program with the <find-id> as parameter.
- In the example, RUN RESPECT/SURE/FIND("SIMON").



- The RESPECT/SURE/FIND program produces a report containing all the lines that matched the find criteria. This report can be viewed using button "Show Found Report" on "Tools/Find."
- All files that contain one or more of the find-target-strings get relation FOUND(<find-id>). These files can be selected using the button "Show Found Files" on the Tools/Find dialog. The found files are then listed in folder "Query."
- A next find for the same <find-id> deletes the previous FOUND(<find-id>) relations.

### **Relations**

The following relations control the find process.

#### **Online**

- Delete the result of the previous SURE-find when a new find action is defined for the same find-id.

```
Delete          FILE-CONTROL|<filename>:FOUND(<find-id>)
```

- Add relations for the sources that must be scanned.

```
Add            FILE-CONTROL|<filename>:FIND(<find-id>)
```

#### **Batch**

- Select all files with relation FIND(<find-id>).
- Delete the find-trigger when the file is scanned.

```
Delete          FILE-CONTROL|<filename>:FIND(<find-id>)
```

- Add a found relation for the sources where a find-target is found.

```
Add            FILE-CONTROL|<filename>:FOUND(<find-id>)
```

## **19.1.2. FIND Results**

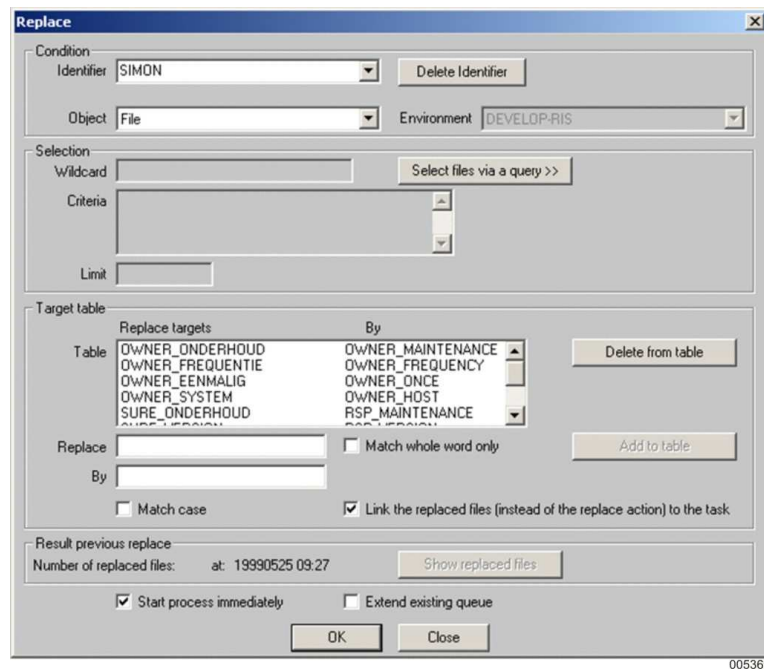
The FIND program creates an overview with the FIND results.

- The FIND output is now only written in a backup file.
- Each line where the find target was found is fully printed, including the line number and the mark-id.
- The name of the backup file is (<SURE-batch-usercode>)SUREFINDOUTPUT/<find-id> ON <SURE-batch-pack>.
- The backup file will not be removed automatically by the printing system.
- The backup file is overwritten when another SURE find is started by the same user with the same find identifier.
- The "Show found report" button downloads the backup file and opens it in notepad.

The [Show found files] button opens the Query folder with the programs names that contain one of the find-targets of the last FIND request. From there it is easy to view the files for further analysis.

## 19.2. Replace

The REPLACE dialog is opened using the function Tools/Replace.



Add a replace-target as follows:

- Enter a new replace-target in the fields "Replace" and "By" and add them to the replace-table with "Add to Table" button.
- Delete a replace-target from the table as follows:
  - Click a line in the replace-table and use the "Delete from table" button.

In this example, the string "OWNER\_UNDERHOUD" must be replaced by "OWNER\_MAINTENANCE," and so on.

### Match whole word only

The "match whole word only" option is used for individual replace-target. If this option is not checked, then a "@" sign is placed behind the target, which triggers a "replace literal" for that target.

### Match case

"Match Case" is by default off, to ensure that the replace is not case-sensitive.

The "match case" option caters for all replace-targets in the list.

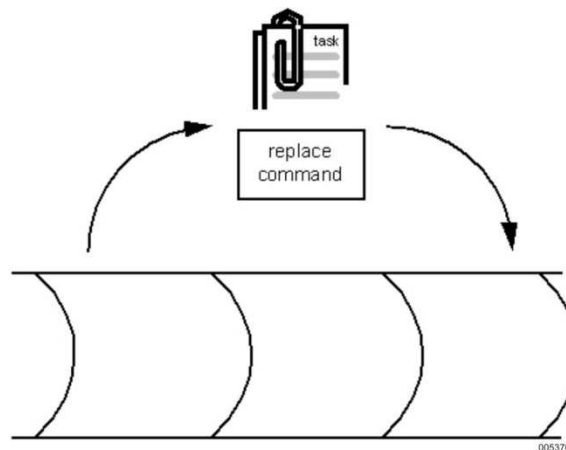
**Intermediate spaces are compressed to one space.**

For example, consider the replace target “AAA BBB”. If the source contains a line with “AAA BBB”, then it will be selected because of the space compression.

**Link the replaced files (instead of the replace action) to the task**

The replace function requires a current task, because it modifies sources in the repository. This task is used to transfer the replace action to the next environments. There are two methods to do this:

- Link the replaced sources to the current task and transfer these replaced sources to the next environments (the default method).
- Link the replace command + replace-tables to the current task and execute the replace action again on the next environments.



Consider the case that a big replace has to be done, and 1000 files are actually replaced. If all these replaced files are linked to the replace-task, then task-overlaps are created with all the other tasks that are linked to one or more of the replaced files. Then, it will be very difficult to transfer the replace-task to the next environment. In this situation, it is better to link the replace command to the task and to reprocess the replace again on the next environment after the task is transferred.

This procedure “link replace command to a task and reprocess replace action after transfer” has many extra checks and actions that guard the consistency of the files in the repository, such as:

- The transferred replace action will be started automatically by the next SURE batch.
- It is not possible to transfer a task with a replace-action to a next environment, if another replace action is still queued or busy on that environment. This may block the transfer of other tasks.

Consider the case that a new application system with 1000 files is loaded in SURE, and that some replaces have to be done before the files of that application-system are

transferred to the next environment. In this case, all the files are already linked to the same task: the task that was used to load the files. Now, it is better to link the replaced files to the task, because then the extra consistency checks and actions are not involved. In fact, such a replace is equal to: Checkout 1000 files; modify them all; check-in 1000 files.

### Extra options

The first lines of the find table can be used to define extra options for the batch find program.

Statement @@CASE ON	Make the replace case-sensitive
Statement @@CASE OFF	Make the replace not case-sensitive (default)
Statement @<char>	The "@" sign is used for technical purposes in the scanning routine, and therefore it is by default not possible to use this character in a replace target. However, in the first line of the replace-table it is possible to change this special character.

### 19.2.1. Start the REPLACE Process

The name of the batch program is RESPECT/SURE/REPLACE. Note that only one replace request can be in progress at any one time (for the whole repository).

The replace command is automatically linked to a replace-id. The replace-id in the example dialog is SIMON. A user can define his replace-ids. The results of a specific replace action are linked to the replace-id and remain available for the user until he uses that same replace-id for another replace action.

#### Start process immediately

If this option is set, then the RESPECT/SURE/REPLACE batch program is immediately started with the replace-id as parameter when the find dialog is completed.

If this option is reset then the batch program is not started. In that case, the replace-request is scheduled.

#### Extend existing queue

If this option is set then the existing (and scheduled) replace-request (indicated by the replace-id) is extended. If this option is not set and the find-request is scheduled or busy, then an error is given.

## Start

Completing the REPLACE entry format with the strings to be replaced results in the following actions:

- The selected files are added to the replace queue using the relation `REPLACE(<replace-id>.`
- If option "Start replace process immediately" is set.  
Start the RESPECT/SURE/REPLACE program with the <replace-id> as parameter.  
In the example: `RUN RESPECT/SURE/REPLACE("SIMON")`.
- During the batch procedure, the sources are interrogated for the strings specified in the replace-table. If one of the replace targets is found in a file, then that file is replaced and the changed source is saved in the repository.
- All replaced files get the relation `REPLACED(<replace-id>)`. These files can be selected using the "Show Replaced Files" button on the Tools/Replace dialog. The replaced files are then listed in the "Query" folder.
- A next replace for the same <replace-id> deletes the previous `REPLACED(<replace-id>)` relations.

## Relations

The following relations control the replace process:

### Online

- Delete the result of the previous SURE-replace when a new replace action is defined for the same find-id.

```
Delete          FILE-CONTROL|<filename>:REPLACED(<replace-id>)
```

- Add relations for the sources that must be scanned and replaced.

```
Add            FILE-CONTROL|<filename>:REPLACE(<replace-id>)
```

### Batch

- Select all files with relation `REPLACE(<replace-id>)`.
- Delete the replace-trigger when the file is scanned.

```
Delete          FILE-CONTROL|<filename>:REPLACE(<replace-id>)
```

Change the status of the replaced sources.

```
Add            FILE-CONTROL|<filename>:REPLACED(<replace-id>)
```

```
Change          FILE-CONTROL|<filename>:STATUS(<environment>)
```

```
Change          FILE-CONTROL|<filename>:CHANGED(<date-time>)
```

```
Add            FILE-CONTROL|<filename>:EXAMINE-STATUS(TO-EXAMINE)
```

```
Add            FILE-CONTROL|<filename>:REQUEST(MATCH)
```

Batch interface

The RESPECT/SURE/REPLACE program scans all files that are resident in the replace queue for replace targets, and replaces these targets by the corresponding strings.

Each replaced file is marked with the relation `REPLACED(<replace-id>)`. Files that are not replaced are not marked with this relation.

Restrictions

The function is secured by the REPLACE authorization.

When a file is in use by a user, a warning is given to remind the user to perform the replaces manually. The replace utility replaces only sources that are stored in SURE. A checked out file must be replaced manually by the programmer having the source under his usercode. At the moment that the programmer wants to save the source back into SURE with function SAVE, a warning is given, "source is replaced in SURE; compare the replaced version with your CANDE version."

A replace command is linked to a task. A task can only contain one replace command. Only one replace command can be active or scheduled at one point in time for each environment. A task with a replace command cannot transfer to a higher environment if another replace-command is still active or scheduled on that higher environment.

19.3.RESPECT/SURE/FIND

Run RESPECT/SURE/FIND as follows:

Input	parameters	<div><div></div><div>&lt;find id&gt;</div></div>
		: See "RESPECT/SURE/FIND" in Section 47.
	Task string	: <environment>.
Output	overview	: FIND listing. All lines in all selected sources that contain one of the find targets are printed.
Example	RUN RESPECT/SURE/FIND(" ");TASKSTRING = "DEVELOPMENT"	
Where	This program runs in the SURE evening batch. The program can be started from the SURE-browser through function FIND.	
Security	MP +PU	

A FIND command is linked to a find-id. The files that are selected for the find command are also linked to that find-id, they form a specific find-queue for that find-id. It is possible that multiple find requests are active at the same time, where each request has its own find-id. Each request forms its own find queue.

If the RESPECT/SURE/FIND program is started with a find-id as parameter then only that find-queue is processed. If the RESPECT/SURE/FIND program is started without a parameter then all scheduled find-queues are processed.

The RESPECT/SURE/FIND runs each evening (in the daily batch) without a parameter.

### Example

File	Class	Find-id
TEST/1	FIND	USERA
TEST/2	FIND	USERA
PROG/A	FIND	USERB
PROG/B	FIND	USERB
PROG/C	FIND	USERB

```
RUN OBJECT/RESPECT/SURE/FIND( "USERA" )
```

This example processes find-request USERA with programs TEST/1 and TEST/2.

```
RUN OBJECT/RESPECT/SURE/FIND( " " )
```

This example first processes find-request USERA with programs TEST/1 and TEST/2, and the find-request USERB with programs PROG/A, PROG/B, and PROG/C.

This mode is used in the SURE batch run and processes all the scheduled find-requests.

## 19.4. RESPECT/SURE/REPLACE

Run the RESPECT/SURE/REPLACE program as follows:

Input	parameter	: <input type="text" value="&lt;replace id&gt;"/>
		See "RESPECT/SURE/REPLACE" in Section 47.
	Task string	: <environment>.
Example	RUN RESPECT/SURE/REPLACE( " " ); TASKSTRING = "DEVELOPMENT"	
Where	This program can run in a user-written job. The program can be started from the SURE browser through function REPLACE.	
Security	MP +PU (the program accesses files in other directories).	

A REPLACE command is linked to a replace-id. The files that are selected for the replace command are also linked to that replace-id: they form a specific replace-queue for that replace-id.

If the RESPECT/SURE/REPLACE program is started with a replace-id as parameter then only that replace-queue is processed. If the RESPECT/SURE/REPLACE program is started without a parameter then all scheduled replace-queues are processed.

The RESPECT/SURE/REPLACE program runs each evening (in the daily batch) without a parameter.

### Example

File	Class	Replace-id
TEST/1	REPLACE	USERA
TEST/2	REPLACE	USERA

```
RUN OBJECT/RESPECT/SURE/REPLACE( "USERA" )
```

This example processes replace-request USERA, with programs TEST/1 and TEST/2.

```
RUN OBJECT/RESPECT/SURE/REPLACE( " " )
```

This example processes replace-request USERA with programs TEST/1 and TEST/2.

This mode is used in the SURE batch run and processes all queued replace requests.



## Section 20

# ADDS Support

The ADDS statements in ALGOL and COBOL files are recognized by the examine process. The ADDS structured information can be used for inquiry purposes in the query syntax. It can also be used to maintain application system integrity.

The RESPECT/SURE/EXAMINE program creates the following ADDS relations:

```
FILE-CONTROL|<filename>:ADDS-DICTIONARY(<adds-id>)  
FILE-CONTROL|<filename>:ADDS-PROGRAM(<adds-id>)  
FILE-CONTROL|<filename>:ADDS-FILE(<adds-id>)  
FILE-CONTROL|<filename>:ADDS-GROUP(<adds-id>)
```

## 20.1. Manual Integrity Trigger Overview

With the RESPECT/REPOSITORY program, it is possible to add a manual integrity trigger. This integrity trigger instructs the RESPECT/SURE/COMPILE program to create an integrity chain for files having a relation that is indicated by the trigger. All compiled objects that are in the same integrity chain are deployed together. If one compilation of the chain fails, nothing is deployed.

## 20.2. Manual Integrity Trigger Example

Consider two programs, F/1 and F/2, with relation `ADDS-FILE(MY/ADDS/FILE)`:

```
RUN OBJECT/RESPECT/REPOSITORY("SELECT CLASS ADDS-FILE ASSET MY/ADDS/FILE  
FUNCTION RELATE IMPACT MY/ADDS/FILE");TASKSTRING = "DEVELOP"
```

This selects all files with `class = ADDS-FILE` and `asset = MY/ADDS/FILE` and creates the following manual integrity triggers:

```
FILE-CONTROL|F/1:IMPACT(MY/ADDS/FILE)  
FILE-CONTROL|F/2:IMPACT(MY/ADDS/FILE)
```

The manual integrity trigger instructs the RESPECT/SURE/COMPILE program to create the following integrity chain:

```
FILE-CONTROL|F/1:INTEGRITY(MY/ADDS/FILE)  
FILE-CONTROL|F/2:INTEGRITY(MY/ADDS/FILE)
```

See “Integrity Mechanism” in Section 23 for more information.



## Section 21

# Patch Files

When programs are maintained using patch files, the original source remains unchanged, and the actual changes are placed in a patch file. At compile-time, the patch files are merged using the Unisys SYSTEM/PATCH facility into one original source before the compilation starts. Multiple patch files for one source are allowed.

Different programmers can maintain different patch files belonging to the same source at the same time. Each patch file must be taken into maintenance and production in the usual way. A patch file that is loaded in the SURE database but not linked to a source is not merged into a source.

**Note:** *If there are any \$ options in the sources, the \$ should be in the second column for ALGOL and there should be two (\$\$) in the first and second columns for COBOL74 sources. The position of this \$ option allows the option to be included in the resulting source.*

SURE handles patch files as follows:

- The master source is loaded in the repository in the usual way.
- The patch files for this source are loaded in the repository as separate files.
- The patch file always gets file type PATCH-FILE.
- The name of the patch file is automatically created by SURE according to a naming standard formula. This formula is linked to file type PATCH-FILE. It is possible to change the formula to your own naming standard. The default naming standard for the patch files is "PATCH/"<source>"/"<3 digits number>.
- Function "Check Out" on the master source creates a new patch file for that source.
- Special relations connect the patch files to the original source.
- At compile-time, the related patch files are merged into the source before the compilation starts.
- Sequence-range overlap will not be checked by SURE. As long as the SYSTEM/PATCH agrees with the specified sequence-numbers in the original source and the related patch files, SURE will compile the program and transfer it to production.
- A patch file can be merged into the original source with function "Merge patch file." If this function is used for the original source, then all its patch files are merged into it. If this function is used for a patch file, then only that patch is merged into its original source.

The master source should not be changed directly. Therefore, it is not possible to modify a master source. However, in emergencies, the following procedure can be used to change it (using RIS/MENU):

Copy the source to disk using the COPY function.

- Modify the source.
- Use the function CHECK IN (only for authorized users) or OBJECT/RESPECT/SURE/LOAD to save the source back in the repository.
- Use the COMPILE function to merge and compile the program and set the master source back into status production.

### Relations

The PATCH function adds the following relation:

```
<mastersource-name>:PATCH FILE(<patch filename>) with an additional sequence number.
```

### Restrictions

The PATCH function is secured with the SURE authorization PATCH.

## 21.1. (Patchfiles) Baselines

It is possible to define extra baselines for an application system. The default development baseline is the environment itself. A user-defined baseline is a separate workspace where a team of developers can work on a large project parallel to other baselines. This method of development is not supported for PC files.

In the case of baselines, the sources are maintained using patch files.

For more information on Maintenance using Baseline, refer to the *SURE Assistant* (power point) from SURE Help menu show, select Source management using multiple baselines for a detailed example.

## 21.2. Integration of Resources, Patch Files, and Compile Queues

Consider the following scenario:

A company has medium to large sources and multiple developers make changes to these sources at the same time. For this concurrent file maintenance, the patch file mechanism is enabled. These sources refer to other sources, using copy or include statements. These referred sources contain record layouts and generic routines.

The company has different application systems and each of these application systems has its own release schedule. In principle, every month, quarter, and year, a release of the application system is provided.

Therefore, a source is maintained at the same time by different developers, which are busy with a task for a possible different release schedule.

This means that a developer writing a patch for the monthly release needs a source with all the patches currently in production and all the patches which are included in the monthly release. The patches scheduled for the releases coming are not yet relevant for him. Secondly, he needs the layout of the files currently in production and he does not need the changes scheduled for the yearly release.

The last requirement of this company is that they want to deploy all new objects on Tuesday, but they want to compile upfront, so that they can react on problems during compilation, such as patch overlap and other incidents. After this compilation they still want to be able to make a quick fix (a fix on the production executable).

### The Configuration

The above described scenario is achieved by using the SURE entities Compile-Queue, Resource-definition, Task, and Patch-files.

**Compile-Queue** A compile queue named TUESDAY. This compile queue is activated on environment PRODUCTION and compilations are done under a specific usercode, named TUESDAY. This means that the compile job under this usercode can COMPILE every evening, but only on TUESDAY the software is deployed from this usercode.

**Resource-definition** A resource-definition contains characteristics from where resources are retrieved. A resource is a database description or a copy or include file. The resource-definition defines where these resources are located. There are three resource definitions, MONTHLY, QUARTERLY, and YEARLY. MONTHLY and QUARTERLY retrieve the resources from PRODUCTION while YEARLY retrieves the resources from DEVELOP. This resource-definition alters the source when checked out and adds usercodes and packnames to the database declaration and copy statements, so that it locates these resources at the correct physical location. Later, when the file is checked in, these statements are corrected again and the physical locations are removed.

**Task** One task is used as a release holder (one task each release, monthly, quarterly, and yearly) and connected to this task are the functional tasks: the tasks that describe the functional requirements. So, one release task controls many functional tasks and each develops works for a functional task.

The release task contains the following attributes.

DELIVERY DATE	date scheduled for delivery of the task
RESOURCES	the resource definition, which will be MONTHLY, QUARTERLY, or YEARLY, depending on the release type
COMPILE-QUEUE	TUESDAY

**Patch-files** The patch file mechanism is enabled each application system. If enabled, the developer may check out a file using patch files. The following rules apply:

- By default, the dialog will leave patch files disabled.
- Patch files can only be enabled if the file version is equal in all environments.
- Patches are merged into the source if all patch files have equal versions in all environments. This process is done by the evening batch program RESPECT/SURE/MATCH (optional).
- If a source has patch files, it cannot be maintained in full source mode.
- By default, the name of the patch file is PATCH/<source>/<TaskName>, but the suffix <TaskName> may be altered at check out time. This suffix may be important, because it plays a role in the order that the patch file is applied.
- If a file is checked out, the check-out dialog contains a list of current patches. This list is sorted as follows:
  - All solved tasks in the order that they were solved.
  - If the solved date is equal, then the suffix is used in the sort order.
  - All non-solved tasks with a date LESS or EQUAL to the delivery date of the task worked on.
  - If your task has no delivery date, none of the non-solved tasks will be used.
- The patch-dialog contains an indication whether the patch is included. By double clicking the line, this indication is reversed.

### How It Works?

The functionality is explained by an example.

The PROG/SRC/X source has three patch files, of which number 0020 is in production, and the other two are checked in at development. These two patches are for releases further in the future.



005371

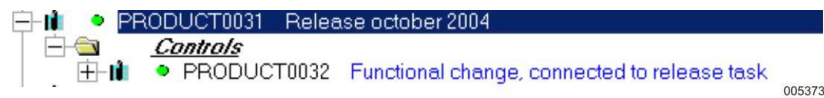
There is a “release” task with the following properties.

Next Release  
 Compile queue TUESDAY  
 Resource MONTHLY  
 Master ☐  
 Power ☐

Planning  
 Effort (hrs)  
 Entered 09/09/2004  
 Delivery 01/10/2004

Status  
 Solved in r  
 005372

There is a working task, dependent of this release task, which is used by the developer. This task does not have these special properties; it inherits them from the release task.



The developer checks out the PROG/SRC/X source and the following dialog appears.

Check Out: PROG/SRC/X

Environment DEVELOP

Local editing  
☒ Enabled  
☒ Edit/Open the file after CheckOut  
☐ Do Not overwrite existing file  
 Compare with Repository File

Local work file E:\DATA\RI\RUNTIME\DEVELOP\WORK\PROG\SRC\X\ALG  
 Local source file E:\DATA\RI\RUNTIME\DEVELOP\SOURCE\PROG\SRC\X\ALG

SURE options  
 File task: PRODUCT0032: Functional change, connected to release task  
☐ Replace existing file on server  
☐ Don't copy include files

Resource  
 Qualified name  
 \*REF/COPY/A ON BEHEER (in version <PRODUCTION>)  
 \*PROG/COPY/A ON BEHEER (in version <PRODUCTION>)

Patches  
☒ Change via patch: PRODUCT0032

Patch	Delivery D...	Incl...
PATCH/...../PRODUCT0020	08-09-2004	Y
PATCH/...../PRODUCT0021	01-11-2004	N
PATCH/...../PRODUCT0022	01-01-2005	N

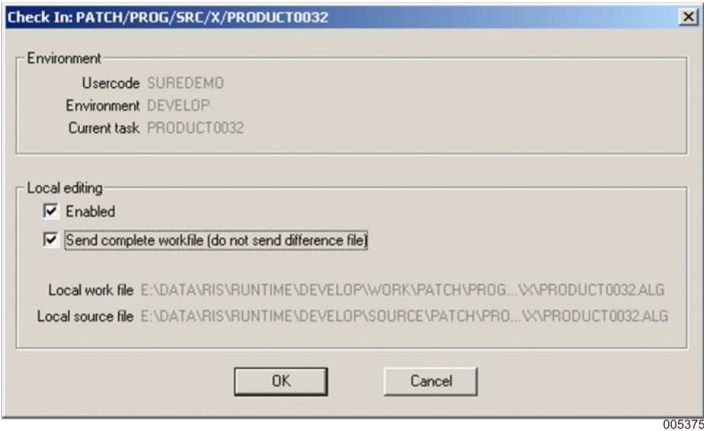
OK Cancel

005374

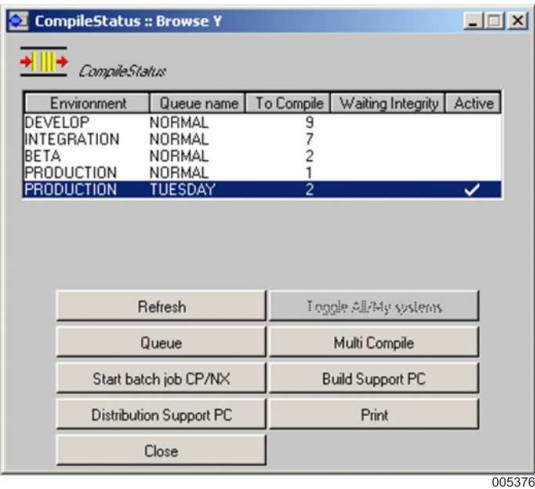
- Because it is a task, which has the RESOURCE definition MONTHLY, it relocates the defined copy files to the PRODUCTION location.
- The patch list contains the patches included when doing the check out. Number 0020 is included, but the other patches are not included. You can override this by double clicking the line of the patch that you want to alter.
- You can only check out using path files, because there are patch files currently connected to the source.
- The name of the suffix is, by default, the name of the task; it is possible to override this.
- Changing the task will update the resource and the patch list.

Patch Files

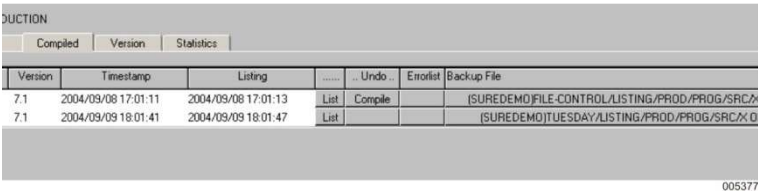
Check in the patch file, use send complete work file if you want to retain your own sequence numbers; otherwise, SURE will create sequence numbers for the changed lines.



When the release task is transferred to production it takes all dependent tasks with it. The task is linked to compile-queue TUESDAY and therefore all sources that are linked to the task are placed in that queue. It is now possible to perform upfront compilations and tests.



You can validate the compiler listing for the results of the compilation, if required. This can be done using the file properties on the compile tab. Click the backup file for viewing the listing.

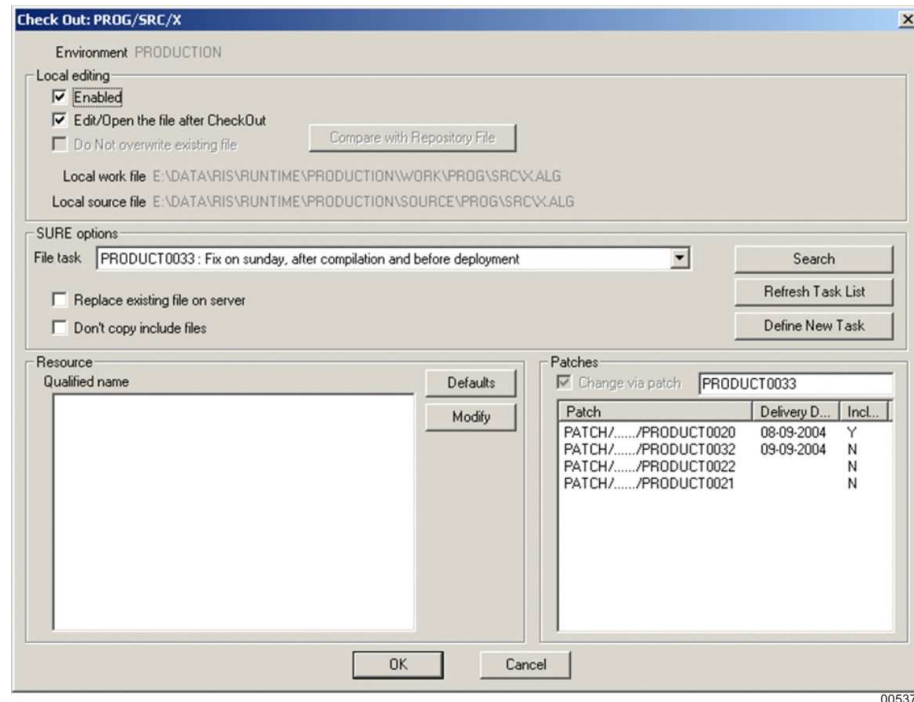




## Step 2 of the Example

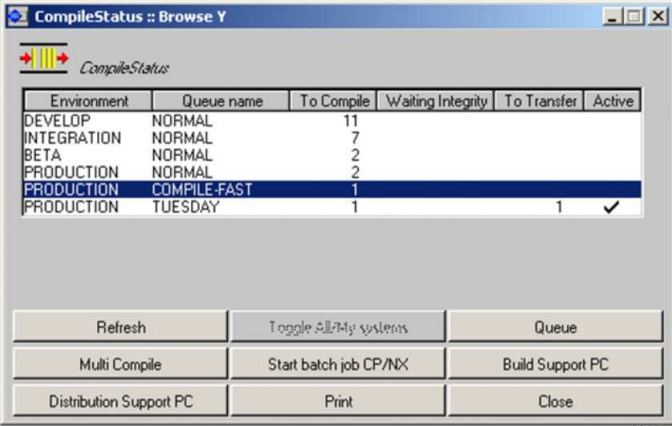
Create a quick fix task which does not have any special attributes, except that it is a quick fix.

Use this task to check out the file again (notice that we do not want the new patch included yet, because that is scheduled for Tuesday and only compiled but not deployed yet). When checked out, the following dialog is shown.



- The resource part is empty, because use of resources was only defined at development.
- The patch list contains all patches, the one in production 0020 is included, and the one scheduled and compiled 0032 is not included.
- The retrieved source only contains the patches actually in production, so omitting the one in the TUESDAY compile queue.

After check in, the compile interface shows the following state.



The screenshot shows a window titled "CompileStatus :: Browse Y". Inside, there is a table with the following data:

Environment	Queue name	To Compile	Waiting Integrity	To Transfer	Active
DEVELOP	NORMAL	11			
INTEGRATION	NORMAL	7			
BETA	NORMAL	2			
PRODUCTION	NORMAL	2			
PRODUCTION	COMPILE-FAST	1			
PRODUCTION	TUESDAY	1		1	✓

Below the table are several buttons: Refresh, Toggle All My systems, Queue, Multi Compile, Start batch job CP/NX, Build Support PC, Distribution Support PC, Print, and Close. The number 005379 is visible in the bottom right corner of the window.

The source is scheduled for compilation in the COMPILE-FAST queue, where the production version with the fix will be compiled. The source is also scheduled for compilation in the TUESDAY queue again, because the FIX should also be incorporated in the TUESDAY release. The contents and results of these compilations may be verified using the compiler listings. The quick fix modification should be reprocessed in the usual way.

## 21.3. Merge Patchfile Permanently into the Main Source

A patchfile which is transferred and stable in production can be merged into the main source. There are two methods to merge stable patchfiles: the manual and the automatic procedure.

### 21.3.1. Merge Patchfiles Manually

Dealing with patchfiles within large teams requires some manual validation and control. This chapter describes the various functions from a functional and a technical viewpoint.

#### Configuration

The patch files are configured on PROJECT level:

From the Configuration menu, select System, select <System-name>, select <project-name>, click Project-properties.

- Merge patch files automatically

If this option is set, the RESPECT/SURE/MATCH program merges all the production patch files if they match the "after # days" criterion. If this option is not set, the patchfiles needs to be merged manually.

- Merge solved patchfiles after # days

This option defines the number of days after a patch file is taken into production, that it is merged into the source. If this field is not filled, every patch file in production is merged into the source.

## **DEVELOPER**

- CHECKOUT

The developer uses the CHECKOUT function that allows him to select which patches are included (based on delivery date) and which resources (copy files) are used. The developer then gets a source file including the selected patches.

- COMPILE

The COMPILE function on the patch files compiles the source in maintenance and produces an object name based on the source name (which contains the patches).

- CHECK-IN

CHECK-IN creates a patch file and stores it in the repository. There is a check performed on patch conflict. This check merges all patch files into the source and indicates if there is a conflict in any of the patch files, thus not only the checked in patch file. If there is a conflict, the details can be shown using the produced backupfile of system or patch. Refer to "Appendix A."

Furthermore, the developer can see his changes using View my Changes. Refer to "Appendix B."

- COMPARE and VALIDATE

If the developer needs to view the changes in a specific patch file, the VIEW function on a patchfile just lists the contents of the patchfile. If the patch file is selected and using the CTRL button the source files is also selected. The patchfile difference is shown in context of the source; refer to "Appendix C."

## **PROJECT LEADER**

- TRANSFER

At transfer time, the project leader is informed if there is any conflict situation in the environment transferred to. If so, a listing is present (as indicated) and output is produced as described in "Appendix A." The project leader can then verify the conflicts.

- CHECK CONFLICTS

The project leader verifies the conflicts from the report as shown in "Appendix A."

- VALIDATE CONFLICTS DEVELOPMENT

Before merging the patchfiles (or at any other desired time), the project leader can, using Miscellaneous and Verify conflicts ALL patchfiles start a conflict report for all patchfiles on a source. Using the report as shown in "Appendix A," it can be verified.

- VALIDATE CONFLICTS BEFORE MERGE

Before merging the patchfiles (or at any other desired time), the project leader can, using Miscellaneous and Verify conflicts IN PRODUCTION patchfiles start a conflict report for the patchfiles in production on a source. Using the report as shown in "Appendix A," it can be verified.

- MERGE PRODUCTION PATCHFILES

At any time, the project leader can merge the patches into the source using the Miscellaneous or Merge patchfile. The setting of the project option "merge solved patchfiles after # days" determines which patchfiles are included. The printer output in "Appendix D" shows the output of the RESPECT/SURE/MATCH program which actually merged the patchfiles.

## Appendix A

```
DATE Wednesday, November 1, 2006 09:45:24 SYSTEM/PATCH COMPILED 08/10/2004 01:29 PM SSR 50.1 (50.189.8005)

$.SET NEW COBOL74

=====

PATCH NUMBER      1
000001$# PATCHDECK
000002$.FILE SUREPATCH/PATCH/SRC/PATCHFILES/DBP/0015/T_0400
002202 * patch 2 D      |T_0400      (DEVELOPER) SUREPATCH/PATCH/SRC/PA
002203 * patch 4 D      |T_0400      (DEVELOPER) SUREPATCH/PATCH/SRC/PA
002501 * patch A D      |T_0400      (DEVELOPER) SUREPATCH/PATCH/SRC/PA
002510 * patch B D      |T_0400      (DEVELOPER) SUREPATCH/PATCH/SRC/PA
002601 * test C D       |T_0400      (DEVELOPER) SUREPATCH/PATCH/SRC/PA
002620 * some other lines 1 |T_0400      (DEVELOPER) SUREPATCH/PATCH/SRC/PA
002621 * some other lines 1 |T_0400      (DEVELOPER) SUREPATCH/PATCH/SRC/PA
002622 * some other lines 1 |T_0400      (DEVELOPER) SUREPATCH/PATCH/SRC/PA

=====

PATCH NUMBER      2
000003$# PATCHDECK
000004$.FILE SUREPATCH/PATCH/SRC/PATCHFILES/DBP/0015/T_0402
003210 * test patch      |T_0402      (DEVELOPER) SUREPATCH/PATCH/SRC/PA

=====

PATCH NUMBER      3
000005$# PATCHDECK
000006$.FILE SUREPATCH/PATCH/SRC/PATCHFILES/DBP/0015/T_0396
002203 * patch 4         |T_0396      (DEVELOPER) SUREPATCH/PATCH/SRC/PA

THE FOLLOWING CONFLICT(S) WERE ENCOUNTERED IN PATCH 3

+ 002203 * patch 4         |T_0396      REPLACED
- 002203 * patch 4 D      |T_0400      PATCH 1

=====

PATCH NUMBER      4
000007$# PATCHDECK
000008$.FILE SUREPATCH/PATCH/SRC/PATCHFILES/DBP/0015/T_0392
002202 * patch 2         |T_0392      (DEVELOPER) SUREPATCH/PATCH/SRC/PA
000009$.EOF

THE FOLLOWING CONFLICT(S) WERE ENCOUNTERED IN PATCH 4

+ 002202 * patch 2         |T_0392      REPLACED
- 002202 * patch 2 D      |T_0400      PATCH 1

0 ERROR(S) WERE FOUND      0 WARNING(S) WERE ISSUED      2 CONFLICT(S) WERE ENCOUNTERED

*****
TITLE OF OLD SYMBOLIC FOR THIS RUN WAS
(DEVELOPER)PATCH/CONFLICT/1778 ON IDR
CREATED : 11/01/2006
*****

DATE Wednesday, November 1, 2006 09:45:24 SYSTEM/PATCH COMPILED 08/10/2004 01:29 PM SSR 50.1 (50.189.8005)
```





### Merge patchfiles depending on project options

```
Patch PATCH/SRC/PATCHFILES/DBP/0015/T_0392 patchfile not in PRODUCTION
Patch PATCH/SRC/PATCHFILES/DBP/0015/T_0396 patchfile not in PRODUCTION
Patch PATCH/SRC/PATCHFILES/DBP/0015/T_0400 ----- will be merged <-----
Patch PATCH/SRC/PATCHFILES/DBP/0015/T_0402 ----- will be merged <-----
```

```
Start loading...
Merged source is now loaded in SURE
Delink patch: PATCH/SRC/PATCHFILES/DBP/0015/T_0400
Delink patch: PATCH/SRC/PATCHFILES/DBP/0015/T_0402
```

### 21.3.2. Merge Patchfiles Automatically

The RESPECT/SURE/MATCH program offers the auto-merge function for patch files

- Stable patch files are merged permanently into the main source and are delinked from that source.
- The new version of the source (with the stable patches merged in it) is saved back into SURE.

The auto-merge function is by default disabled. The function must be enabled using the option “merge solved patch files after” on the global option screen, tab “history.”

1. Right-click the Environment folder.
2. Click **Global Options**.
3. Select the **History** tab.

The screenshot shows the 'Task / Global' dialog box with the 'history' tab selected. The dialog contains several sections for configuring historical records and parameters.

**Definition for historical records in repository**

	Days	Min Entry	Max Entry
Remove log information after	2		
Remove file taskhistory after	30		
Remove solved tasks after	30		

**Parameters for RESPECT/SURE/CLEANER**

	Days
Remove datafiles after	0
Remove sources files after	0
Remove object files after	0

**Merge solved patchfiles after**

	Days
Merge solved patchfiles after	0

Buttons at the bottom: Apply, Delete Line, Insert Line, Print, Close.

005383

If the “Merge solved patch files after” field is empty then the option is disabled.

If the field is entered, then the number determines the “merge delay period” (in days). A patch file will only be merged permanently into the main source if that patch file is stable in production for that amount of days.

If the field is not entered, one must use the manual method to merge patchfiles.

In this example, the merge-delay period is zero, so the patch files are merged during the first run of the RESPECT/SURE/TRANSFER after they are transferred to production.

## Patch Files

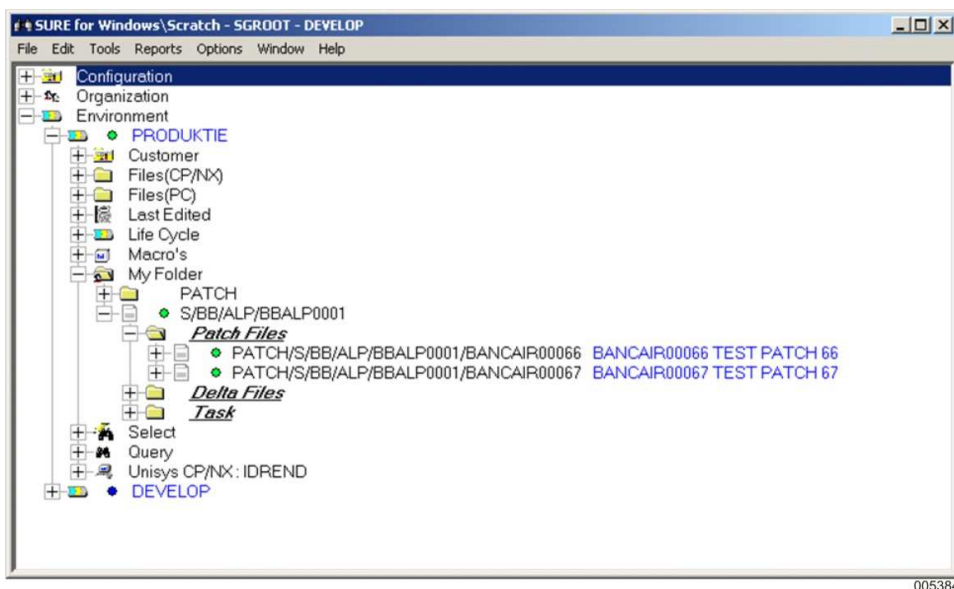
The following global conditions of a main source must be true before any of the patch files of that main source can be merged permanently:

- The main source must be equal in all environments.
- The main source may not be connected to a user-defined compile queue.
- The original full source (main source + patches before the merge) must be exactly the same as the resulting full source (main source + patches after the merge). The next paragraph gives an example.

The following conditions of a patch file must be true before that patch file is merged permanently:

- The patch file must be transferred to the production environment.
- The task of the patch file must have status solved.
- The merge-delay period must be expired for that patch file.
- The patch file may not be connected to a user-defined compile queue.
- The patch file may not be resident in a recompile-control-queue (baselines only).
- The patch file must be equal in all environments.

### Example



This is a source with two patch files.

Both patch files are transferred to production and are linked to a task with status solved. Task BANCAIR0066 was transferred first to production, before task BANCAIR0067.



This means that when the source is compiled, the order of merging the patches into the source is determined by the names of the patch files. So, the patch `.../BANCAIR0066` is merged first and the patch `.../BANCAIR0067` is merged secondly (at compile time).

The `PATCH/S/BB/ALP/BBALP0001/BANCAIR0066` patch file has the following contents:

```
00001260    DISPLAY( "66 A" );
00001300    DISPLAY( "66 B" );
```

The `PATCH/S/BB/ALP/BBALP0001/BANCAIR0067` patch file has the following contents:

```
00001300    DISPLAY( "67 A" );
00001460    DISPLAY( "67 B" );
```

There is an overlap between the two patch files at line 1300.

The following is the compile listing when the source is compiled. At line 1300 you see the line of patch file `.../BANCAIR0067` which proves that patch file is merged after the patch file `.../BANCAIR0066`.

```
( S G R O O T ) O B J E C T / S / B B / A L P / B B A L P 0 0 0 1
= = = = =

SOURCE TAPE:      (SGROOT)SURESOURCE/S/BB/ALP/BBALP0001 ON IDR.D.

$ SET LISTINCL          C 00000000 0000:0000:0
$ RESET WARNSUPR        C 00000000 0000:0000:0
$SET PROD               C 00000000 0000:0000:0
$ SET BASE              C 00000000 0000:0000:0
$ SET ALP               C 00000000 0000:0000:0
$ SET ALGOL             C 00000000 0000:0000:0
BEGIN                   00000500 0000:0000:0
                        BLOCK#1 IS SEGMENT 0003
                        1 00001000 0003:0000:1
                        00001050 0003:0002:2
                        00001100 0003:0004:4
                        00001200 0003:0007:0
                        00001250 0003:0009:2
                        00001260 0003:000B:4
                        00001300 0003:000E:0
                        00001400 0003:0010:2
                        00001450 0003:0012:4
                        00001460 0003:0015:0
                        00001500 0003:0017:2

        DISPLAY( "PROGRAM" );
        display( "62" );
        DISPLAY( "BANCAIT0054 1" );
        display( "bancair0061 1" );
        DISPLAY( "PATCH 60" );
        DISPLAY( "66 A" );
        DISPLAY( "67 A" );
        DISPLAY( "BVANCAIT0061 2" );
        DISPLAY( "61" );
        DISPLAY( "67 B" );
END.
```

### CASE 1

Patch ../BANCAIR0066 will not be merged because it is still linked to a compile queue.

Patch ../BANCAIR0067 is ready to be merged.

The following is a piece of the merge overview, created by the RESPECT/SURE/MATCH program:

```
(SCRATCH)INFDB ON IDRD2          DEVELOP          20050407 17:54:45 Page    1

Merge patch files that are solved since 20050407 17:54:45

-----
Source S/BB/ALP/BBALP0001
Patch  PATCH/S/BB/ALP/BBALP0001/BANCAIR00066      still in compile queue SIEM in
                                                environment PRODUKTIE
Patch  PATCH/S/BB/ALP/BBALP0001/BANCAIR00067      -----> will be merged <-----
Start merging...
Comparing files before and after the merge....
ERROR: line not equal
      before:      DISPLAY("67 A");              00001300
      after:       DISPLAY("66 B");              00001300
-----
```

The RESPECT/SURE/MATCH program did the following:

1. Check which patch files are ready to be merged permanently.
2. Start the merge if one of the patches is ready to be merged.
3. Compare the original full source (original main source + patches) before the merge with the new full source (new main source + patches) after the merge. If both files are not completely equal then the permanent merge is skipped.

The original full source before the merger is

```
Source S/BB/ALP/BBALP0001 from SURE version 4.7
+ patch ../BANCAIR0066
+ patch ../BANCAIR0067
```

The new full source after the merger is:

```
Source S/BB/ALP/BBALP0001 merged with patch ../BANCAIR0067
+ patch ../BANCAIR0066
```

In this example, the full source before the merge does not have the same contents as the full source after the merge, there is a problem on line 1300 and that is the line with the patch overlap. The cause of this problem is that the order how the patch files are merged is changed. In the original situation, the patch ../BANCAIR0067 was merged after the patch ../BANCAIR0066 (see the compile overview), but in the new situation, the patch ../BANCAIR0067 is merged permanently into the source, so it is always merged before the patch ../BANCAIR0066.

The new main source will only be loaded in SURE when the original full source is exactly the same as the new full source. In this case, the two full sources are not equal, so the permanent merge of this source is aborted, and the original situation remains. Nothing happens for the source S/BB/ALP/BBALP0001 and its patches.

**CASE 2**

In this case, the patch file ../BANCAIR0066 is not linked anymore to a compile queue, so it is stable.

Patch ../BANCAIR0066 is ready to be merged.

Patch ../BANCAIR0067 is ready to be merged.

The following is a piece of the merge overview, created by the RESPECT/SURE/MATCH program:

```
(SCRATCH)INFDB ON IDRD2          DEVELOP          20050408 10:13:41 Page    1

Merge patch files that are solved since 20050408 10:13:41

-----
Source S/BB/ALP/BBALP0001
Patch  PATCH/S/BB/ALP/BBALP0001/BANCAIR00066          -----> will be merged <-----
Patch  PATCH/S/BB/ALP/BBALP0001/BANCAIR00067          -----> will be merged <-----
Start merging...
Comparing files before and after the merge....
Start loading...
Merged source is now loaded in SURE
Delink patch: PATCH/S/BB/ALP/BBALP0001/BANCAIR00066
Delink patch: PATCH/S/BB/ALP/BBALP0001/BANCAIR00067
-----
```

The RESPECT/SURE/MATCH program did the following:

1. Check which patch files are ready to be merged permanently.
2. Start the merge if one of the patches is ready to be merged.
3. Compare the original full source (original main source + patches) before the merge with the new full source (new main source + patches) after the merge. If both files are not completely equal then the permanent merge is skipped.

The original full source before the merger is:

```
Source S/BB/ALP/BBALP0001 from SURE version 4.7
+ patch ../BANCAIR0066
+ patch ../BANCAIR0067
```

The new full source after the merger is:

```
Source S/BB/ALP/BBALP0001 merged with patches ../BANCAIR0066 and
../BANCAIR0067
```

4. The original full source is the same as the new full source, so the new main source is loaded in SURE.
5. If the source is correctly loaded in SURE, the merged patch files are delinked from the main source.

**END CASE**

The example shows that the result of the permanent merge of a range of patch files is always equal to original situation where those patch files were merged temporarily.

The example in CASE-1 where the patch file is not merged happens basically only in the case that there is an overlap in the patch files and when some of those overlapping patch files are not yet ready to be merged permanently. The situation will be solved automatically a few days later, when all overlapping patches are ready (in this example: after the user defined compile queue was closed).

An important feature of the merge process is that the "source + patchfiles before the merge" is compared with the "source + patchfiles after the merge." If these two "total" files are not equal then the new source is not loaded in SURE and the merge is skipped.

### Example

Consider source P1 and two patch-files PATCH/P1/T1 and PATCH/P1/T2. The PATCH/P1/T2 patchfile is ready to be merged and PATCH/P1/T1 patchfile is not ready to be merged. In this case, it is checked that  $P1 + \text{PATCH/P1/T1} + \text{PATCH/P1/T2}$  is equal to  $(P1 + \text{PATCH/P1/T2}) + \text{PATCH/P1/T1}$ .

Notice the parentheses: PATCH/P1/T2 becomes a part of source P1 and PATCH/P1/T1 remains a separate patchfile.

If the above comparison is not OK, then the merge of PATCH/P1/T2 is skipped and that patch remains a separate patchfile. The merge check is done again at the next run of the RESPECT/SURE/MATCH.

The reason that the comparison fails is most of the times that the order of the patchfiles changes after the merge. In the above example: the original order of the patchfiles is PATCH/P1/T1 followed by PATCH/P1/T2 and the new order is just the other way around.

The order of the patchfiles BEFORE and AFTER the merger is written in the merge overview.

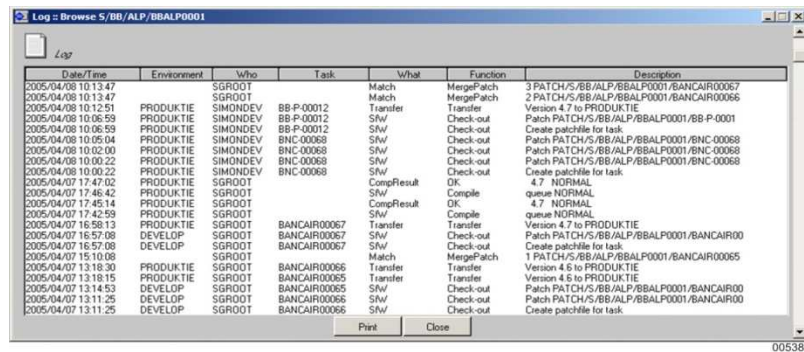
```
-----
Source SRC/PATCHFILES/DBP/0019
Patch PATCH/SRC/PATCHFILES/DBP/0019/T_0203      still in compile queue
REORGANIZATION Patch PATCH/SRC/PATCHFILES/DBP/0019/T_0204 --> will be merged <---
Start merging...
Comparing files before and after the merge....
ERROR: line not equal
    before: END.                                00002500|
    after:  DISPLAY("LINE 2A");                  00001500

Order of patchfiles BEFORE the merge      Order of patchfiles AFTER the merge
P:PATCH/SRC/PATCHFILES/DBP/0019/T_0203  M:PATCH/SRC/PATCHFILES/DBP/0019/T_0204
P:PATCH/SRC/PATCHFILES/DBP/0019/T_0204  P:PATCH/SRC/PATCHFILES/DBP/0019/T_0203
```

**Note:** The order of the patchfiles is changed in this example. The patches that are merged in the source (in the second column) are marked with "M."

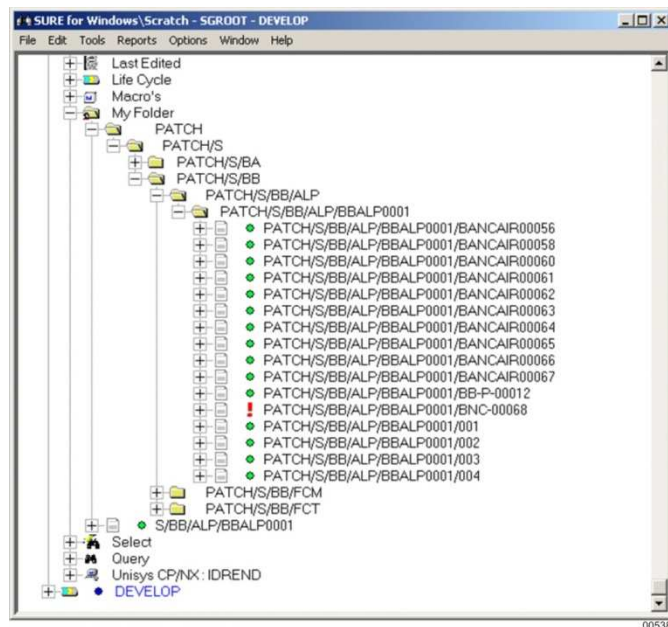
## Other Features of the Automatic Merge Function

- The fact that a patch file is merged permanently is added to the log of the main source.



Date/Time	Environment	Who	Task	What	Function	Description
2005/04/08 10:13:47	SGROOT	SGROOT		Match	MergePatch	3 PATCH/S/BB/ALP/BBALP0001/BANCAIR00067
2005/04/08 10:13:47	SGROOT	SGROOT		Match	MergePatch	2 PATCH/S/BB/ALP/BBALP0001/BANCAIR00066
2005/04/08 10:12:51	PRODUKTIE	SIMONDEV	BB-P-00012	Transfer	Transfer	Version 4.7 to PRODUKTIE
2005/04/08 10:06:59	PRODUKTIE	SIMONDEV	BB-P-00012	SW	Check-out	Patch PATCH/S/BB/ALP/BBALP0001/BB-P-0001
2005/04/08 10:06:59	PRODUKTIE	SIMONDEV	BB-P-00012	SW	Check-out	Create patchfile for task
2005/04/08 10:05:04	PRODUKTIE	SIMONDEV	BNC-00068	SW	Check-out	Patch PATCH/S/BB/ALP/BBALP0001/BNC-00068
2005/04/08 10:02:00	PRODUKTIE	SIMONDEV	BNC-00068	SW	Check-out	Patch PATCH/S/BB/ALP/BBALP0001/BNC-00068
2005/04/08 10:00:22	PRODUKTIE	SIMONDEV	BNC-00068	SW	Check-out	Patch PATCH/S/BB/ALP/BBALP0001/BNC-00068
2005/04/08 10:00:22	PRODUKTIE	SIMONDEV	BNC-00068	SW	Check-out	Create patchfile for task
2005/04/07 17:47:02	PRODUKTIE	SGROOT		CompResult	OK	4.7 NORMAL
2005/04/07 17:46:42	PRODUKTIE	SGROOT		SW	Compile	queue NORMAL
2005/04/07 17:45:14	PRODUKTIE	SGROOT		CompResult	OK	4.7 NORMAL
2005/04/07 17:42:59	PRODUKTIE	SGROOT		SW	Compile	queue NORMAL
2005/04/07 16:58:13	PRODUKTIE	SGROOT	BANCAIR00067	Transfer	Transfer	Version 4.7 to PRODUKTIE
2005/04/07 16:57:08	DEVELOP	SGROOT	BANCAIR00067	SW	Check-out	Patch PATCH/S/BB/ALP/BBALP0001/BANCAIR000
2005/04/07 16:57:08	DEVELOP	SGROOT	BANCAIR00067	SW	Check-out	Create patchfile for task
2005/04/07 15:10:08	SGROOT	SGROOT		Match	MergePatch	1 PATCH/S/BB/ALP/BBALP0001/BANCAIR00065
2005/04/07 13:18:30	PRODUKTIE	SGROOT	BANCAIR00066	Transfer	Transfer	Version 4.6 to PRODUKTIE
2005/04/07 13:18:15	PRODUKTIE	SGROOT	BANCAIR00065	Transfer	Transfer	Version 4.6 to PRODUKTIE
2005/04/07 13:14:53	DEVELOP	SGROOT	BANCAIR00065	SW	Check-out	Patch PATCH/S/BB/ALP/BBALP0001/BANCAIR000
2005/04/07 13:11:25	DEVELOP	SGROOT	BANCAIR00066	SW	Check-out	Patch PATCH/S/BB/ALP/BBALP0001/BANCAIR000
2005/04/07 13:11:25	DEVELOP	SGROOT	BANCAIR00066	SW	Check-out	Create patchfile for task

- Patch files that are merged permanently into the main source are not deleted from the repository. They are only delinked from the main source. Therefore, it is still possible to look to the contents of the original patch file, but that patch will not be merged anymore into the main source. Those "old" patch files can be found under the "PATCH" folder.



**Note:** This is a list of ALL patch files: the old delinked patch files, but also the current patch files.

### 21.3.3. Merge Method for Patch-Files

The default method to merge patch-files is by using the SYSTEM/PATCH.

SURE offers an alternative integrated merge method that does not use the SYSTEM/PATCH.

There are two problems with SYSTEM/PATCH:

- SYSTEM/PATCH is an external process and it must be started by SURE each time when patch files need to be merged. This gives much overhead and makes the SURE functions slowly. Especially, a task-transfer with many patchfile is slow.
- SYSTEM/PATCH removes all lines that start with a single \$ from the source. INCLUDE statements in ALGOL and JOB sources must start with a \$ and these lines are removed by SYSTEM/PATCH. For ALGOL sources this can be changed to a double \$ (for example, \$\$ INCLUDE "copyfile"), but for jobs this is not possible.

These problems do not occur with the SURE merge method.

All SURE functions with merging patch-files work identical with the SYSTEM/PATCH method or with the SURE method, except for the folder where the check on conflicts between patchfile is reported.

- In the case of SYSTEM/PATCH the conflict report must be opened using the following folder:  
From the Unisys MCP menu, select Printer Output, select SYSTEM/PATCH, click PATCH/CONFLICTS/<source>.
- In the case of the new SURE merge method the conflict report must be opened using the following folder:  
From the Unisys MCP menu, select Printer Output, select OBJECT/RIS/API/LFI/99, and click PATCH/CONFLICTS /<source>.

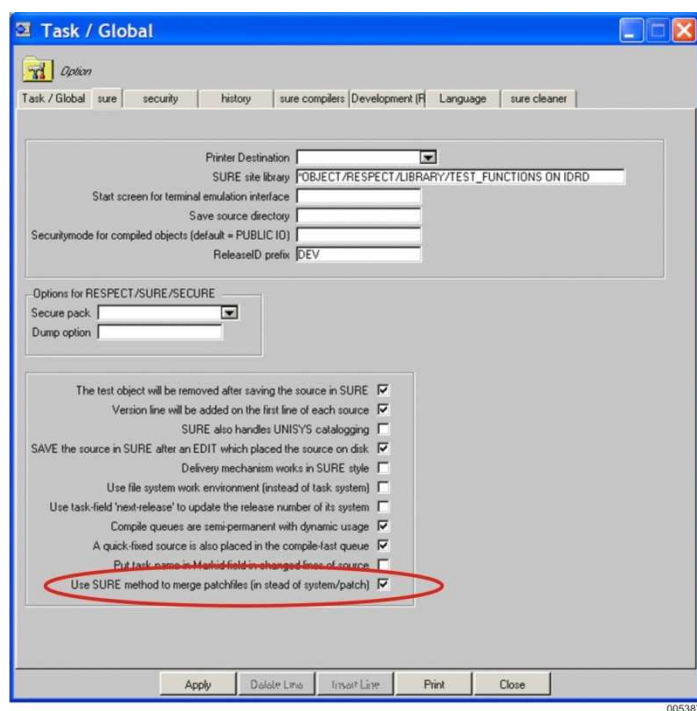
An example of the patch conflict report.

The following conflict(s) were found:

```
File: PATCH/SIMON/JOB1/ACCOUNTING0026
Overlap at lines:
      DISPLAY("HELLO WORLD12"); 00001500
With: PATCH/SIMON/JOB1/ACCOUNTING0025
```

The SURE merge method must be enabled using an option on the Global Options screen.

1. Right-click the Environment folder.
2. Click **Global Options**.
3. Select the **SURE** tab.



## 21.4. Manual Maintenance of Patch Files

SURE supports the automated mechanism for patch files where this automated mechanism is designed for concurrent source maintenance (see previous paragraphs). This mode has the following behavior:

At checked out time, a patch file is created by SURE and a symbol is presented to the developer containing all the patches required for the delivery date of the change. SURE also merges in stable patch files.

Therefore, using this mechanism, the patch deck is completely under control of SURE and also the number of patch files, their names and their existence are controlled by the SURE software.

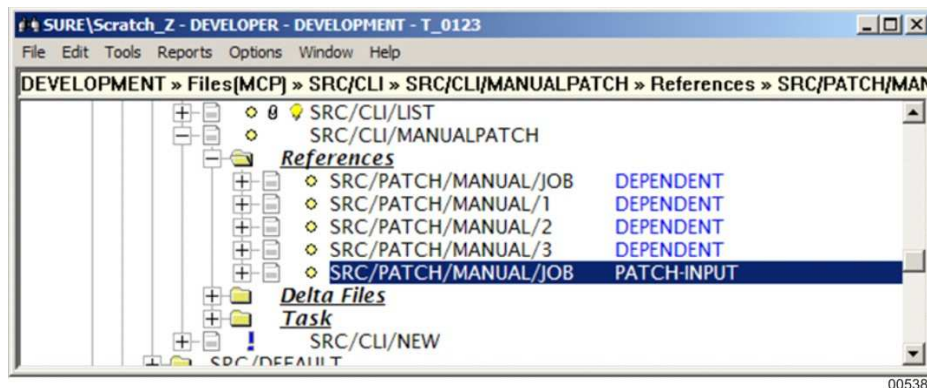
### Manual Mode

SURE also supports manual patch file mode. In this manual mode, the developer controls the patch deck and the dependent patch files on a symbol file. Because all input now consist of files and versioned relations, the normal SURE versioning applies. For that reason, the patch deck, the patch files, and the content of these files may differ in every environment.

This manual mode may be used for patch files on symbol releases provided by third party suppliers. It may also be used for concurrent source maintenance with detailed control by the developer.

## Patch Files

The following screen example shows a file with patch files and a patch input source.



The patch files are configured as dependent files; this places the files in the work-environment of the developer. This allows running system/patch with the patch deck if the sources are not checked out. Secondly, it enforces a compilation on the symbol if one of the dependent files is changed.

The patch input file is the CARD file for SYSTEM/PATCH. In principle, there may be a mismatch between the dependent files and the files listed in the patch input file. SURE does not validate this and it is the responsibility of the development team.

The patch input file is a regular SURE saved file. Note that the SURE file property "skip version line in source" must be set. Otherwise, SYSTEM/PATCH generates a syntax error.

When checking out patch files or the patch input source, SURE retrieves the files. It is up to the developer to start SYSTEM/PATCH and start an editor which merges the patch files like UED or NxEdit.

A PATCH-INPUT file can also be configured for a MULIT-OBJECT defined file. In this case, the dependent files are related to the symbol file. However, when the RESPECT/SURE/COMPILE creates the input for the multi object compilation, it uses the defined patch input. This allows creating multiple different objects for a single symbol with different patch input files. For example, if a country specific version is needed for a symbol.

Patch input and dependent files are defined from <file-properties>, Configuration button.



## 21.5. Batch Function to Repair Patch-File Relations

If a source is maintained using patch-files, then its patch-file relations must be equal in all environments. This batch function can be used in the case that the patch-file relations are not equal in all environments.

Run the batch function as follows:

```
Run RESPECT/REPOSITORY( "SYNCHRONIZE-PATCHFILES SOURCE=<source>  
FROM=<environment-from" )
```

where,

<Source>

is the name of the source.

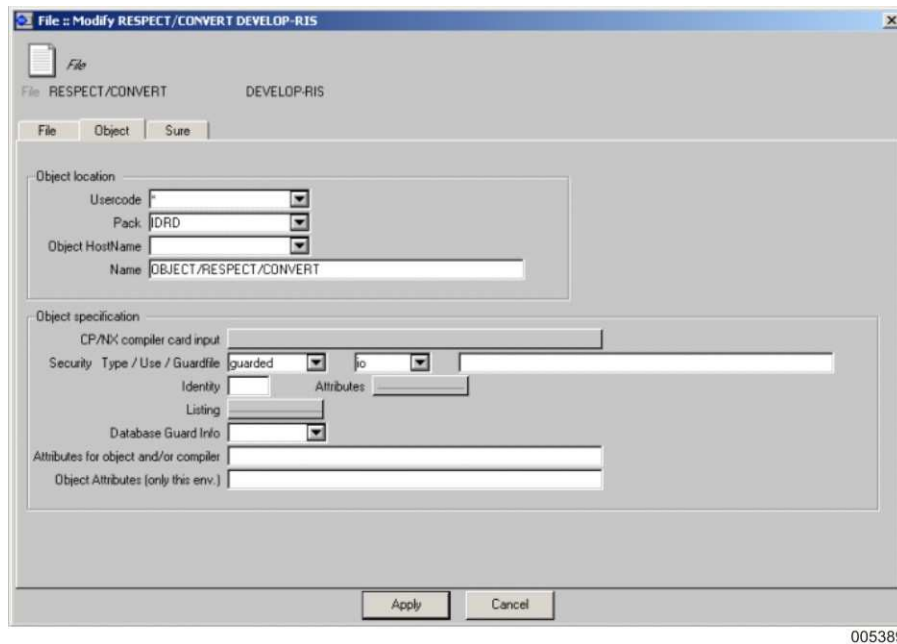
<Environment-from>

is the name of the environment where the patch-file relations are correct, and from where they must be copied to the other environments.



## Section 22

# Guard Files



SURE supports guard files in two ways:

- Define the security of an object

The security of an object can be defined on the File-Properties screen. The default security is PUBLIC IO. It is possible to define the security-use as GUARDED and to define a corresponding guard-file.

- Generate a guard file for a database

Field "Database Guard Info" makes it possible to define guard file information for a source.

The syntax is:

```
┌ <ReadWrite info> ───────────────────────────────────┐  
└ <Read-only info> ───────────────────────────────────┘
```

With SURE function GUARD <database> in RIS/MENU it is possible to define guard info for a database. The syntax for DATABASE guard info is:

```
— RW <ReadWrite info> — ; — R0 <Read-only info> ───────────────────┘
```

The RESPECT/SURE/GUARD batch program creates an output file containing the guard file information for a specific database.

A file inherits the database guard RW info if it is an update program [relation <filename>:UPDATE(<database>)]; otherwise, it inherits the database RO guard info.

If guard info is defined for a file, then this guard info overrides the database guard info for this specific file.

Example of the possible input for a database

```
RW , DMVERBS = ALL; RO , DMVERBS = (OPENINQUIRY, FIND)
```

Example of the possible input for a file

```
, DMVERBS = FIND
```

## 22.1. RESPECT/SURE/GUARD Overview

The RESPECT/SURE/GUARD ("<dbname>") program creates an input file for the Unisys Guard file utility. The program has to be started with the database name as a parameter, selects the guard info of that database, and all sources that reference that database.

The output file is called "SURE/GUARD/<dbname>".

## 22.2. SURE/GUARD/<dbname> Sample

A sample listing of the SURE/GUARD/<dbname> output file is detailed below:

```
10000010PROGRAM OBJECT/S/NP/SER/NPSER0001 ON IDRD = RW  
10000020, DMVERBS = ALL;  
10000030PROGRAM *OBJECT/LFI/DCM/ONLINE ON IDRD = RW  
10000040, DMVERBS = ALL;  
10000050PROGRAM *OBJECT/RIS/DUMP/TRANSACTIONS ON IDRD = R0  
10000060, DMVERBS = (OPENINQUIRY, FIND)
```

## Section 23

# Compilation and Object Files

The SURE software maintains source files. For ClearPath servers, it also maintains the object environment. Using SURE on ClearPath servers implicitly provides a mechanism that compiles sources after changes. This maintenance and synchronization of the object environments are a major function in the SURE package and provides a hundred percent automated and controlled mechanism.

It is obvious that the production-object-environment differs from a development-object-environment. The objects in the production object environment should only be updated using a formalized procedure. This chapter describes the compilation and transfer procedure of SURE.

The objects in a develop-object-environment can be updated through a formalized compilation evening batch (similar to the evening batch of the production environment), but also through manually started compilations of programmers.

The compilation procedure consists of the following steps:

- One of the attributes of a file is the object-location. The object-location consists of three parts: the object-usercode, the object-pack and the object-host. Together, they define the location where the compiled object of a file must be placed. The object-location is inherited from the default object-locations that are defined for each system and environment. It is possible to overrule the default object-location by a manually defined object-location. Refer for "23.3 Object Attributes" for more information.
- After a source is changed (generated, checked-in, or transferred), a relation is added which triggers the RESPECT/SURE/COMPILE program to compile the source. This relation is always created, also if the file has no object-location.
- The RESPECT/SURE/COMPILE program sorts out what to do with the file. If the file has an object-location then it is compiled. If it is a copy file then, every file using the copy file is compiled.
- The RESPECT/SURE/COMPILE program queues files to be transferred through a transfer relation. Depending on options, the application system integrity is guaranteed.
- The RESPECT/SURE/TRANSFER program creates a transfer tape each destination pack or it copies the files through BNA file transfer to the object-location.

## 23.1. Daily Batch

The target of the SURE batch is to process the impact of the changes that were made in the repository.

- The following changes in the repository have impact on the SURE batch:
- Modifications of sources and copyfiles by developers.
- Changes of options, task-attributes, or file-attributes.
- Changes because tasks are added in the repository.
- Transfer of tasks to higher environments.
- RIS users: modifications of RIS-entities that trigger generations of sources.

During a working day, the development teams are working on their application systems. New tasks are added, sources and copyfiles are modified, and tasks are transferred to the next environment, and so on. At the end of the day, the SURE batch recompiles all programs that are changed during that day. If a copyfile is changed, then all its referencing programs are changed. If a source is changed, then a deltafile is created (the differences between the previous version and the new version of the source) and loaded in SURE. If a source is changed then it is re-scanned for references to other files (such as datasets, copyfiles, and libraries). The next morning, when the development team starts a new working day, the repository is up-to-date again and all necessary compilations are done.

A standard SURE batch job is generated for each environment generated when SURE is installed. The name of such a standard batch job is WFL/<environment>/SURE.

### Example: WFL/DEVELOP-RIS/SURE

```
NEXT+      ....*....1....*....2....*....3....*....4....*....5....*....6....*.
10000100BEGIN JOB WFL/DEVELOP-RIS/SURE
10000200      (STRING RSPTITLE OPTIONAL DEFAULT="RESPECT/TITLES");
10000600STRING ENVIRONMENTNAME;
10001000
10001100ENVIRONMENTNAME:="DEVELOP-RIS";
10001300START (DEVELOP_RIS)WFL/DEVELOP-RIS/SURE ON IDRD(RSPTITLE);
10001400      STARTTIME = 22:00 ON + 1;
10001200
10001900$ INCLUDE SURE/WFLINCLUDE 20000000 TO 29999999
10002400RUN *OBJECT/RESPECT/SURE/COMPILE ON IDRD
10002700      FILE RESPECT = #RSPTITLE;
10002800      TASKVALUE =1      ;
10002900      TASKSTRING=ENVIRONMENTNAME;
10003000RUN *OBJECT/RESPECT/SURE/TRANSFER ON IDRD
10003100      ("NORMAL-MODE,NO-DUMPJOB,THROUGH-BNA");
10003200      FILE RESPECT = #RSPTITLE;
10003300      TASKSTRING=ENVIRONMENTNAME;
10003400RUN *OBJECT/RESPECT/SURE/MATCH ON IDRD("");
10003600      FILE RESPECT = #RSPTITLE;
10003700      TASKSTRING=ENVIRONMENTNAME;
```

```

10003800RUN *OBJECT/RESPECT/SURE/FIND ON IDRD(" ");
10004000 FILE RESPECT = #RSPTITLE;
10004100 TASKSTRING=ENVIRONMENTNAME;
10004200RUN *OBJECT/RESPECT/SURE/EXAMINE ON IDRD(" ");
10004400 FILE RESPECT = #RSPTITLE;
10004500 TASKSTRING=ENVIRONMENTNAME;
10004600RUN *OBJECT/RESPECT/PRINT ON IDRD("TASK-OVERVIEW SORT-BY-SYSTEM");
10004800 FILE RESPECT = #RSPTITLE;
10004900 TASKSTRING=ENVIRONMENTNAME;
10005000$ INCLUDE SURE/WFLINCLUDE 30000000 TO 39999999
10005100END JOB.
#DISPLAY COMPLETE

```

The exact content of the SURE batch job depends on various options which can be set through selecting the Environment-Properties and clicking the tab SURE-batch. The job is automatically regenerated when one or more of these options change.

It is possible to customize the SURE batch to your own needs. In that case, you must load the job in SURE or you must give it a dethoughting name, so that your customized job will not be overwritten when it is regenerated automatically.

The example job above puts itself automatically in the queue for the next evening. It is also possible to start the batch job on demand through the Compilation Interface dialog. In that case, a temporary batch job is generated and started.

### 23.1.1. Open Up SURE Batch Overviews

All overviews that are created by the SURE batch are saved in a directory and ready to be downloaded and opened through SUREforWindows.

#### Detailed Information

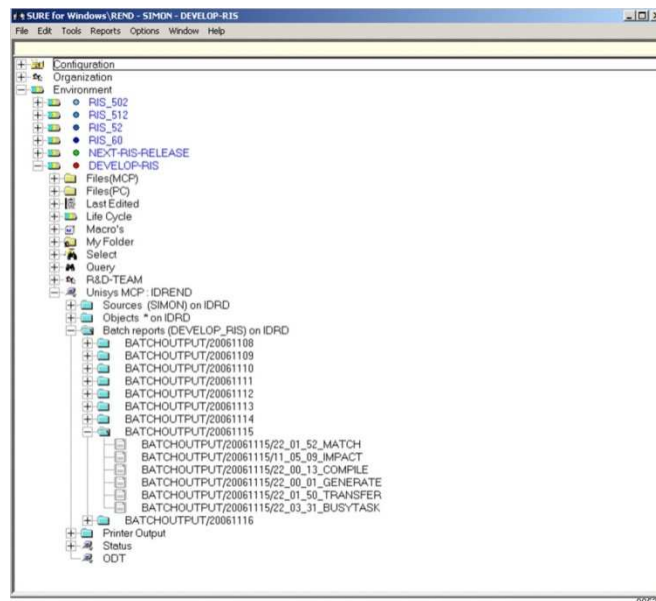
Many SURE batch programs create overviews. These overviews are not easy to track through the SUREforWindows browser, and therefore many users do not know about the existence of these listings. Therefore, the batch overviews are placed on the mainframe in directory:

(SURE-batch-usercode)BATCHOUTPUT\<yyyymmdd>\<name of overview>

The SURE browser contains a folder that is mapped to the MCP directory with batch output:

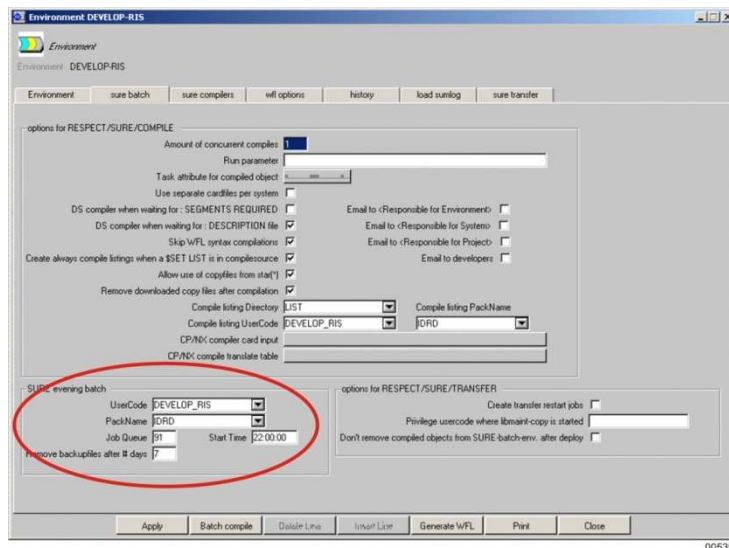
1. Open the desired folder.
2. Select **Unisys MCP**.
3. Select **Batch reports**.

## Compilation and Object Files



- Each SURE environment has its own directory with batch overviews.
- The date and time when the overview was created is part of the file name.
- Right-click a file name to download the overview and to load it into MS Word.

It is possible to define for each environment after how many days the batch overviews must be removed from disk. You can select the tab "SURE batch" from the Environment properties.



In this example the batch output is kept on disk in directory

(DEVELOP\_RIS)BATCHOUTPUT ON IDRD

and removed from disk after seven days.



### 23.1.2. SURE/WFLINCLUDE

SURE/WFLINCLUDE is an include file that can be used to customize the SURE jobs. SURE-jobs are generated by a number of SURE-programs. If a SURE job is generated and SURE/WFLINCLUDE is resident on disk, then two or three include statements for SURE/WFLINCLUDE are generated in the job. See the previous paragraph for an example.

In the case that a SURE-job is generate, the SURE-program searches first for a specific SURE/WFLINCLUDE/xxx file. If that specific include file is not found on disk, then the job will search for the default SURE/WFLINCLUDE (to keep compatibility with older installations). If the default include file is not found too, then no include file is used.

The following table gives a list of specific WFL include files.

<b>Program</b>	<b>Generates job</b>	<b>With WFLINCLUDE-suffix</b>
Evening batch	WFL/<environment>/SURE	Sure/Wflinclude/BATCH
Respect/Sure/CLEANER	Sure/CLEAN/<pack>	Sure/Wflinclude/CLEANER
Respect/Sure/SECURE	Sure/SECURE/SOURCES	Sure/Wflinclude/SECURE
Respect/Sure/STARTLOG	Sure/STARTLOG	Sure/Wflinclude/STARTLOG
Respect/Sure/TRANSFER	Sure/TRANSFER/<pack>	Sure/Wflinclude/TRANSFER
	Sure/DUMP/<pack>	Sure/Wflinclude/DUMP
	Sure/TRANSFER/REMOVE	Sure/Wflinclude/ REMOVE_OBJECTS

## 23.2.Compilation: SURE Commands

Various on-line commands are available to add files into a compile queue, delete files from a compile queue, to show the compile queue, and so on.

### 23.2.1. Compile

Files that are checked into the repository or transferred to a higher environment are automatically placed into the compile queue of that environment.

It is also possible to add programs manually into a compile queue using SURE command COMPILE. (Through file popup window or through file properties or configuration)

COMPILE	If the selected file name is linked to a user defined compile queue, then the file is added into that queue; otherwise, the file is added into the NORMAL queue.
COMPILE Normal	Add a file into the normal compile queue.
COMPILE Fast	Add a file into the compile fast queue.

COMPILE Specific <queue>	Add a file into a user defined compile queue, and link the file to that queue.
BLOCK (file/properties/configuration)	Block all compilations of this file through program RESPECT/SURE/COMPILE.
UNDO BLOCK (File/properties/configuration)	Remove the compile block. If the program had to be compiled for integrity reasons, then it is added into today's compile queue.

### 23.2.2. Compile <task>

Changing a file requires a task. The task is linked to the file, and by transferring the task, the files are promoted to the next environment.

Files that are checked into the repository or transferred to a higher environment are automatically placed into the compile queue of that environment.

It is also possible to "re-compile a task": all programs that are linked to the task are again added to the compile queue.

#### Function

1. Right-click **task-name**.
2. Select **Compile**.

### 23.2.3. UNDO Compile

This command deletes a file from a compile queue (file properties configuration).

UNDO COMPILE	If the file is linked to a user defined compile queue, then it is removed from that queue; otherwise, it is removed from the normal compile queue.
UNDO COMPILE Normal	Remove a file from the normal compile queue.
UNDO COMPILE Fast	Remove a file from the compile fast queue.
UNDO COMPILE <queue>	Remove a file from a user defined compile queue.

Other characteristics:

- It is not possible to use UNDO COMPILE if the file has impact relations, because this triggers an integrity chain. The file can be blocked for compilation through the COMPILE BLOCK command.
- If a file is removed from a user defined compile queue, and there is no other compile status left for that compile queue, then the file is automatically delinked from the queue.
- The file is removed from the compile-queue, remove relation COMPILE-STATUS(TO-COMPILE).

The file is removed from the transfer-queue: remove relation  
TRANSFER-STATUS(TO-TRANSFER).

### 23.2.4. Purge a Compile Queue

This command purges a total compile queue (through Tools/Compile Interface/Queue)

All programs are removed from the selected compile queue by deleting the following relations for that queue:

```
FILE-CONTROL | <file>:COMPILE-STATUS(TO-COMPILE)
FILE-CONTROL | <file>:COMPILE-STATUS(ACTIVE)
FILE-CONTROL | <file>:COMPILE-STATUS(COMPILING)
```

All programs are removed from the transfer queue that belongs to the selected compile queue by deleting the following relations:

```
FILE-CONTROL | <file>:TRANSFER-STATUS(TO-TRANSFER)
```

If the selected queue is not the compile-fast queue and not the recompile queue, then all impact relations and integrity chains are removed too as follows:

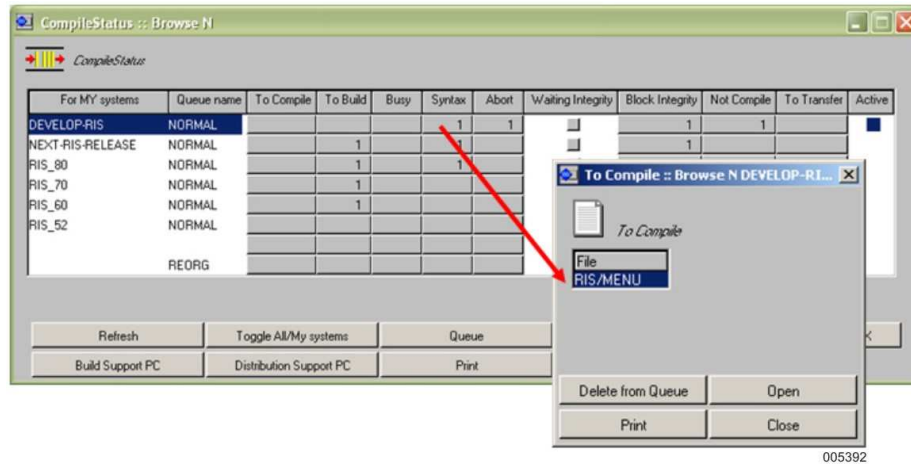
```
FILE-CONTROL | <file>:IMPACT(<impact-id>)
FILE-CONTROL | <file>:INTEGRITY(<impact-id>)
FILE-CONTROL | <file>:INTEGRITY-PROD(<impact-id>)
FILE-CONTROL | <file>:ABORT(INTEGRITY-PROD)
```

If the selected queue is a user defined compile queue then all COMPILED timestamps and TRANSFERRED timestamps are removed too as follows:

```
<queue> | <file>:COMPILE-STATUS(COMPILED)
<queue> | <file>:TRANSFER-STATUS(TRANSFERRED)
```

### 23.2.5. Compile Interface

Inquire the status of the batch compilations through the Compile Interface (Select Tools and click Compile Interface).



This command shows the compile queues. All available compile queues (NORMAL, FAST and user defined queues) of all environments are checked, and the following information is given for a queue:

<Environment>                      The name of the environment where the queue is active.

<Name of queue>                      NORMAL, FAST, <user's queue>

- Amount of ClearPath server files that have to be compiled.
- Amount of PC files that have to be built (on a build server).
- Amount of files with COMPILE-STATUS(ACTIVE) or COMPILE-STATUS(COMPILING).
- Amount of files with COMPILE-STATUS(SYNTAX)
- Amount of files with COMPILE-STATUS(ABORT)
- Are there impact relations for this queue? Yes or No.  
(These impact relations result in integrity chains).
- Are there integrity-prod relations for this queue? Yes or No.  
(These are waiting integrity chains.)
- Amount of files with relation ABORT(INTEGRITY-PROD).  
(These files block the waiting integrity chains).
- Amount of files that are blocked for compilation.  
(Through command COMPILE BLOCK).

- Amount of files that have relation TRANSFER-STATUS(TO-TRANSFER).  
(These objects are going to be transferred to the object-location).
- Is the user defined compile queue active?

You can click a number (the button with that number) to view programs with the corresponding compile status.

The following commands are supported:

- Refresh, which updates the status of the information.
- Multi Compile, which inserts programs in the compile queue.
- Queue, which activates queues and shows detailed queue information.
- Start Compile Job CP/NX, which allows the build process on the ClearPath server system.
- Build support PC, which initiates the build process on the PC.
- Distribution support, which initiates distribution support on the PC.

By default, only the sources that belong to a system of the user's system list are checked. The "Toggle all/my systems" button is used to select files of all systems.

When a compile is initiated from a SURE compile queue a batch process is started through WFL-deck. These generated online-WFL-decks always have the following name: WFL/<usercode-who-initiated-the-action>. This is only a temporary job. It is overwritten on disk with the next action that results in a batch process started through a WFL-deck. The internal name of the WFL-deck (after the BEGIN JOB statement) differs for each action. The Job title for these decks starts with "RISWFL/SURE/BATCH" and ends with nodes designating the environment, and SURE queue they are associated with.

This WFL deck is placed on disk under the usercode where the batch must run. For the normal compile-queue this batch-usercode is defined through the environment options. For user-defined compile queues the batch-usercode is entered on the screen when you started the batch. If a problem is experienced with this WFL decks they may be started and debugged from CANDE.

### **23.2.6. Activate a Compile Queue**

This command can be used to activate a user defined compile queue. Notice that a program can only be compiled through a user defined compile queue if that compile queue is active; otherwise, the file is compiled through the normal compile queue.

### **23.2.7. Deactivate a Compile Queue**

This command can be used to deactivate a user defined compile queue. If the selected queue is not empty, the files in that queue are moved to the normal compile queue before the selected queue is deactivated.

### 23.2.8. Check the Integrity Status

It can happen that a program is compiled by SURE (without syntax errors), but the object is not copied to the object environment because of integrity reasons. The program is blocked by another source that has compilation syntax errors. This command gives the names of the sources with syntax errors that block the file that was compiled correctly but not copied.

### 23.2.9. View the Compilation Listing

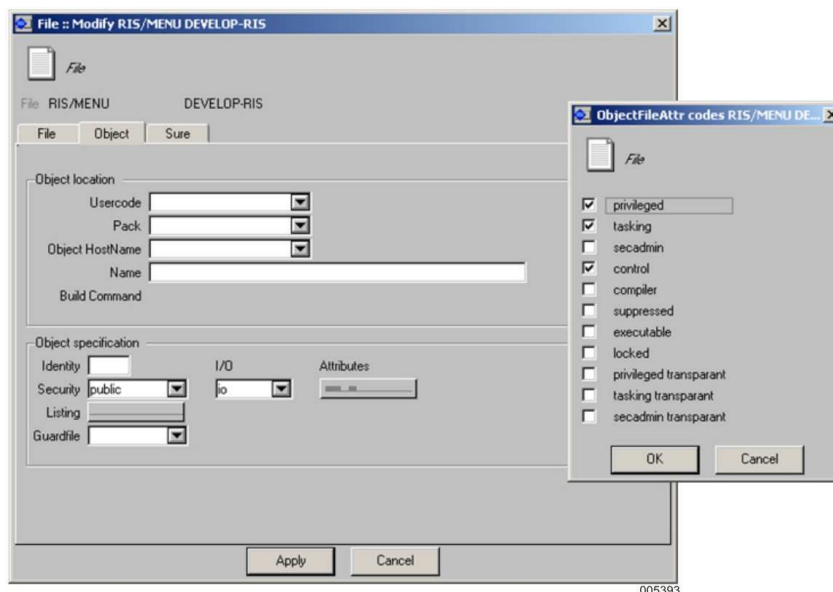
This command can be used to look into the last compile listing of the source. This command is only meaningful if the compile listing is copied to a backup directory by the SURE compile batch. Refer to “23.11 Compile Listings” for more information.

### 23.2.10. Set the Compile Listing Option

This command can be used to define extended compile listing options for a source. Refer to “23.11 Compile Listings” for more information.

## 23.3.Object Attributes

The “Object” sheet in the SURE Explorer allows modification of the additional object attributes. The attributes are updated through the File/Properties/Maintenance/Modify properties.



- The “guard filename” is added to the file attribute SECURITYGUARD.
- The “object identity” is a marker that can be added to an object. This marker is placed before each display by the program. This has the same effect as the Unisys MP IDENTITY command.

- Specifying on the particular fields can change the object attributes. For further information, refer to the *Unisys IO-SUBSYSTEM Manual* for the security options and the *Unisys ODT Manual* for the object options.

These object attributes are given to the objects after:

Compilations through program	OBJECT/RESPECT/SURE/COMPILE
Binding through programs	OBJECT/RESPECT/SURE/FINISH and OBJECT/RESPECT/SURE/COMPILE

The RIS/COMPLETE/OBJECT program can be started from the normal programmer's compile-job. This program also gives the desired options to the compiled object.

The "object usercode," "object-pack," and "object-host" define together the location for the object.

By default, the object-location is automatically inherited from the SYSTEM of the file. It is possible to define three different object-locations for each system on each environment.

### Example

Consider system SYS1 with the following object-locations:

Object files = (OBJ1) ON SYS1PACK

Bindobject = (OBJ2) ON SYS1PACK

Alternate = (OBJ3) ON SYS1PACK

The FILE-TYPE indicates which object-location must be inherited.

### Example

Consider file-type WFL with inherit option "Inherit alternate-user/pack as object-user/pack."

A file of system SYS1 with file-type WFL gets (automatically) object-location (OBJ3) ON SYS1PACK.

- If you change the object-location of a system, then the new object-location is automatically linked to all files that INHERITED the old location.
- If you change the "inheritance-option" of a file-type, then the object-location of the files with that file-type is changed too.
- If you change the file-type or system of a file, then the object-location of that file is changed too, according to the new file-type/system.

The above means that the object-location of a file is always equal to the object-locations of the file-type/system. If you change the attributes of the file-type/system then the object-locations of all those files are changed. That is flexible and powerful.

However, this automatic object-location inheritance is turned off for a file if you define the object-location of that file manually. Because then you explicitly define another object-location and that will not be overwritten by an automatic inheritance.

## 23.4.Compile Options

The SURE compile option screen allows the task attribute "option" for the resulting object to be created. This task attribute contains indicators for the dump in case of fault termination.

It is possible to define additional task attributes for the resulting objects or for the started compiler. Refer to the Unisys manual "TASK ATTRIBUTES," chapter "Assigning Task Attributes through HANDLE ATTRIBUTES" for detailed information.

These attributes are transferred to all environments and compiled in the object on each of those environments. They are applicable for the current environment (in this case DEVELOP).

The following screen shows the file property screen:

005394



Object attributes can be defined at the following levels:

- Each application system
- Each project
- Each file-type
- Each file (for all environments)
- Each file (for a specific environment)

The defined attributes are passed to the compiler object in the following order: system, project, and file-type, file (all environments), or file (this environment). This means that an attribute that is defined at system level can be overruled by another value of that same attribute at project, file-type or file level.

If a task attribute is preceded with the word COMPILER then the compiler obtains that attribute value. Additional task attributes can be defined at system and project level (project dialog), file-type level (file-type dialog) and each file (modify file properties).

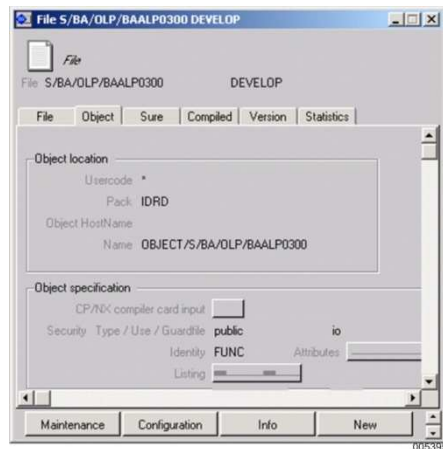
On the SURE compile option screen, other behavior for compilations can be defined such as location of copy files and the behavior if waiting conditions occur.

Together, these attributes allow definition of the following attributes for an object.

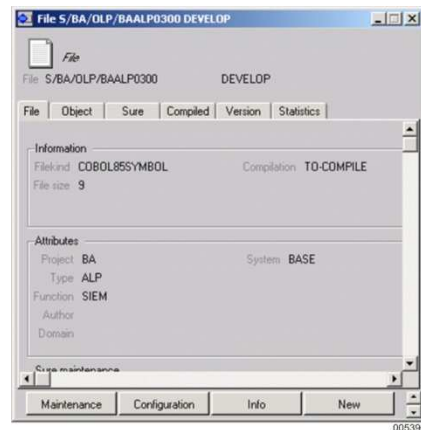
Attribute	Where
Security	Modify properties
Charge Code	<Compile options each project>
License Key	<Compile options each project>
Task attributes	<Compile options each project> or <each file-type>
Option	<Compiler option each system>
Identity	Modify properties

### Identity

Field "Identity" is used to address file attribute IDENTITY of the compiled object. If "FUNC" is entered in the identity-field, then the program function is used as identity for the object.



The program has identity FUNC.



The program has function SIEM.

The resulting object has identity SIEM:

```
LFILE OBJECT/S/BA/OLP/BAALP0300 :IDENTITY
#RUNNING 3958
#?
ON IDRD
(SGROOT) : DIRECTORY
. OBJECT : DIRECTORY
. . S : DIRECTORY
. . . BA : DIRECTORY
. . . . OLP : DIRECTORY
. . . . . BAALP0300 : COBOL85CODE IDENTITY="SIEM"
#
```

### 23.4.1. The ReleaseID of the Compiled Object

SURE puts information in the releaseid of the compiled objects, so that it is easier to determine how an object is created. The following information is placed in the releaseid of an object:

Prefix	The default prefix is "SURE": This default prefix can be customized through option "ReleaseID prefix" ( Select Global Options and click the tab SURE).
Environment	The name of the SURE environment from which the object is compiled.
File version	The file version of the source that was used for the compilation.
Descriptionfile version	The releaseid's of all the database descriptionfiles that were used at the compilation.

Compiler version                   The compile-version from the releaseid of the compiler.

Consider the following compilers with their release-id:

```
*SYSTEM/COBOL85 RELEASEID="IC COBOL85-048.1A.23 [48..."
*SYSTEM/COBOL74 RELEASEID="SSR 48.1 [48.150.000] (48..."
*SYSTEM/DMALGOL RELEASEID="SSR 48.1 [48.150.000] (48..."
```

The compiler version of the cobol85 compiler is "COBOL85-048.1A.23".

The compiler version of the cobol74 compiler is "48.1".

The compiler version of the dmalgol compiler is "48.1".

In general, the compiler-version is the first token of the compiler-releaseid not equal to "IC" or "SSR".

Example releaseid

```
lfile *object/ris/menu :releaseid
#RUNNING 6528
#?
ON IDRD
*OBJECT : DIRECTORY
. RIS : DIRECTORY
. . MENU : DMALGOLCODE
        RELEASEID="ITSforSURE: DEVELOP-RIS 133.1;
        Descriptionfile: INFDB_2_DMS491; Compiler: ALGOL-050.1A.3"
#
```

where

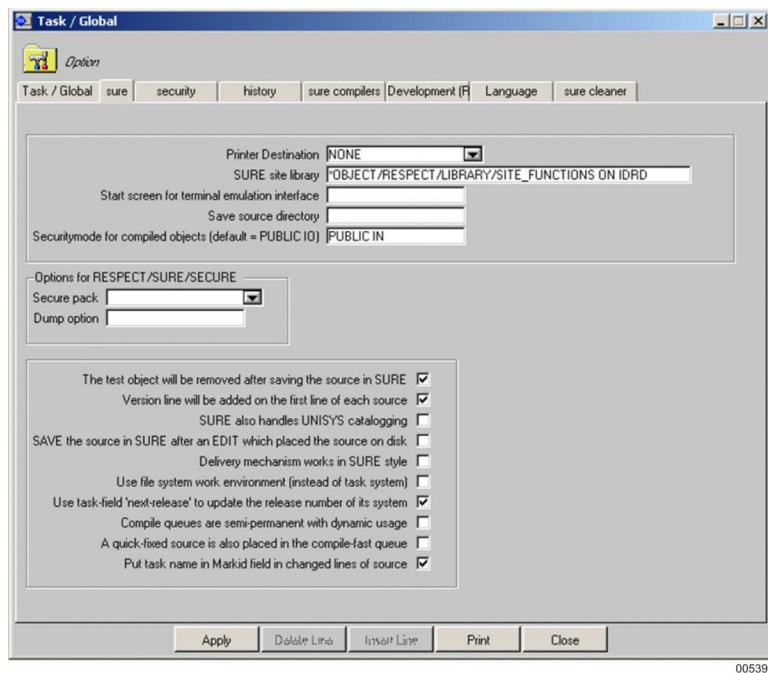
Prefix	=	ITSforSURE:
Environment	=	DEVELOP-RIS
File version	=	133.1
Descriptionfile version	=	Descriptionfile: INFDB_2_DMS491;
Compiler version	=	Compiler: ALGOL-050.1A.3

23.4.2. The Security of the Compiled Object

The default security mode for compiled objects is PUBLIC IO. It is possible to override this default with a site-specific value.

Example

- 1. Select **Global Options**.
- 2. Click the **Tab Sure**.



In this case, the default security mode for compiled objects is PUBLIC IN.

Secondly, it is possible to define a specific security mode for an individual file (file properties).

### 23.4.3. SURE FileVersion nr in Fileattribute "Version" of Objectcode

It is possible to put the SURE version number of a source into the VERSION file-attribute of the object code.

The RESPECT/SURE/COMPILE program scans the defined compiler card input for special tokens, which are then replaced by corresponding SURE-file-attributes.

The following tokens are recognized and replaced.

Token	Replace by
<RELEASENR>	The release number of the system of the file.
<ENVIRONMENT-NR>	The environment sequence number.
<FILE-CYCLE>	The file cycle number.
<FILE-PATCH>	The file patch number.

Example

Consider the following:

- A file with VersionNumber 15.6
- The file cycle number = 15; The file patch number = 6
- The system of the file has release number 4
- The repository has 3 environments: DEVELOP, TEST and PROD.
- The environment sequence numbers are: DEVELOP = 2; TEST = 1; PROD = 0

The following table shows examples of the syntax in the ClearPath server-compiler-card-input and the resulting VERSION attribute, when the RESPECT/SURE/COMPILE program runs for environment TEST.

ClearPath server compiler card input	VERSION attribute
\$ SET VERSION <RELEASENR>.<FILE-CYCLE>.<FILE-PATCH>	\$ SET VERSION 4.15.6
\$ SET VERSION <ENVIRONMENT-NR>.<FILE-CYCLE>.<FILE-PATCH>	\$ SET VERSION 1.15.6
\$ SET VERSION 0.<FILE-CYCLE>.<FILE-PATCH>	\$ SET VERSION 0.15.6

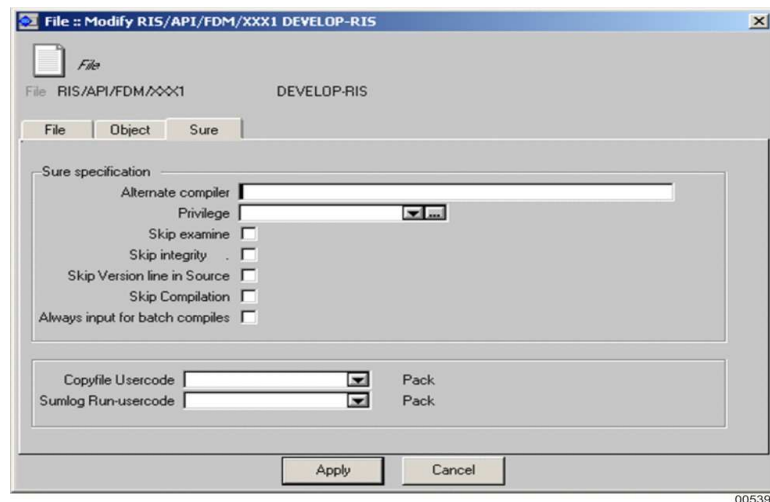
### 23.5.File as Fixed Input for the Compile Process

It is possible to indicate that a file has to be copied from SURE to disk by the RESPECT/SURE/COMPILE program during the initialization phase of the RESPECT/SURE/COMPILE.

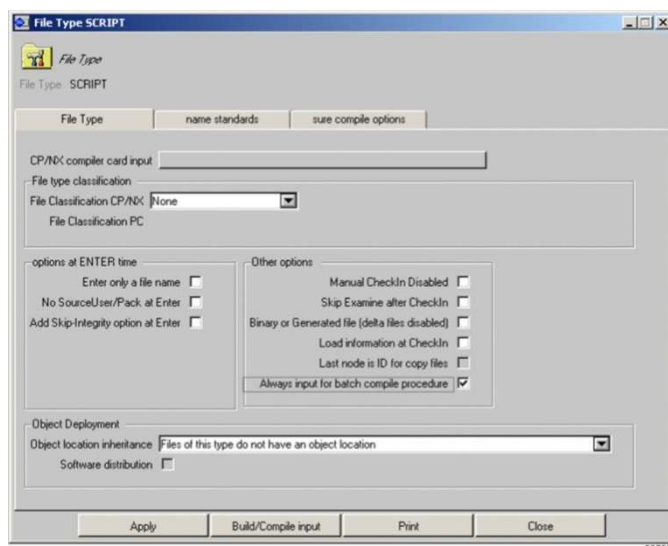
Some files are required during the SURE evening batch by generators or other external processes (such as XGEN, SDF, and DFDL) that are zipped by the RESPECT/SURE/COMPILE.

Option “Always input for batch compile procedure” is available for files and file-types. The RESPECT/SURE/COMPILE copies all files that have/inherit this option from the repository to disk (on the SURE batch location).

Set the option for an individual file.



Set the option for a file type. All files with this file-type inherit the option.



Files that have or inherit this option, are copied from SURE to the SURE-batch-location by the RESPECT/SURE/COMPILE. Old versions of these files are overwritten.

These files remain on disk when the RESPECT/SURE/COMPILE goes to end of task, so it is possible to use these files in other processes.

## 23.6.PRE and a POST Compile Job

SURE compiles all the (changed) programs using the RESPECT/SURE/COMPILE program that spawns off the appropriate compiler. It may be required to perform some special actions before or after the compilation when a specific file is compiled. For that reason, a PRE and a POST compile job can be defined that are invoked by the RESPECT/SURE/COMPILE at the appropriate time.

The PRE and POST compile job work essentially the same as a STARTJOB. They have the same parameters and they signal their completion by the RESPECT/SURE/FINISH program.

PRE and the POST compile jobs are defined through, Selecting File-properties and then clicking the Configuration button.

The following is an example pre-compile job.

```
00000100BEGIN JOB PRE/COMPILE/JOB(STRING VERSION,STRING TITLE);
00000150DISPLAY("PRE COMPILE JOB");
00000160WAIT(OK);
00000200RUN OBJECT/RESPECT/SURE/FINISH(TITLE);TASKSTRING=VERSION;
00000300END JOB.
```

## 23.7.Compiler Control Cards

The RESPECT/SURE/COMPILE starts the compiler with a primary and secondary input file (the "card- file" and the "tape-file"). The tape-file is file-equated to the source that has to be compiled and this tape-file is merged with the card-file.

The card-file contains dollar options that are used to control the compiler. A well-known dollar option is \$ OMIT, which informs the compiler to omit a number of records. The card-file must contain at least option \$ SET MERGE. The organization is free to declare its own dollar options in the card- file.

This compiler card input may be different for each environment, allowing to set different control cards for each environment.

The content of the card-file must be defined in the repository. That can be done at the following levels.

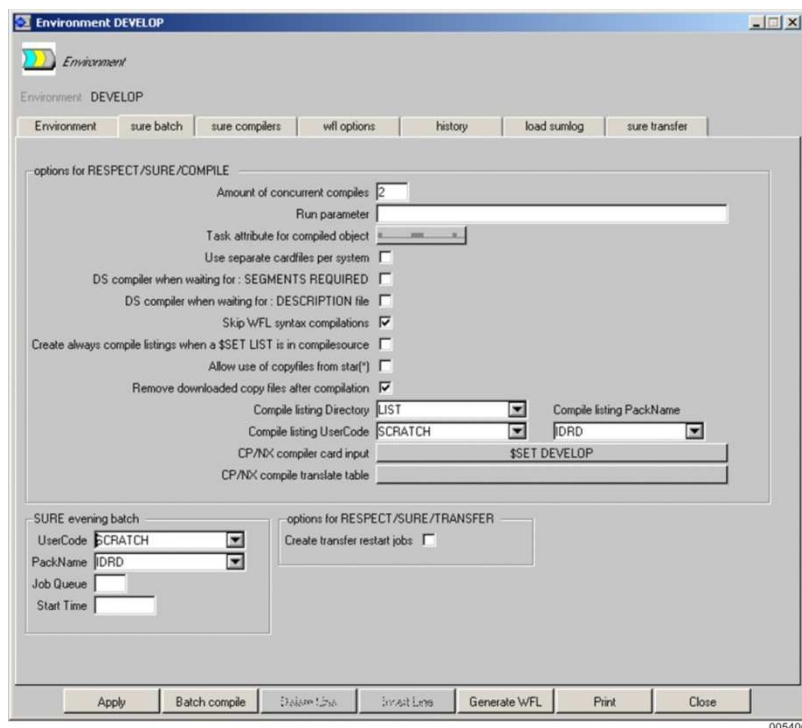
At level	Through function
Environment	<ol style="list-style-type: none"><li>1. Select Global Options.</li><li>2. Select tab, SURE batch.</li><li>3. Click the field, CP/NX compiler card input.</li></ol>
System	<ol style="list-style-type: none"><li>1. Select Project.</li><li>2. Select properties of the system.</li><li>3. Click the field, CP/NX compiler card input.</li></ol>
Project	<ol style="list-style-type: none"><li>1. Select Project properties of the project.</li><li>2. Click the field, CP/NX compiler card input.</li></ol>
File type	<ol style="list-style-type: none"><li>1. Select File type properties.</li><li>2. Click the field, CP/NX compiler card input.</li></ol>
Filekind	<ol style="list-style-type: none"><li>1. Select File type properties.</li><li>2. Click the field, CP/NX compiler card input.</li></ol> <p><b>Note:</b> This requires that the filekind is declared as file-type in SURE.</p>
File	<ol style="list-style-type: none"><li>1. Select File properties.</li><li>2. Select tab, Object .</li><li>3. Click the field, CP/NX compiler card input.</li></ol>

The dollar options that are declared on these levels are all merged by the RESPECT/SURE/COMPILE into one card-file in the above given order. The card-file is then offered to the compiler and removed when the compilation is finished.

It is not necessary to define \$ SET MERGE because that is done automatically by the RESPECT/SURE/COMPILE. This gives an extra benefit: if a site does not have any site-specific compiler dollar options, then it is not necessary to declare anything at all.



The **CP/NX compile translate table** field in the following figure is the first line of the ClearPath server compiler card input at environment level. Click the field (button), or click the button "Batch compile" to modify the card file input.



## 23.8.Compiler Translate Tables

This facility is only applicable for ClearPath server files.

Many sites have different names for databases, libraries, files, and so on in production and in develop. A source in development uses the develop names and if the source is released for production then the develop-names are translated to production names.

This concept is supported by SURE through "compile translation tables." The source loaded in SURE uses develop-names. If the source must be compiled for production, then that source is automatically translated through a predefined compile translate table before it is offered to the compiler.

A compiler replace table may be different for each environment. In most circumstances, the development environment does not use a compiler replace table, however, the production environment does.

The compiler replace table contains the target and destination strings for the translation. All odd lines in the table contain a target string, and all even lines contain a destination string. If a target string is found in the source, then the destination string that is specified on the next line of the table will replace this string.

If the target string and destination string are both terminated with a "@" sign, the replace will behave as a "replace literal." If these strings are not terminated with a "@"

sign, the replace is a “normal” replace, where only full names are replaced. In the normal mode, the characters just before and after the target string may not be alphanumeric characters or hyphens or underscores.

The content of the translate table must be defined in the repository. That can be done at the following levels.

At level	Through function
Environment	<ol style="list-style-type: none"><li>1. Select Global Options.</li><li>2. Select tab, SURE batch.</li><li>3. Click the field, CP/NX compile translate table.</li></ol>
System	<ol style="list-style-type: none"><li>1. Select Project properties of the system.</li><li>2. Click the field, CP/NX compile translate table.</li></ol>
Project	<ol style="list-style-type: none"><li>1. Select Project properties of the project.</li><li>2. Click the field, CP/NX compile translate table.</li></ol>

The translate actions that are declared on these levels are all merged by the RESPECT/SURE/COMPILE into one translate table in the above given order. The table is then offered to the compiler and removed when the compilation is finished.

### Example

```
File    F/1 System = SYSA
      100 BEGIN
      200   DATABASE TESTDB;
      300   OPEN UPDATE TESTDB;
      400 END
```

Compiling this program for development, without a compiler replace table results in using database DB.

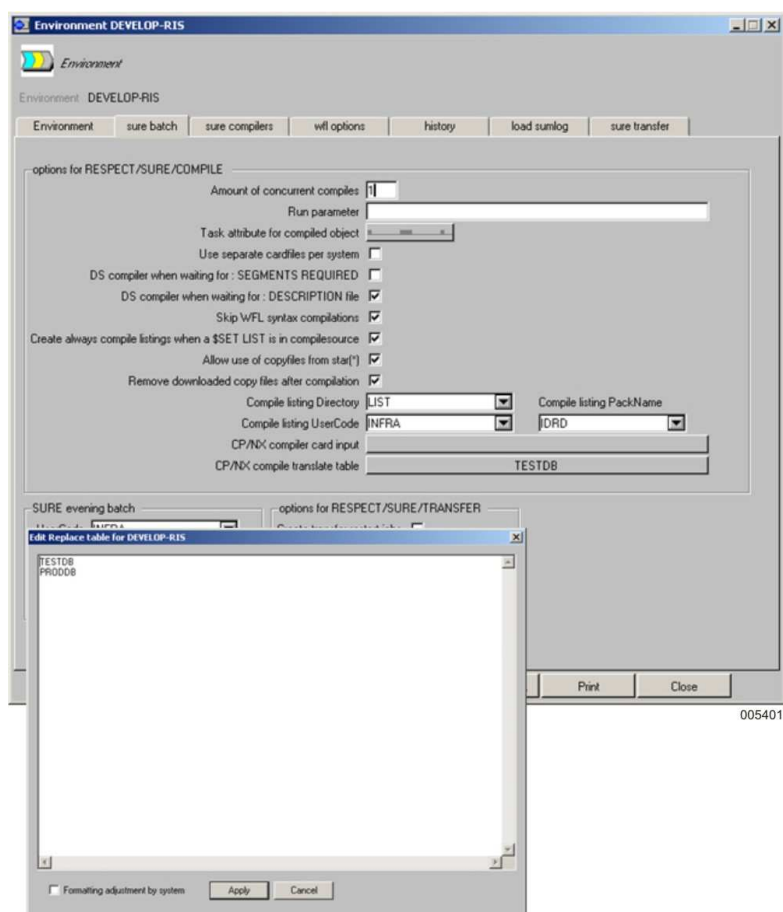
For production, the following compiler replace table is present.

```
TESTDB
PROddb
```

Compiling this program for production with the defined compiler replace table results in using database proddb.

The **CP/NX compile translate table** field in the following figure is the first line of the ClearPath server compile translate table at environment level. Click the field (button), or click the button "Batch compile" to modify the table.

Each line of text will move up one level. For example, the text on the first line will be replaced by the text on the second line, and the text on the third line will be replaced by the text on the fourth line.



## 23.9. Compiler Names

Through the SURE compiler options, dethoughted compiler names may be defined. Normally, the compilers are delivered by Unisys under a standard name, for example \*SYSTEM/COBOL74. If, the compiler is renamed, for example, \*SYSTEM/REL394/COBOL74 then you can inform SURE by entering that name in the COBOL74 compiler field.

This function also allows all compilers of the same kind to point to their superset, for example, use \*SYSTEM/DMALGOL for all ALGOL compilers.

It is possible to define compiler names each SURE environment. This creates the possibility to upgrade the application databases to a higher MCP level for each environment.

## 23.10. Object Names

By default, SURE uses the CANDE naming conventions for object names. An object is called OBJECT/<source name>.

A dethoughting object name can be defined each file on the SURE main screen. For these dethoughting object names the CANDE naming conventions are also active: The object is called OBJECT/<name> unless <name> starts with a "\$".

### Example

Consider source PROG/P1.

- If this source has a dethoughting object PRG/P1 the object is called OBJECT/PRG/P1.
- If this source has a dethoughting object \$OBJ/P1 then the object is called OBJ/P1.

In the case that CANDE naming conventions for object files are not adequate, it is also possible to define a library that determines the object-names. The global-options dialog contains a field "user library" and an option "User library supports object names." If this option is set then the user library determines all object names. (Unless a dethoughting object name is defined through the SURE main screen, see above). The site maintains the content of the user library.

The user library must contain two entry points:

- String procedure GIVE\_OBJECT\_NAME(S\_VERSION, S\_SRC, S\_PRJ);

```
VALUE    S_VERSION, S_SRC, S_PRJ;  
STRING   S_VERSION, S_SRC, S_PRJ;
```

- String procedure GIVE\_SOURCE\_NAME(S\_VERSION, S\_OBJ);

```
VALUE    S_VERSION, S_OBJ;  
STRING   S_VERSION, S_OBJ;
```

Procedure GIVE\_OBJECT\_NAME determines the object name of the source, using the environment name, the source name and the name of the project of the source. This procedure does not work for files with filekind DATA, SEQDATA, TESTDATA, and JOBSYMBOL. If objectnames need to be determined for these filekind then procedure GIVE\_OBJECT\_NAMES\_ALWAYS must be used.

### Example

```
String procedure GIVE_OBJECT_NAME(S_VERSION, S_SOURCE, S_PROJECT);
VALUE    S_VERSION, S_SOURCE, S_PROJECT;
STRING   S_VERSION, S_SOURCE, S_PROJECT;
BEGIN
    IF S_PROJECT NEQ "AAA" THEN
        GIVE_OBJECT_NAME:=S_PROJECT!!"/OBJ."!!S_SOURCE
    ELSE
        GIVE_OBJECT_NAME:="OBJECT/"!!S_SOURCE;
    END;
END;
```

Procedure GIVE\_SOURCE\_NAME determines the name of the source from the object name. This procedure is the opposite of the GIVE\_OBJECT\_NAME procedure. This procedure is used by the RESPECT/SURE/LOG program (read the runinfo of an object from the system log and link this info to the source) and by the RESPECT/SURE/EXAMINE program (create EXECUTE relations between the job and the source names of the objects that are executed by the job).

### Example

```
String procedure GIVE_SOURCE_NAME(S_VERSION, S_OBJECT);
VALUE    S_VERSION, S_OBJECT;
STRING   S_VERSION, S_OBJECT;
BEGIN
    IF S_PROJECT NEQ "AAA" THEN
        GIVE_SOURCE_NAME:=DROP(S_OBJECT, LENGTH(S_PROJECT)+5)
    ELSE
        GIVE_SOURCE_NAME:=DROP(S_OBJECT, 7);
    END;
END;
```

Notice that all above-mentioned procedures have to be available in the "object-name library." If one of the procedures is not defined or not correctly defined, then a warning is displayed each time a SURE program is started (on-line or batch).

## 23.11. Compile Listings

Various options are available to control the creation of compile listings:

- Parameter NO-LISTINGS of the RESPECT/SURE/COMPILE program  
If RESPECT/SURE/COMPILE is started with parameter NO-LISTINGS then there will never be any compile listing created. All other options are ignored in this case.
- SURE compile option <Create always compile listings>  
This option can be set through dialog <environment>/properties/tab sheet "SURE Batch."  
In this case, the dollar option \$ SET LIST is important. A compile listing is not created if the \$ SET LIST option is not applied to the compiler. This dollar option can be declared in the source or in the compiler card file.

If SURE compile option <Create always compile listings> is not set, then a compile listing is only made if the source is changed since the previous compilation by SURE and if \$ SET LIST is applied to the compiled.

If SURE compile option <Create always compile listings> is set, then compile listings are always created (but still depending on the \$ LIST option).

- Extended compile listing options

This extended option can be declared for the following levels:

- At system level (Project properties for system name)
- At project level (Project properties for system name)
- At file-type level (File-Type properties)
- At file level (File properties)

Extended compile listing options are passed to the compiler in the above-mentioned order. This means that an option that is defined at system level is overruled by another declaration at project level, file-type level and file level.

The RESPECT/SURE/COMPILE evaluates for each compilation the extended compile listing options. If an option is found, then it overrides the value of SURE compile option <Create always compile listings>.

An extended compile listing option consists of the following attributes:

- <Do not create a compile listing>
- <Create a compile listing>

If a compile listing has to be created:

- <Create it for each compilation by SURE>
- <Create it only when a file is changed>
- <Save the compile listing in a predefined backup directory>
- <Print the compile listing>
- <Suppress warnings in the compile listing or not> (ALGOL)
- <Suppress the contents of include-files in the listing or not> (COBOL)

It is possible to define a backup directory where the SURE compile listings are placed after compilation. (<Environment> properties in the SURE batch sheet). This backup directory contains only the newest compile listings. The name of the compile listing is shown through properties in the Compiled tab. Clicking this name opens the listing in WORD.

## 23.12. Compilation Error Files

Syntax errors that are detected by the compiler are placed in a compilation error file. The name of this error file is ERRORS/<source name>.

The error file is loaded in SURE and can be shown as follows:

- Click File popup menu then, select errors from SURE compile.
- Click File-properties then, select button "Info" from Errors.
- Click File-properties then, select tab "compiled" button in column "errorlist."

## 23.13. Multiple Object Files

SURE offers the possibility to create multiple objects for a source. The SURE Explorer provides access to the multiple object definition screen through the menus "file properties Configuration" or "Multiple Object".

This function allows the user to specify that more than one object can exist for one source, or an object is available under different names. On screen "Multiobject", it is possible to define other run usercodes, packs and host names than on the main SURE menu. In this case, only one compile takes place but the object is transferred to the destinations entered on the multiobject menu.

If a dethroughing system is defined and a compiler replace table is available for that system, for each system a compile is performed, after the source has been translated, and the resulting object is transferred to its defined destination.

If the object name satisfies the Unisys default OBJECT/<filename>, the object name field is left empty during the add function and a temporary object is compiled "<filename>/<obj-user-name>/<obj-pack-name>/<obj-host-name>". The transfer copies the object under its final name. All required relations for this function are linked to the object name instead of the source-name, as usually would be the case.

### Relations

The following relations are added or deleted:

```
FILE-CONTROL|<source-name>:MULTI-OBJECT(<object-name>)
FILE-CONTROL|<object-name>:OBJECT-USERCODE(<usercode>)
FILE-CONTROL|<object-name>:OBJECT-PACK(<packname>)
FILE-CONTROL|<object-name>:OBJECT-HOST(<hostname>)
FILE-CONTROL|<object-name>:SECURITY<securitytype securityuse>)
PROJECT|<system-name>:SYSTEM(<object-name>)
```

### 23.13.1. Allow to Add Multi-Object Definition in Batch Mode

It is now possible to define multi-object definitions in batch mode through an input file.

```
RUN RESPECT/REPOSITORY("MULTI-OBJECT <input-filename>")
```

<input-filename> is the name of the data file that contains the multi-object definitions.

The record layout of the input file is:

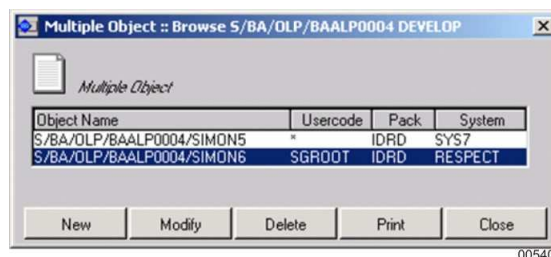
- Position 01-50 the source name.
- Position 51-100 Alternative source name (= the key for the name of the new object).
- Position 101-118 the system name.
- The source name and system name must be known in SURE.
- The object name may not be known in SURE as a source, or as an object of another source.

#### Example

Consider the following input file:

```
<pos 1 - 50 -----><pos 51 - 100 -----><pos 100 - 118 ----->
S/BA/OLP/BAALP0004          S/BA/OLP/BAALP0004/SIMON5  SYS7
S/BA/OLP/BAALP0004          S/BA/OLP/BAALP0004/SIMON6  RESPECT
```

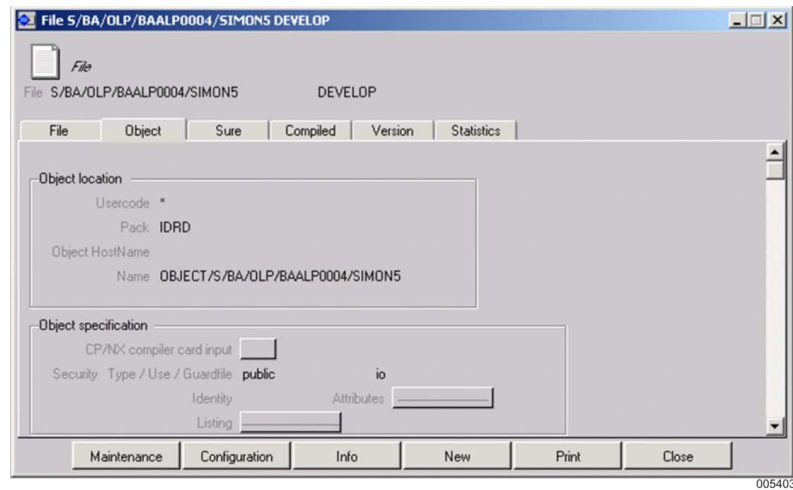
This results in the following multi-object definition.



Each alternative source name gets file-type MULTI-OBJECT. This file-type is used to inherit the default object-location (usercode/pack) from system the system. In this case, the object-location \* ON IDRD is inherited from system SYS7.

S/BA/OLP/BAALP0004/SIMON5 is the alternative source name. The corresponding object name is OBJECT/S/BA/OLP/BAALP0004/SIMON5 (see the next example screen).





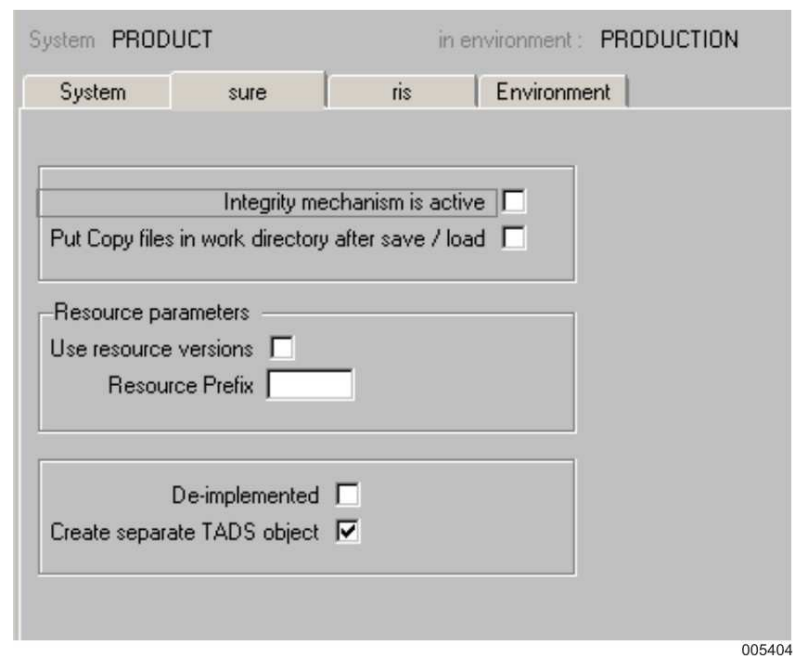
## 23.14. Additional Object with the TADS Option

On system level, in the SURE tab, you can specify that a program needs to be compiled again using the TADS compile option. Setting this option causes the program to be compiled twice, first with its normal object name and secondly with the name TADS/<object name>.

The additional tads-objects" get prefix TADS/= and must be deployed by the user through its internal jobs. The default deploy mechanism from SURE does not touch these files.

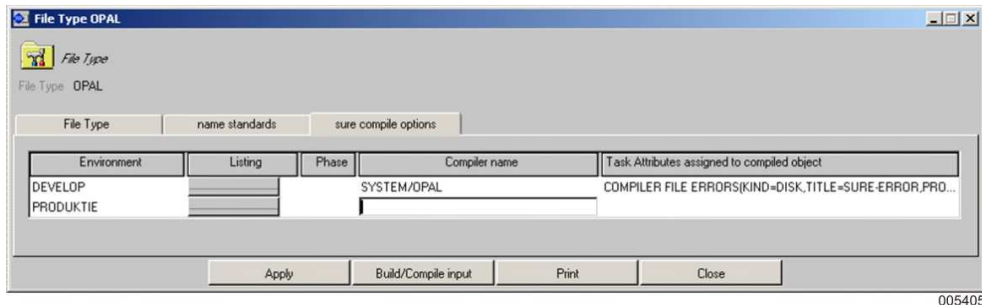
### Example

The following diagram shows the system properties.



## 23.15. Define your own Compile in SURE

### Example



This example shows a file-type OPAL with compiler name SYSTEM/OPAL.

### Technical details and considerations

#### Compiler files

By default, the following compiler input files are addressed.

#### Compiler Internal

File Name	Usage
CARD	The source-name: <Source-name>
TAPE/SOURCE	Not used
LINE	The compile listing: <Queue>/<prefix>/<source-name>
ERRORFILE	The syntax errors: ERRORS/<source-name>
CODE	The compiled object: <Object-name as defined in SURE>

Other internal compiler input files must be file-equated through the option “task attributes assigned to object.” If such a task attribute is preceded by the word compiler then that task attribute is assigned to the compiler instead of the object.

For example, consider the following attribute sentence:

```
PRIORITY = 55; COMPILER FILE ERRORS (KIND = DISK, TITLE = ERROR/FILE/ABC);
```

In this case, the compiled object gets always priority 55, and the compiler input file ERRORS is file-equated to ERRORS/FILE/ABC.

The above example is very static, because all programs with this file-type will have the same title for the error file, always ERRORS/FILE/ABC.

There are four generic tokens available that can be used in the attribute sentence and that are replaced by a correct file name.

SURE-CARD	The source name that must be compiled
SURE-LIST	The compile listing
SURE-CODE	The name of the compiled objects
SURE-ERROR	The name of the error file

### **Example**

Consider file PROG/123 and the following attribute sentence:

```
COMPILER FILE CARD (KIND = DISK, TITLE = SURE-CARD); COMPILER FILE ERRORS
(KIND = DISK, TITLE = SURE-ERROR, PROTECTED = SAVE, MYUSE = OUT); COMPILER
FILE LISTING (FILENAME = SURE-LIST, FAMILYNAME = BACKUPPACK, USERBACKUPNAME,
SECURITYTYPE = PUBLIC)
```

If the PROG/123 program is compiled, then the error file is called ERRORS/PROG/123 and the listing is called <queue>/<prefix>/PROG/123 ON BACKUPPACK.

### **ClearPath Libra Server compiler card input**

The SURE-compiler-card-input-definitions (which are used to define compiler-dollar-options for the system, projects, file-type or file), are not applicable in this mode and are not passed to your compiler (to avoid syntax errors on unexpected dollar cards).

### **Compile phase**

By default, the compilations are sorted by name: a source with name AAA is compiled first and a source with name ZZZ is compiled last. The compile phase can be used to overrule this default order.

- The default compile phase is 50. If no phase is entered then 50 is assumed.
- Binder-symbols have always compile phase 95.
- Copy files with a start-job have always compile phase 5.

You can link a compile-phase to a file-type. All sources with that file-type are then compiled (in alphabetical order) in the phase that you specified. For example, files with compile-phase = 40 are compiled before files with compile phase = 50.

### Compiler listing

If listing option "Make a compile listing" is enabled, then an extra compiler card "\$ SET LIST" is merged into the source at record 0. In that case, the records of the source must have the following layout.

Filekind	Sequence number	Record text
COBOL, COBOL74, COBOL85	1-6	7-72
JOB	1-80	83-90
Other filekinds	1-72	73-80

If listing option "Make a compile listing" is disabled, then no extra compiler card is merged into the source and then the layout of the record is not relevant.

### Compiler sheet array

The compiler is started with the usual sheet array as parameter. This array is initialized as follows:

SHEET[0].[47:1] = 1;SHEET[8]:=VALUE(LIBRARY); All other bits are 0.

## 23.16. Integrity Mechanism

The integrity mechanism is based on the following two statements:

- If a copy file is compiled, all files using that copy files must be transferred together.
- If a task is transferred, all files connected to that task must be transferred together.

It is obvious that these statements apply to acceptance and production environments and not necessarily to development environments. The manual initiated compilations in a development environment will constantly jeopardize the integrity of a development environment. For this reason, the integrity mechanism is an option defined at system level for each environment.

**Note:** *Integrity may prohibit the transfer of files. In development environments, it may easily occur that a single file blocks the transfer of many other files. Again, it is emphasized that setting this option for a development environment is in conflict with the usage of this environment. Therefore, it is advised resetting this option for any development environment.*

What are the effects of a system if the integrity mechanism is reset? Suppose there are two sources, a running program and a library that communicate through a record area, which is defined in a copy file. Changing the copy file requires recompilation of both programs. If compilation for one program fails due to syntax errors, and the other program is transferred, fatal errors may occur during program execution. The old

program uses the previous layout of the record area while the new program uses the newly changed layout.

If the integrity mechanism is active, none of these compiled objects would be deployed and the old versions of the objects would remain active.

At this point the question arises, how can a program to be taken into production result in syntax errors? In the Unisys MCP environment, the program source is a part of the total environment used at compilation. The program may use various copy files, database description files, ADDS record layouts and other environmental software. Therefore, not changing the program does not guarantee that it will be compiled syntax free.

An integrity chain is a group of files with the same `INTEGRITY(<ASSET>)` relation. The `<ASSET>` contains the changed resource that the files share. The `<ASSET>` can be a copy file, a task, or any other item. For the compile process, the `<ASSET>` is just the name of an object that groups the files together.

### **23.16.1. Integrity Introduction**

Integrity is an option in the SURE software that guarantees that a group of software modules is released simultaneously to the object environment.

The following functions are handled by the integrity mechanism:

- If a copy file is changed, all programs using the copy file are recompiled and the resulting objects are deployed simultaneously to the object environment.
- If a task is transferred to a higher environment, then all files that are linked to that task are recompiled and the resulting objects are deployed simultaneously to the object environment.

It is obvious that all objects in a run-time environment must be generated and compiled with the same input (= copy files and generator input). The following example illustrates this.

#### **Example**

Consider a system with two programs: a running program and a library that communicates through a record area, which is defined in a copy file. Changing the copy file requires recompilation of both programs. If compilation for one program fails due to syntax errors, and the other program is transferred, fatal errors may occur during program execution. The old program uses the previous layout of the record area while the new program uses the newly changed layout.

For RIS sites only, during the procedure, all changed RIS modules are regenerated and all changed programs are recompiled. The on-line impact analysis enforces that generations are started for all RIS-modules that use a changed RIS-identifier (for example, a changed format or a changed rule).

The RESPECT/SURE/COMPILE program enforces that compilations are started for all changed programs and for all programs that use a changed copy file.

If one of the generations or compilations fails, then none of the other objects (that are re-generated or recompiled for the same reason) is deployed to the run-time location.

The above-described procedure is called "Integrity Mechanism."

Notice that the integrity option does not influence the amount of compilations that are started in the evening procedure. If a RIS definition is changed then all RIS modules that use this definition are regenerated and recompiled. As soon as a copy file is changed, all programs that use this copy file are recompiled during the daily evening batch. The purpose of the integrity option is to ensure that all objects that are re-created for the same reason (changed definition or changed copy file) are released for the run-time environment as a group.

The integrity mechanism is implemented as an option, which can be defined for each system and environment. If the option is set for a system then the compilations of all programs that are linked to that system (or to one of the system's projects) are controlled by the integrity mechanism. It is possible to override the integrity option each individual file.

The integrity option is implemented each system because it can happen that the option must be set for system A and reset for system B at the same time.

The integrity option is implemented for each environment, because for some environments this option should be set (acceptance, production), while other environments do not need this option (development environment). The manual initiated compilations in a development environment will constantly jeopardize the integrity of the set of objects in that environment. This makes the integrity option useless at development level.

### 23.16.2. Integrity Reason

The purpose of the integrity option is to ensure that all objects that are re-created for the same reason are released simultaneously to the run-time environment.

There are several causes why an object has to be re-created:

- The source of the object is adapted. In this case, the task that is linked to the source describes the reason why the object must be recreated.
- A copy file of the source is adapted. In this case, the task that is linked to the copy file describes the reason why the object has to be recompiled.
- For RIS sites only, the source of the object is regenerated because of a changed RIS identifier. In this case, the task that is linked to the RIS identifier, describes the reason why the object has to be recreated.

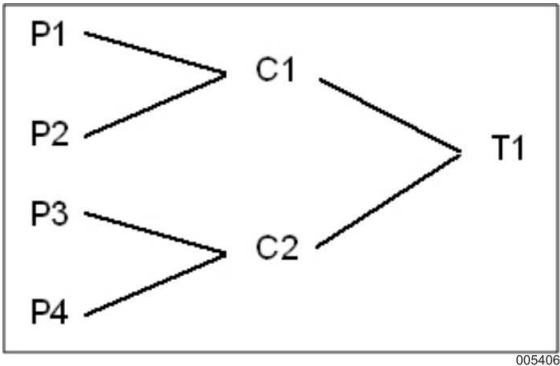
The reason why an object must be compiled is in all cases a task. This ensures that all adaptations that were made for the same task are released simultaneously to the run-time environment. This reason is linked to the program name through an integrity relation, and this is done during the initialization of RESPECT/SURE/COMPILE. All programs with the same integrity relation form together an integrity chain. All objects in the same integrity chain are deployed simultaneously to the run-time environment.

23.16.2.1. Integrity Chains Indicated by Tasks

Integrity chains are identified by tasks.

Example

Consider two copy files, which are adapted for the same task. Each copy file is in use by two programs: Programs P1 and P2 use copy file C1. Programs P3 and P4 use copy file C2. Copy files C1 and C2 are adapted due to task T1.



The following integrity chain is created.

		program	T1
P1	integrity T1	P1	*
P2	integrity T1	P2	*
P3	integrity T1	P3	*
P4	integrity T1	P4	*

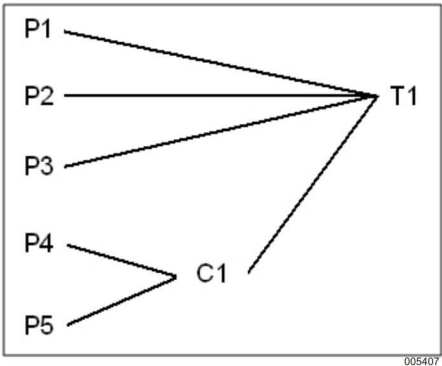
One integrity chain is created which enforces that all programs that have to be re-compiled because of adaptations due to task T1, are deployed simultaneously.

23.16.2.2. Integrity Chains and Adapted Copy Files

Programs that use a modified copy file are placed in an integrity chain that is indicated by the task that is linked to that copy file.

Example

The following picture shows a task T1 and four files (P1, P2, P3, and C1) that were adapted due to that task. One of these four files is a copy file that is used in two other programs (P4 and P5). In this case, all programs are placed in the same integrity chain, indicated by the task.



The following integrity chain is created.

		program	T1
P1	integrity T1	P1	*
P2	integrity T1	P2	*
P3	integrity T1	P3	*
P4	integrity T1	P4	*

One integrity chain is created which enforces that all programs that have to be re-compiled because of adaptations due to task T1, are deployed simultaneously.

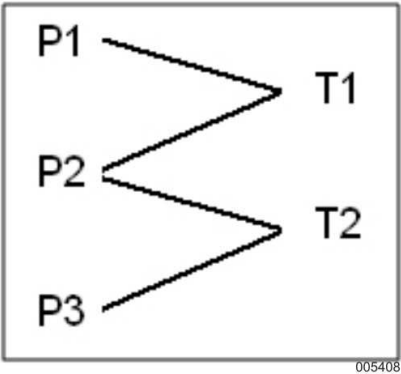


23.16.2.3. Overlapping Integrity Chains

Overlapping integrity chains are treated as one total integrity chain.

Example

Consider two tasks T1 and T2. Programs P1 and P2 are adapted because of task T1. Programs P2 and P3 are adapted because of task T2.



The following integrity chain is created.

		Program	T1	T2
P1	integrity T1	P1	*	
P2	integrity T1	P2	*	*
P2	integrity T2	P3		*
P2	integrity T2			

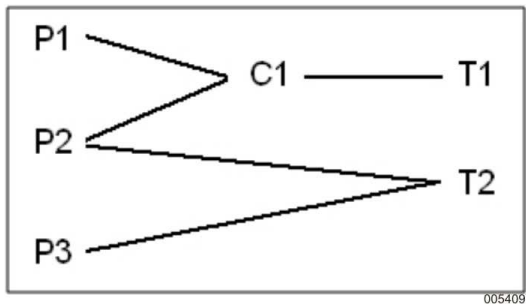
The second picture shows integrity chain T1 in the first column (all programs that are marked with "\*" are in the chain), and integrity chain T2 in the second column. P2 and P1 have to be released simultaneously (reason T1), and P2 and P3 have to be released simultaneously (reason T2). It is clear that the two chains are overlapping at program P2. Because of this overlap, the two chains are treated as one big chain to enforce that programs P1, P2 and P3 are released simultaneously.

23.16.2.4. Overlapping Integrity Chains and Copy Files

Overlapping integrity chains can also be created by adapted copy files.

Example

Consider copy file C1, which is adapted because of task T1. Copy file C1 is used by program P1 and P2. Programs P3 and P2 are adapted because of task T2.



This results in the following overlapping chains.

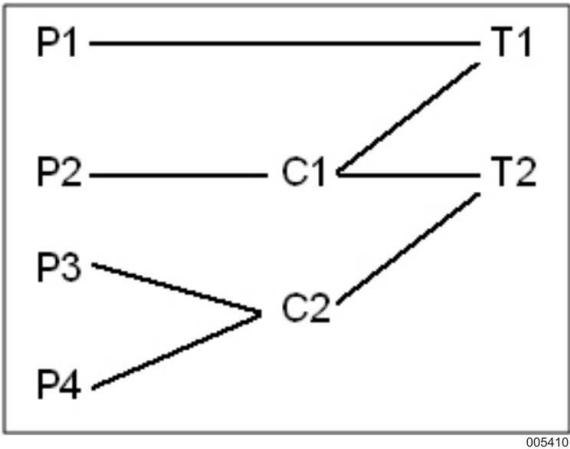
		program	T1	T2
P1	integrity T1	P1	*	
P2	integrity T1	P2	*	*
P2	integrity T2			
P3	integrity T2	P3		*

23.16.2.5. Multiple Integrity Overlaps in Copy Files

Overlapping integrity chains can also be created by adapted copy files.

Example

Consider copy file C1, which is adapted because of task T1. Copy file C1 is used by program P1 and P2. Programs P3 and P2 are adapted because of task T2.



This results in the following overlapping chains.

		program	T1	T2
P1	integrity T1	P1	*	
P2	integrity T1	P2	*	*
P2	integrity T2			*
P3	integrity T2	P3		*

23.16.3. Manually Defined Integrity Reason

The integrity mechanism is a very automatic procedure. If the integrity option is set, then all programs that have to be recompiled because of the same task are deployed simultaneously. Programs that have to be released simultaneously are placed in the same integrity chain and the creation of these integrity chains is based on the copy file relation between programs and copy files and on impact relations between files and tasks.

In some cases, it is necessary to create integrity chains manually. This is necessary if a global part of the application system is changed, but this modification is not known in the repository. For example, an adaptation of an ADDS-layout, an adaptation of an XGEN format, and so on.

With the RESPECT/REPOSITORY program it is possible to add a manual integrity trigger. This integrity trigger instructs the RESPECT/SURE/COMPILE program to create an integrity chain for files having a relation that is indicated by the trigger. All compiled objects that are in the same integrity chain are deployed together. If one compilation of the chain fails, then nothing will be deployed.

### Example

Consider two programs F/1 and F/2, both having relation `ADDS-FILE(MY/ADDS/FILE)`:

```
RUN OBJECT/RESPECT/REPOSITORY("SELECT CLASS ADDS-FILE ASSET MY/ADDS/FILE
FUNCTION
RELATE IMPACT MY/ADDS/FILE");TASKSTRING = "DEVELOP"
This selects all files with class = ADDS-FILE and asset = MY/ADDS/FILE and
creates the following manual integrity triggers:
FILE-CONTROL|F/1:IMPACT(MY/ADDS/FILE)
FILE-CONTROL|F/2:IMPACT(MY/ADDS/FILE)
```

The manual integrity trigger instructs the RESPECT/SURE/COMPILE program to create the following integrity chain:

```
FILE-CONTROL|F/1:INTEGRITY(MY/ADDS/FILE)
FILE-CONTROL|F/2:INTEGRITY(MY/ADDS/FILE)
```

The integrity triggers are removed automatically by the RESPECT/SURE/COMPILE program during the initialization.

## 23.16.4. Integrity Procedure

The purpose of the integrity option is to ensure that all objects in the run-time environment are created with the same set of global definitions. Therefore, the integrity mechanism can be split into the following parts:

- For RIS sites only, ensure that all generated sources are generated using the same set of RIS-definitions.
- Ensure that all objects are compiled using the source set of copy files.
- Through manual integrity triggers, it is possible to ensure that all objects are compiled using the same set of non-repository definitions (ADDS, XGEN).

For RIS sites only, adaptations to sources and RIS identifiers are linked to a task. The SURE software ensures that all modules are using the same adapted RIS-id are regenerated and recompiled.

The integrity option ensures that that the objects that were recompiled for the same reason are released simultaneously. The following paragraphs describe the total procedure in detail.

### 23.16.4.1. Integrity Adaptations in RIS Modules

**Note:** *This chapter contains only information for RIS-sites*

A source or RIS identifier can only be adapted if the user is connected to a current task.

Each adapted source or RIS identifier is automatically linked to the user's task that was current at the moment of adaptation. This link is made through a "problem" relation between the source/RIS-id and the task. If the task is transferred to a higher environment then all sources/RIS-id's that are connected to that task through a "problem" relation are transferred too.

A second action that is done when a source or RIS identifier is adapted is the automatic impact analysis. This impact analysis selects all RIS-modules that need to be regenerated because of this adaptation. The selected RIS modules are placed in the generate queue, and they are linked to the task through an "impact" relation. In the case that a source is adapted, then the 'impact' relation is also added to the source itself.

#### Example

Consider format F1, which is adapted because of task TASK01. Format F1 is used in LFI, LFI0001 and in FCM XX. A COBOL copy file is generated from format F1. The name of this copy file is CF1.

The adaptation results in the following relation.

Entity Owner:Class(Asset)	Reason
FORMAT-CONTROLIF1:PROBLEM(TASK01)	F1 is adapted because of task TASK01, and has to be transferred together with TASK01.
LFI-CONTROLI LFI0001: IMPACT(TASK01) LFI-CONTROLI LFI0001: GENERATE-STATUS(TO-GENERATE)	LFI0001 has to be regenerated because format F1 is part of this LFI and this format is adapted. The impact relation identifies why LFI0001 has to be regenerated.
FCM-CONTROLIXX:IMPACT(TASK01) FCM-CONTROLIXX:GENERATE-STATUS(TO-GENERATE)	FCM XX has to be regenerated because of a change of format F1 (due to task TASK01).
FILE-CONTROLICF1:IMPACT(TASK01) FORMAT-CONTROLIF1:GENERATE-STATUS(TO-GENERATE)	Copy file CF1 of format F1 has to be regenerated. The reason for this regeneration is indicated by the impact relation.

In this example the following modules are placed in the generate queue.

LFI	LFI0001	to generate the corresponding source
FCM	XX	to generate the corresponding source
Format	F1	to generate copy file CF1.

The reason for these regenerations is each time indicated through the `IMPACT` relation. The reason is: task TASK01.

The automatic impact analysis places RIS modules into the generation queue and links the reason why the RIS module has to be regenerated through an impact relation to that RIS module.

It is obvious that programs that are regenerated for the same reason also have to be compiled for that same reason.

This function is performed through the following steps:

- A RIS module is placed into the generation queue and the task that identifies the reason why the RIS module has to be regenerated is linked to that RIS module through an impact relation.

### Example

Entity Owner:Class (Asset)	Reason
LFI-CONTROL   LFI0001:GENERATE-STATUS(TO-GENERATE) LFI-CONTROL   LFI0001:IMPACT(TASK01)	LFI0001 has to be regenerated because of task TASK01.
FCM-CONTROL   XX:GENERATE-STATUS(TO-GENERATE) FCM-CONTROL   XX:IMPACT(TASK01)	FCM XX has to be regenerated because of task TASK01.
SEL-CONTROL   DTER/01:GENERATE-STATUS(TO-GENERATE) SEL-CONTROL   DTER/01:IMPACT(TASK02)	SEL DTER/01 has to be regenerated because of task TASK02.
SRT-CONTROL   SRT0001:GENERATE-STATUS(TO-GENERATE) SRT-CONTROL   SRT0001:IMPACT(TASK02)	SRT0001 has to be regenerated because of task TASK02.

- The correct RIS generator is started, and if the source of that RIS module is generated without errors and loaded in SURE, then all impact relations that were linked to the RIS module, are copied to the generated source and removed from the RIS module.

This means, if all RIS modules that were placed in the generation queue are generated successfully, then there are no impact relations anymore to RIS modules, but only to generated sources. If a generation failed, then the impact relations of that module are not moved to the source.

### Example

The generation of LFI0001, XX and DTER/01 were successful, but the generation of SRT0001 failed.

Entity/Owner:Class (Asset)	Reason
LFI-CONTROL LFI0001:GENERATE-STATUS (GENERATED) LFI-CONTROL SRC/LFI0001:COMPILE-STATUS (TO-COMPILE) LFI-CONTROL SRC/LFI0001:IMPACT (TASK01)	LFI0001 is generated-OK, the corresponding source has to be recompiled because of task TASK01.
FCM-CONTROL XX:GENERATE-STATUS (GENERATED) FCM-CONTROL SRC/XX:COMPILE-STATUS (TO-COMPILE) FCM-CONTROL SRC/XX:IMPACT (TASK01)	FCMXX is generated-OK, the corresponding source has to be recompiled because of task TASK01.
SEL-CONTROL DTER/01:GENERATE-STATUS (GENERATED) FILE-CONTROL SRC/DTER/01:COMPILE-STATUS (TO-COMPILE) FILE-CONTROL SRC/DTER/01:IMPACT (TASK02)	SEL DTER/01 is generated-OK, and the corresponding source has to be recompiled because of task TASK02.
SRT-CONTROL SRT0001:GENERATE-STATUS (SYNTAX) SRT-CONTROL SRT0001:IMPACT (TASK02)	The generation of SRT0001 failed the impact indication still exists.

- When all generations are finished, the RESPECT/SURE/COMPILE program is started. During the initialization of this program, the impact relations to files are changed to integrity relations. This ensures that all programs that had the same impact relation are recompiled and released simultaneously to the run-time environment. If an impact relation is still linked to a RIS-module (instead of its corresponding source), then the generation of this RIS-module failed, and the integrity chain is marked as "waiting integrity chain" to ensure that the correctly compiled objects are not transferred.

### Example

Entity/Owner:Class (Asset)	Reason
LFI-CONTROL LFI0001:GENERATE-STATUS (GENERATED) LFI-CONTROL SRC/LFI0001:COMPILE-STATUS (TO-COMPILE) LFI-CONTROL SRC/LFI0001:INTEGRITY (TASK01)	The impact relations of SRC/LFI0001 are changed to integrity relations.
FCM-CONTROL XX:GENERATE-STATUS (GENERATED) FCM-CONTROL SRC/XX:COMPILE-STATUS (TO-COMPILE) FCM-CONTROL SRC/XX:INTEGRITY (TASK01)	The impact relations of SRC/XX are changed to integrity relations.
SEL-CONTROL DTER/01:GENERATE-STATUS (GENERATED) FILE-CONTROL SRC/DTER/01:COMPILE-STATUS (TO-COMPILE) FILE-CONTROL SRC/DTER/01:INTEGRITY (TASK02)	The impact relations of SRC/XX are changed to integrity relations.
SRT-CONTROL SRT0001:GENERATE-STATUS (SYNTAX) SRT-CONTROL SRT0001:IMPACT (TASK02)	The generation of SRT0001 failed, so the impact indication still exists.

FILE-CONTROL   TASK02: INTEGRITY (TASK02)	Task02 is still linked through an impact relation to a RIS module (in this case SRT0001) and therefore this integrity chain TASK02 is marked as "waiting integrity chain."
FILE-CONTROL   TASK02: ABORT (INTEGRITY-PROD)	

Notice the two relations that block the integrity chain (last part of the diagram). Relation `ABORT (INTEGRITY-PROD)` ensures that the other objects that were compiled for reason "TASK02" are not transferred. Relation `TASK02: INTEGRITY (TASK02)` gives an indication that the reason of the waiting integrity chain is not a compilation failure but a generation failure.

- Correctly compiled objects are not transferred when they are part of a waiting integrity chain. Objects that are not part of a waiting integrity chain are placed into the SURE transfer queue, and their `INTEGRITY` relations are moved.

### Example

Entity Owner:Class (Asset)	Reason
LFI-CONTROL   LFI0001: GENERATE-STATUS (GENERATED) FILE-CONTROL   SRC/LFI0001: COMPILE-STATUS (COMPILED) FILE-CONTROL   SRC/LFI0001: TRANSFER-STATUS (TO-TRANSFER)	LFI0001 is generated-OK, the source is compiled-OK and not part of a waiting integrity chain, thus placed into the transfer queue.
FCM-CONTROL   XX: GENERATE-STATUS (GENERATED) FCM-CONTROL   SRC/XX: COMPILE-STATUS (COMPILED) FCM-CONTROL   SRC/XX: TRANSFER-STATUS (TO TRANSFER)	XX is generated-OK, the source is compiled-OK and not part of a waiting integrity chain, thus placed into the transfer queue.
SEL-CONTROL   DTER/01: GENERATE-STATUS (GENERATED) FILE-CONTROL   SRC/DTER/01: COMPILE-STATUS (COMPILED) FILE-CONTROL   SRC/DTER/01: INTEGRITY-PROD (TASK02)	File SRC/DTER/01 is compiled-OK, but still part of a waiting integrity chain (TASK02) so it is not placed in the transfer queue.
SRT-CONTROL   SRT0001: GENERATE-STATUS (SYNTAX) SRT-CONTROL   SRT0001: IMPACT (TASK02) FILE-CONTROL   TASK02: INTEGRITY-PROD (TASK02) FILE-CONTROL   TASK02: ABORT (INTEGRITY-PROD)	The generation of SRT0001 failed, the impact indication still exists.  These relations block the "waiting integrity chain."



Notice that all `INTEGRITY` relations of the waiting integrity chain are changed to `INTEGRITY-PROD`. This is done to prevent unnecessary compilations. The objects are already compiled and are waiting to be deployed. Files with an `INTEGRITY` relation are always recompiled.

- When the generation syntax errors are solved and the newly generated program is compiled, then the waiting integrity chain is released. All objects that were part of the waiting integrity chain are transferred together with the program that originally blocked the chain.

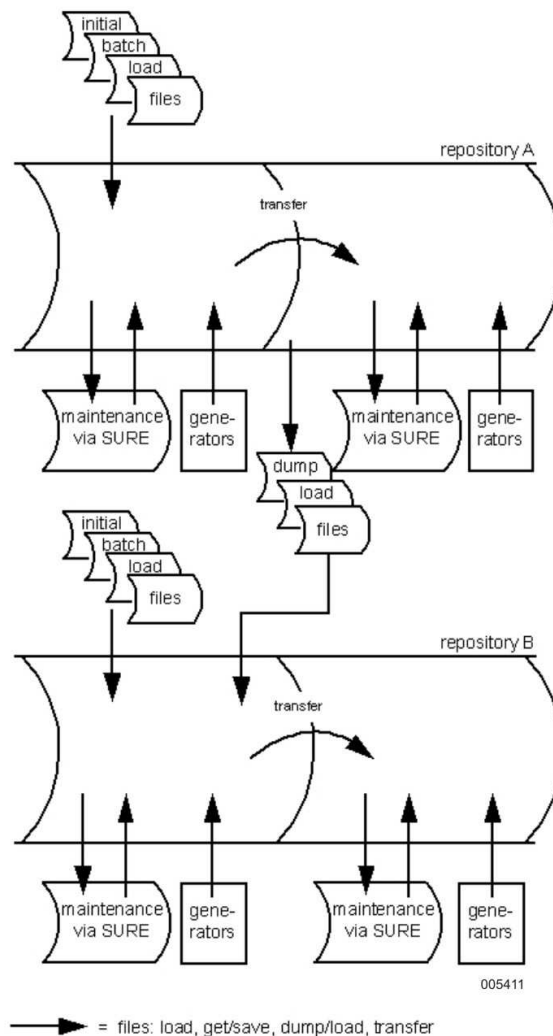
### Example

Entity Owner:Class (Asset)	Reason
SEL-CONTROL DTER/01:GENERATE-STATUS(GENERATED) FILE-CONTROL SRC/DTER/01:COMPILE-STATUS(COMPILED) FILE-CONTROL SRC/DTER/01:TRANSFER-STATUS(TO-TRANSFER)	SRC/LFI0001 is compiled-OK, and not part of the waiting integrity chain, thus placed into the transfer queue.
SRT-CONTROL SRT0001:GENERATE-STATUS(GENERATED) SRT-CONTROL SRC/SRT0001:COMPILE-STATUS(COMPILED) SRT-CONTROL SRC/SRT0001:TRANSFER-STATUS(TO-TRANSFER)	
The two relations that blocked the integrity chain are deleted.	

## 23.16.5. Integrity: Loading the Compile Queue

A new version of a file can be saved in an environment because of the following actions:

- Direct adaptation of that file using SURE functions CHECK OUT and CHECK IN.
- A generation of a RIS-module. The generated source is loaded in SURE.
- Initial load of files through the RESPECT/SURE/LOAD program.
- Load a group of files that were dumped from another repository.
- Transfer files from one environment of the responding to a higher environment of that repository.



A SURE compile queue is available for each environment in the repository. When a new version of a file is saved in a specific environment, then this file is placed automatically into the compile queue of that environment. Files can also be placed in the compile queue through SURE command COMPILE.

### 23.16.5.1.Integrity: Compilation Procedure

In the daily evening batch, the RESPECT/SURE/COMPILE program reads the compile queue and performs the appropriate actions for each source of the queue.

These actions are:

- If a source is used as a copy file, then all its master programs are placed into the compile queue with relation COMPILE-STATUS(TO-COMPILE).
- If the source has an object-usercode and object-pack then the correct compiler is started and the source is compiled.

Notice that a source is automatically placed into the compile queue of an environment, when this source changes in that environment. This change can be the result of:

- A manual check in through SURE.
- A regeneration of a source.
- A transfer of the source to the environment.
- An automatic load of the file.

Notice also that each changed source is placed into the compile queue. The OBJECT/RESPECT/SURE/COMPILE program reads this compile queue and performs the correct actions for each source.

The attributes object-pack and object-usercode determine the run-time environment of the source. If these attributes are not defined, there is no run-time environment for the source and then the OBJECT/RESPECT/SURE/COMPILE skips the source for compilation.

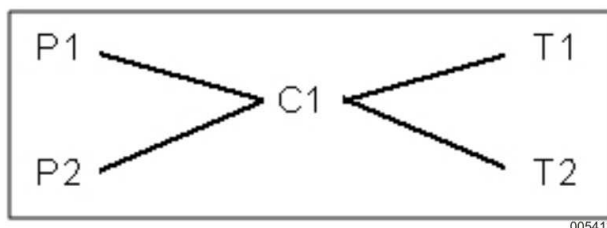
If the run-time environment is defined and the source is a seqdata file, this seqdata file is placed in the run-time environment.

During the initialization of the RESPECT/SURE/COMPILE, the integrity chains are checked. If the integrity option is set for a system and a changed copy file is part of that system, then an integrity chain is created between all programs that use that copy file. The reason of the integrity chain is determined by `IMPACT` impact relation of the copy file. (This relation determines the reason why the copy file was changed.) If the copy file has multiple `IMPACT` relations, then integrity chains between the master programs are created for each impact.

If a source is part of one or more drivers (through driver relations), then the `INTEGRITY` relations of that source is copied to each driver. This enforces that the drivers are part of the same integrity chains as their bind-objects. If the compilation of one of these bind-objects fails, then the driver will not be copied to its object-location.

### Example

Consider copy file C1 that is used by two programs P1 and P2. Copy file C1 is adapted because of task T1 and task T2.



The following relations represent this situation.

Entity Owner:Class (Asset)	Reason
FILE-CONTROL   P1 : COPY-FILE ( C1 )	Copy file C1 is used by programs P1 and P2. C1 is adapted because of tasks T1 and T2.
FILE-CONTROL   P2 : COPY-FILE ( C1 )	
FILE-CONTROL   C1 : IMPACT ( T1 )	
FILE-CONTROL   C1 : IMPACT ( T2 )	

The integrity chains are created during the initialization of the RESPECT/SURE/COMPILE. The reasons of these integrity chains are determined by the impact relations. A program that is linked to an integrity chain is automatically added into the compile queue, if this program has object relations.

### Example

Entity Owner:Class (Asset)	Reason
FILE-CONTROL   P1 : COPY-FILE ( C1 )	The reasons why copy file C1 was adapted (tasks T1 and T2) are linked through integrity relations to the files that use copy file C1 and to C1 itself.
FILE-CONTROL   P2 : COPY-FILE ( C1 )	
FILE-CONTROL   P1 : INTEGRITY ( T1 )	
FILE-CONTROL   P1 : INTEGRITY ( T2 )	
FILE-CONTROL   P2 : INTEGRITY ( T1 )	
FILE-CONTROL   P2 : INTEGRITY ( T2 )	
FILE-CONTROL   C1 : INTEGRITY ( T1 )	P1 and P2 are added to the compile queue.
FILE-CONTROL   C1 : INTEGRITY ( T2 )	
FILE-CONTROL   P1 : COMPILE-STATUS ( TO-COMPILE )	
FILE-CONTROL   P2 : COMPILE-STATUS ( TO-COMPILE )	

The result of the integrity check during the initialization of the RESPECT/SURE/COMPILE is:

- All changed programs are in the compile queue.
- All programs that are connected to one or more of these changed files through integrity chains are also placed in the compile queue.
- The integrity chains are indicating which files have to be compiled for the same reason.

When all compilations are done, a check is made for syntax errors in integrity chains. If a program is part of an integrity chain and the compilation of that program failed, then the other correctly compiled objects of that same integrity chain are not transferred to the run-time environment. Programs that have syntax errors block their integrity chains through an `ABORT ( INTEGRITY-PROD )` relation. The integrity connection between the files in a waiting integrity chain remains (to ensure that the objects are transferred simultaneously in a later stage), but the chain itself is changed from `INTEGRITY` to `INTEGRITY-PROD`. (Otherwise, the programs in the integrity chain are recompiled again during the next run of the RESPECT/SURE/COMPILE.)

### Example

Consider program P1 and P2. Both programs are placed into the compile queue and are connected through integrity chain T1.

Entity/Owner:Class (Asset)	Reason
FILE-CONTROL   P1 : COMPILE-STATUS ( TO-COMPILE )	P1 and P2 are placed in the compile queue and connected through integrity chain T1.
FILE-CONTROL   P2 : COMPILE-STATUS ( TO-COMPILE )	
FILE-CONTROL   P1 : INTEGRITY ( T1 )	
FILE-CONTROL   P2 : INTEGRITY ( T1 )	

The compilation of P1 fails and the compilation of P2 succeeds: The integrity chain is changed to a waiting chain.

Entity/Owner:Class (Asset)	Reason
FILE-CONTROL   P1 : COMPILE-STATUS ( SYNTAX )	The compilation of P1 failed, therefore the integrity chain is blocked(ABORT INT-PROD) and the chain itself is changed to waiting status.
FILE-CONTROL   P2 : COMPILE-STATUS ( COMPILED )	
FILE-CONTROL   P1 : ABORT ( INTEGRITY-PROD )	
FILE-CONTROL   P1 : INTEGRITY-PROD ( T1 )	
FILE-CONTROL   P2 : INTEGRITY-PROD ( T1 )	

The syntax error of P1 is solved and P1 is compiled again successfully. All programs that were waiting in the chain are placed into the transfer queue.

Entity/Owner:Class (Asset)	Reason
FILE-CONTROL   P1 : COMPILE-STATUS ( COMPILED )	Both programs are compiled-OK, and placed into the transfer queue.
FILE-CONTROL   P2 : COMPILE-STATUS ( COMPILED )	
FILE-CONTROL   P1 : TRANSFER-STATUS ( TO-TRANSFER )	
FILE-CONTROL   P2 : TRANSFER-STATUS ( TO-TRANSFER )	

Notice again that an integrity chain is created between all sources that use a changed copy file. A source can use a copy file directly (if a COPY statement is placed in the source) or indirectly (if the COPY statement is placed in one of the other include files of the source). The examine procedure ensures that all include files used by the source (directly or indirectly), are also linked to that source through copy file relations. These copy file relations are used to create an integrity chain. All sources that use the changed include-file are placed into the same integrity-chain, but only the sources that have object relations are placed into the compile queue.

### Example

Consider copy file C2 that is used by copy file C1, and C1 is used by program P1. Copy file C2 is adapted because of task T1.

P1 ————— C1 ————— C2 ————— T1

The following relations represent this situation.

Entity Owner:Class (Asset)	
FILE-CONTROL	P1: COPY-FILE ( C1 )
FILE-CONTROL	P1: COPY-FILE ( C2 )
FILE-CONTROL	C1: COPY-FILE ( C2 )
FILE-CONTROL	C2: IMPACT ( T1 )

During the initialization of the RESPECT/SURE/COMPILE the integrity chain is created using the impact and copy file relations. The programs that have to be compiled are placed into the compile queue.

Entity Owner:Class (Asset)	
FILE-CONTROL	P1: COPY-FILE ( C1 )
FILE-CONTROL	P1: COPY-FILE ( C2 )
FILE-CONTROL	C1: COPY-FILE ( C2 )
FILE-CONTROL	P1: INTEGRITY ( T1 )
FILE-CONTROL	C1: INTEGRITY ( T1 )
FILE-CONTROL	C2: INTEGRITY ( T1 )
FILE-CONTROL	P1: COMPILE-STATUS ( TO-COMPILE )

The RESPECT/SURE/COMPILE program continues after this initialization as in the previous example.

### 23.16.5.2. Integrity: Various Information

- It is possible to reset the integrity option each file. (field "Skip Integrity")
- If a copy file is changed, then all programs that use the copy file are recompiled during the evening batch. This is regardless of the integrity option. (This option only ensures that the compiled objects are transferred simultaneously.) It is possible to indicate each copy file that the master programs do not need to be recompiled after a change of that copy file. This can be done through option "skip integrity."
- A compiled program that is blocked by a waiting integrity chain is not recompiled in a next run of the RESPECT/SURE/COMPILE, unless the program itself is changed again or one of its copy files is changed again.
- The compilation overview shows a matrix of each waiting integrity chain plus an indication how to solve the problems that block the chain. Notice that there can be multiple waiting integrity chains, all independent of each other. It is possible to solve one waiting chain while the others remain in waiting status. Each waiting chain results in a separate matrix on the compilation overview.
- Function INTEGRITY (on the file-properties screen) can be used to investigate why a correctly compiled program is not copied to its object-location. If the program is part of a waiting integrity chain, then this chain is presented, together with the files that block the chain.

**Note:** The INTEGRITY chain may be blocked by a PC file, and that can only be resolved by executing the BUILD process on the PC.

### 23.16.6. Batch Function to Check the Resources Used for Compilation

The following batch function checks that programs are compiled with the latest version of each copy-book.

```
RUN RESPECT/REPOSITORY("CHECK-COMPILED [PUT]");TASKSTRING=<environment>
```

This function checks the following for each program:

- The compile-timestamp of the program is compared with the save/transfer-timestamp of each of its copy-books.
- If the timestamp of the copy-book is after the compile-timestamp of the program, the program name is displayed.
- Optionally (ff parameter PUT is used), the displayed program is placed into the compile queue.

In normal circumstances, the SURE-integrity mechanism enforces that all programs are compiled with the latest version of all copy files. If a copy file is changed, then all programs that use it are recompiled. However, in the following cases the integrity mechanism is overruled, and then this batch-function is useful:

- If a compile queue is manually purged. All integrity information is then removed as well.
- In the case that a copy file or program is marked with skip-integrity.
- In the case that the integrity mechanism is not activated.

### 23.17. RESPECT/SURE/COMPILE

Several SURE compile queues are available for each defined environment in the repository. Programs that are placed into one of these compile queues are compiled during the daily evening batch job that is started for the environment that belongs to that compile queue. The actual compilations are controlled by the RESPECT/SURE/COMPILE batch program.

The RESPECT/SURE/COMPILE has the following functions:

- Select all files that are placed in the compile queue.
- Ensure that all objects in the run-time environment are compiled with the same set of copy files and RIS definitions by defining integrity chains. (Refer to "23.16 Integrity Mechanism")
- Select all files that are placed in the compile-date queue for today's date.
- Check the versions of copy files that are already resident on disk.
- Compile all selected programs, and all files that are linked with these selected programs through integrity chains.
- Link pre-defined object attributes to the compiled objects (privileged program, control program, and so on).
- Checks that all programs that were linked through integrity chains are compiled ok.
- Place files that are compiled-ok and not blocked by a waiting integrity chain into the transfer-queue.



Run the RESPECT/SURE/COMPILE as follows:

```
RUN RESPECT/SURE/COMPILE("<input-parameter>");value = <taskvalue>
```

Input

Parameters :

NORMAL-Mode

FAST-MODE

QUEUE=<queue>

NO-STARTJOBS

DEBUG

NO-LISTINGS

USERBACKUPNAME

BANNER

PRINTDISPOSITION —<ID>

ALWAYS

TOTAL-BIND

NO-WFL-PRINT

DELIVER

SYSOVERVIEW

PRJOVERVIEW

NO-TADS

USE-UNKNOWN-VERSIONS

NO-CHECKED-OUT-FILES

NO-EXAMINE

FIRST-FILE-TYPE —<filetype>

The default mode = NORMAL-MODE

Task string : <environment>.

Task value : amount of simultaneous compilations, default 1.

File : SURE/COMPILE/<filekind> Compiler card file.

SURE/COMPILE/TABLE/<sys> Translate table

sources to be compiled

Compilers

XGEN/CONFIGS XGEN configuration file.

Output file : Compiled objects.

overview : An overview of the compiled files and the integrity status.

Example RUN RESPECT/SURE/COMPILE; = "DEVELOPMENT" ;

Where In the SURE evening batch.

Security MP +PU

MP +SECADMIN (the program adds predefined securities to the compiled objects)

NORMAL-MODE

Select files that are placed in the normal compile queue. The newest version of each file or copy file in this environment is used for the compilation, unless the file is linked to a specific compile queue. (Previous version is then used.)

FAST-MODE	Select files that are placed in the compile fast queue. The newest version of each file or copy file in this environment is used for the compilation, unless the file is linked to a specific compile queue. (Previous version is then used.)
QUEUE=<queue>	Select files that are placed in the specified compile queue. The newest version of each file or copy file is used for the compilation. A compile-queue is automatically available for each defined baseline.
NO-STARTJOBS	No start-jobs will be started (see later on in this chapter).
NO-LISTINGS	No compilation listings will be created.
USERBACKUPNAME	The compilation backup files are named as follows: COMPILOUT/<source>/LIST on <my family> The backup file of an XREF listing is called COMPILOUT/<source>/XREF on <my family>
PRINTDISPOSITION <id>	Identifies the print disposition of the compilation listings.
BANNER	Print a banner (with the source name) in front of each compilation listing.
DEBUG	Extra DEBUG info is printed on the compilation overview.
ALWAYS	This option can only be used for compilations in a development environment.  If this option is used, then the RESPECT/SURE/COMPILE will not go to end-of-task when all compilations are done, but he will check every minute if new files are placed in the compile-queue. If the compile queue is not empty then the RESPECT/SURE/COMPILE will resume compiling.  The integrity mechanism is ignored in this mode.  DRIVER relations are ignored in this mode. A start-job will be started for each START-JOB relation.  Compiled objects are copied to their object-locations by the RESPECT/SURE/COMPILE. It is not necessary to run the transfer program.
NO-TADS	Enforce that all programs are compiled without TADS.
TOTAL-BIND	Start-jobs can be started for a compiled source, or for the driver of that compiled source. The second situation results in a total bind. By default, a start-job is started for the compiled-source itself, if a START-JOB relation is linked to that source. If this parameter is used, and a start-job relation is linked to the driver, then the driver's start-job will be started.  This parameter is useful in combination with parameter ALWAYS. Refer to Section 24, "Binding of Programs," for more information.
PRJOVERVIEW	A compilation overview is made each application project. This overview reports (each project) the files that are compiled, blocked by integrity or added to the transfer queue.  The tasks that were the reasons for the compilations are also reported. This can be a handy overview for the application project manager.

SYSOVERVIEW	<p>A compilation overview is made each application system. This overview reports (each system) the files that are compiled, blocked by integrity or added to the transfer queue.</p> <p>The tasks that were the reasons for the compilations are also reported. This can be a handy overview for the application system manager.</p>
DELIVER	<p>If this parameter is used then a data file is created for each integrity chain (with objects) that released for the run-time environment. Overlapping integrity chains are treated as one big chain. The name of the data file is TRF/PGM/&lt;task&gt;, and all file names that are part of the chain are written in that data file. The data file can be input for site-specific procedures.</p>
USE-UNKNOWN-VERSIONS	<p>If a copy file is already resident on disk, then this copy file will only be used for the compilations if the RELEASEID of that copy file is correct (because then it is the correct version). Invalid versions of copy files are changed to directory "UNKNOWN/". With this option, it is possible to use unknown versions of copy files.</p>
NO-CHECKED-OUT-FILES	<p>Programs are normally copied from the repository to disk before the compilation is started. If a source is in maintenance by a developer, then the checked-out version is used (only in the development environment). If this option is set, then the file is always copied from SURE and the checked-out version won't be used.</p>
NO-EXAMINE	<p>By default, the files and copy files are changed to directory EXAMINESOURCE/= after the compilations. This is not done if this option is used.</p>
FIRST-FILE-TYPE <filetype>	<p>This option enforces that the files with &lt;filetype&gt; are compiled first and then the other files. This option makes it possible to use the result of the first compilations (or the result of directly started start-jobs of these first compilations) in start-jobs that are triggered by the other compilations</p>
<taskvalue>	<p>The task value determines the amount of compilations that are executed simultaneously. This amount can be adapted during the run of the RESPECT/SURE/COMPILE through ODT command &lt;mixno&gt; HI &lt;amount&gt;.</p>

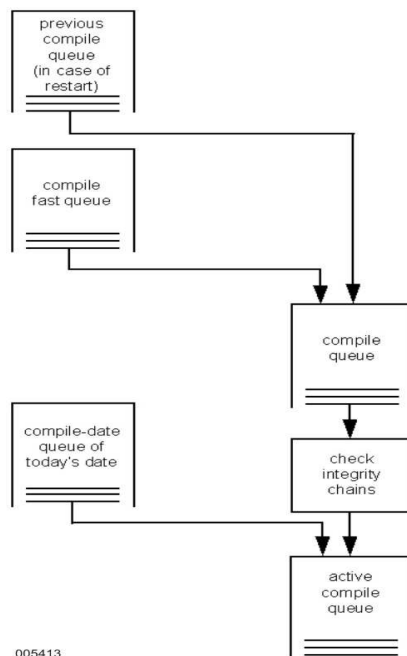
Notice that the RESPECT/SURE/COMPILE can handle various compile-queues (see input parameters), but the actions that are performed by the program are equal for each queue.

The following actions are performed during the initialization of the RESPECT/SURE/COMPILE:

- Check if the previous run of the RESPECT/SURE/COMPILE for the specified compile queue was completed correctly. If the compile queue of the previous run was not completed, then the remaining programs will be compiled during this run.
- Integrity chains are created. Files that are part of an integrity chain are placed into the compile queue too. Refer to "23.16. Integrity Mechanism" for more information.

- If the RESPECT/SURE/COMPILE is started in normal mode, then the files that were placed in the compile-date queue for today's date, are added to the normal compile queue.
- All copy files that are used by the programs that are going to be compiled are copied from SURE to disk. If necessary, these copy files are translated according to the available translate tables.
- The compile-queue is changed to a compile-active queue. All files with relation <FileName>:COMPILE-STATUS(TO-COMPILE) gets relation <FileName>:COMPILE-STATUS(ACTIVE) and the TO-COMPILE relation are removed.

It is not possible to change files during the initialization of the RESPECT/SURE/COMPILE program. Generators that need to save a newly generated source wait until the initialization of the RESPECT/SURE/COMPILE is finished. The Check-In command is blocked during the initialization of the RESPECT/SURE/COMPILE.



After the initialization of the RESPECT/SURE/COMPILE, the actual compilations are processed. Each file that is placed into the active compile queue will be compiled and the following actions will be performed:

- Copy the source from SURE to disk, and translate this source, if necessary, according to the visible translate tables. Refer to the chapter "Compiler Translate Tables in Section 23" for more information.
- Select the correct input card file with compiler control records. Refer to "23.7 Compiler Control Cards" for more information.

- Initialize the correct compiler that has to compile this source. Compiler attributes are passed to the compiler in the following order:
  - Pass compiler attributes that are defined for the program's system.
  - Pass compiler attributes that are defined for the program's project.
  - Pass compiler attributes that are defined for the program's file type.
  - Pass compiler attributes that are defined for the program itself.

Refer to "23.3 Object Attributes" for more information.

- Start the compilation using the retrieved source and copy files.
- Store the result of the compilation with a compile status in the repository. (COMPILED, SYNTAX, or ABORT).
- Add the compilation to the log of the file, together with the reason why the compilation was done.
- Make a backup file of the compilation listing. (optional)
- Create an XREF listing of the source. (optional)
- Change the compiled source to another directory where it can be used as input for the examine procedure. (Directory EXAMINESOURCE/<FileName>.)
- Start start-jobs that are linked to the source. (Start-jobs can be defined for binding purposes, and so on)

The integrity chains are verified when all compilations are executed and finished. If the compilation of a program was not successful, and the program is part of an integrity chain, then that integrity chain is changed to a "waiting integrity chain" and all programs in this waiting integrity chain are not deployed. If the compilation of a program was OK, and the program is not part of a waiting integrity chain, then this program is placed into the transfer queue. The RESPECT/SURE/TRANSFER program reads this transfer queue and generates a job which copies the compiled objects to their run-time environment.

### Start-job mechanism

If the RESPECT/SURE/COMPILE has compiled a program with a STARTJOB relation, then the job that is mentioned in the relation is started with the source name as a parameter.

### Example

Consider the EXAMPLE/PROGRAM program with a start-job relation to job WFL/STARTJOB:

```
FILE | EXAMPLE/PROGRAM:START-JOB(WFL/STARTJOB)
```

If the compilation of EXAMPLE/PROGRAM was successful, then the following start-job trigger is added:

```
FILE-CONTROL | EXAMPLE/PROGRAM:JOB-STATUS(TO-START)
```

At the end of all compilations the queued start-jobs are started with the environment and FileName as parameters:

```
START WFL/STARTJOB( "DEVELOPMENT, EXAMPLE/PROGRAM" )
```

The start-job relation is changed from TO-START to JOB-BUSY:

```
Add:      FILE-CONTROL | EXAMPLE/PROGRAM:JOB-STATUS( JOB-BUSY )
Delete:    FILE-CONTROL | EXAMPLE/PROGRAM:JOB-STATUS( TO-START )
```

The OBJECT/RESPECT/SURE/COMPILE waits until all JOB-BUSY relations are deleted. The started job can change this relation by running the RESPECT/SURE/FINISH. The task value of the RESPECT/SURE/FINISH determines the result of the start-job:

Task value	Action
0	Normal termination: the compile-date (if available) is extracted from the object and added to SURE. COMPILE-STATUS( COMPILED ) is added.
1	Abort. COMPILE-STATUS( ABORT ) is added.
2	Syntax. COMPILE-STATUS( SYNTAX ) is added.

The environment that is passed by the RESPECT/SURE/COMPILE to the start-job must be used as a task string for the RESPECT/SURE/FINISH program to ensure that this program runs for the same environment as the compile program.

### Example

The started job changes the start-job relation from JOB-BUSY to JOB-READY by running the RESPECT/SURE/FINISH program:

```
RUN OBJECT/RESPECT/SURE/FINISH( "EXAMPLE/PROGRAM" ); TASKSTRING="<environment>" .
```

This program updates the following relations:

```
Add:      FILE-CONTROL | EXAMPLE/PROGRAM:JOB-STATUS( JOB-READY )
Delete:    FILE-CONTROL | EXAMPLE/PROGRAM:JOB-STATUS( JOB-BUSY )
```

The RESPECT/SURE/COMPILE proceeds when all JOB-BUSY indications are gone.

### Example of a Start Job

```

00001000BEGIN JOB WFL/TESTRUN(STRING VERSION, STRING FILENAME);
00002000  TASK T;
00003000  T(STATUS=NEVERUSED);
00004000  RUN OBJECT/#FILENAME [T];
00004100  IF T IS COMPLETEDOK THEN BEGIN
00004110      IF T IS COMPILEDOK THEN BEGIN
00004120          RUN OBJECT/RESPECT/SURE/FINISH(TITLE);TASKSTRING=VERSION;
00004130      END ELSE BEGIN
00004140          RUN
00004150          OBJECT/RESPECT/SURE/FINISH(TITLE);VALUE=2;TASKSTRING=VERSION;
00004160      END;
00004170  END ELSE BEGIN
00004180      RUN OBJECT/RESPECT/SURE/FINISH(TITLE);VALUE=1;TASKSTRING=VERSION;
00004190  END;
00010000?END JOB.

```

All files and copy files that were downloaded are changed to the examine directory (EXAMINESOURCE/=). The RESPECT/SURE/EXAMINE program uses the (translated) files in this directory for examine purposes.

The amount of compilations that are started by the RESPECT/SURE/COMPILE is returned through the task value to the WFL by which the program was started.

## 23.17.1.Compilation Overview

The RESPECT/SURE/COMPILE creates a compilation overview with the following information:

- The active run options for the RESPECT/SURE/COMPILE.
- Which programs are going to be compiled.
- Which copy files are used.
- The result of each compilation.
- The waiting integrity chains.
- The objects that are going to be transferred.
- The amount of compilations.

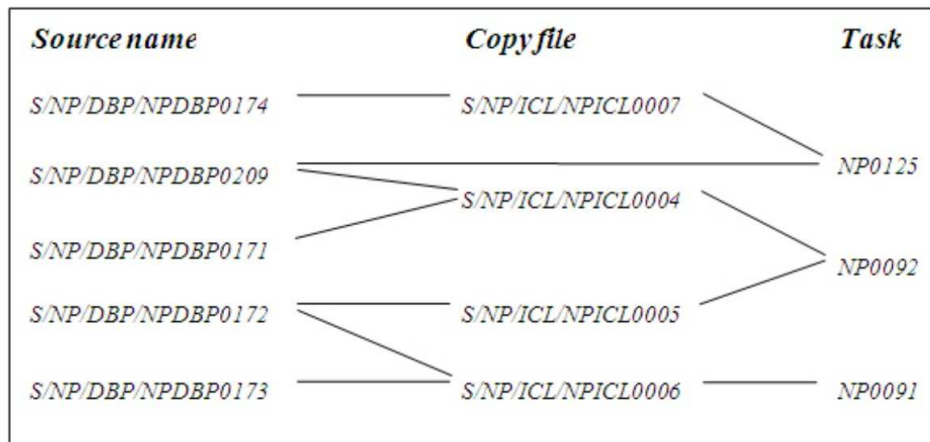
### Example

The following overview is given when four copy files and one program file are changed for three different tasks, and one of the compilations has a syntax error:

Copy file	S/NP/ICL/NPICL0004	is adapted because of task NP0092
Copy file	S/NP/ICL/NPICL0005	is adapted because of task NP0092
Copy file	S/NP/ICL/NPICL0006	is adapted because of task NP0091
Copy file	S/NP/ICL/NPICL0007	is adapted because of task NP0125
Program	S/NP/DBP/NPDBP0209	is adapted because of task NP0125

Copy file	S/NP/ICL/NPICL0004 is used by	program S/NP/DBP/NPDBP0209
	And	program S/NP/DBP/NPDBP0171
Copy file	S/NP/ICL/NPICL0005 is used by	program S/NP/DBP/NPDBP0172
Copy file	S/NP/ICL/NPICL0006 is used by	program S/NP/DBP/NPDBP0172
	And	program S/NP/DBP/NPDBP0173
Copy file	S/NP/ICL/NPICL0007 is used by	program S/NP/DBP/NPDBP0174

The following picture represents this situation:



005414



@E>421 Compilation overview

96-01-18 17:14:18 Page 1

```

Mixnumber : 279/279
Usercode  : SIMON
Family    : IDRD
Repository: (SCRATCH)INFDB ON IDRD1
Warning   Environment: MAINTENANCE
Warning   Compilation mode      : NORMAL
          Never create compilation listings
          Remove downloaded copyfiles after the compilations
          Use integrity mechanism SYSTEM BASE
          Use integrity mechanism      PROJECT NP
          Use integrity mechanism      PROJECT BASE
          Use integrity mechanism      PROJECT IS
          Use integrity mechanism      PROJECT BB
          Use integrity mechanism SYSTEM RESPECT
          Use integrity mechanism      PROJECT SRT
          Use integrity mechanism      PROJECT RIS
          Use integrity mechanism      PROJECT SURE
          Use integrity mechanism      PROJECT RESPECT
          Use start-jobs
          Disk location 'BACKUP' = DBONTW

Warning   This copy-file/task creates an integrity-chain: NP0125
Warning   This copy-file/task creates an integrity-chain: NP0091
Warning   This copy-file/task creates an integrity-chain: NP0092

          Program in the compile-queue: S/NP/DBP/NPDBP0171
          Program in the compile-queue: S/NP/DBP/NPDBP0172
          Program in the compile-queue: S/NP/DBP/NPDBP0173
          Program in the compile-queue: S/NP/DBP/NPDBP0174
          Program in the compile-queue: S/NP/DBP/NPDBP0209

COMP_1    Compile version 1.2      of S/NP/DBP/NPDBP0171
COMP_1    Compilation ok: S/NP/DBP/NPDBP0171

COMP_2    Compile version 1.1      of S/NP/DBP/NPDBP0172
COMP_2    Compilation ok: S/NP/DBP/NPDBP0172

COMP_3    Compile version 1.1      of S/NP/DBP/NPDBP0173
COMP_3    Compilation ok: S/NP/DBP/NPDBP0173

COMP_4    Compile version 1.1      of S/NP/DBP/NPDBP0174
          File is translated: SURESOURCE/S/NP/DBP/NPDBP0209
Warning COMP_4    Compilation syntax errors: S/NP/DBP/NPDBP0174

COMP_5    Compile version 1.2      of S/NP/DBP/NPDBP0209
COMP_5    Compilation ok: S/NP/DBP/NPDBP0209

Warning   Compilation failed: S/NP/DBP/NPDBP0174

```

The following tables show the waiting integrity chains. The programs that are mentioned in these chains will not be transferred, until the reason why the chain is waiting is solved. A program that is marked with SNTX blocks the integrity chain, and this program has to be compiled successfully before the waiting chain can be transferred. A task that is marked with SNTX blocks the integrity chain, because one or more generations that are triggered by the impact analysis failed. The RIS modules that need to be regenerated to solve the task are reported just below the table.

		NP0125
		NP0092
		NP0091
Sntx	S/NP/DBP/NPDBP0174	*
	S/NP/DBP/NPDBP0171	*
	S/NP/DBP/NPDBP0172	* *
	S/NP/DBP/NPDBP0173	*
	S/NP/DBP/NPDBP0209	* *
Compilations with syntax errors:		1
Aborted compilations		: 0
Successful compilations		: 4
		-----
Total:		5
End of task RESPECT/SURE/COMPILE		

The program with the syntax errors creates a waiting integrity chain, and blocks the deployment of the other programs.

23.17.2. Notify Users by Email About Compilation Syntax Errors

SURE can send emails when one or more compilations failed. The email contains an overview of all programs with compile-status syntax or with compile-status abort, and all programs that have integrity problems (waiting in an integrity chain or causing a waiting integrity chain).

Terminology

Developer	The last user who modified a source.
Project-controller	The employee function that controls a project.  This controller must be defined for the project through field "Project Controller" at the project properties screen. A project can have a different controller for each environment.
System-controller	The employee-function that controls an application system.  This controller must be defined for the project through field "System Controller" at the project properties screen. A System can have a different controller for each environment.

Environment-controller	The employee-function that controls an environment.  This controller must be defined for the project through field "Environment controller" at the environment properties screen.
------------------------	---

An email is sent to users with a valid email address in the following cases:

To developers	Developers that modified programs having syntax errors or other compile problems get an email with the programs that are last maintained by that developer. This email is not sent if option "email to developer" is disabled.
To project controllers	The email contains programs with a compilation problem and belonging to that project. This email is not sent if option "Email to Project-controller" is disabled.
To system controllers	The email contains programs with a compilation problem and belonging to that system. This email is not sent if option "Email to System-controller" is disabled.
To environment-controllers	All programs with a compilation problem are written in the email. This email is not sent if option "Email to Environment-controller" is disabled.

### Example

The following output is created:

```
SURE compile errors. Environment DEVELOP.   Date 2003-07-29 16:38
File S/BA/DBP/BADBP0178
---> COMPILE-STATUS: SYNTAX
      System      : BASE
      Project     : BA
      Modified by  : SGROOT                at 20030306 12:00:40

File S/BA/DBP/BADBP0191
---> COMPILE-STATUS: SYNTAX
      System      : BASE
      Project     : BA
      Modified by  : SIMON                 at 20030701 15:32:18

File S/BA/OLP/BAALP0023
---> COMPILE-STATUS: SYNTAX
---> Blocks the deployment of 1 objects
      System      : BASE
      Project     : BASE
      Modified by  : SGROOT                at 20030728 17:26:30
```

Each line with an arrow describes a problem.

Define a system- or project-controller through the project-options, field "Responsible for."

Define which type of emails must be created through the environment options, tab "SURE batch."

## 23.18. Compiling through a Specific Compile Queue

By default, changed programs are placed into the NORMAL compile queue, which is processed during the daily evening batch. In some cases, it may be necessary to compile programs through another compile queue.

### Examples

- A program that has to be compiled immediately to solve an urgent production error. These programs can be compiled through the COMPILE-FAST queue.
- A group of programs that have to be compiled because of database reorganization. These programs can be compiled through a user defined compile queue.
- A group of programs that have to be recompiled for other reasons (example, new MCP). These programs can be compiled through the RECOMPILE queue.

### 23.18.1. Compile Fast Queue

Programs that have to be compiled immediately can be placed into the COMPILE-FAST queue. Notice that the integrity analysis by the RESPECT/SURE/COMPILE can add other programs into this compile fast queue.

The procedure works as follows:

- Place a source into the COMPILE-FAST queue using SURE command COMPILE FAST.
- Run the RESPECT/SURE/COMPILE("FAST-MODE");VALUE=1.
- The RESPECT/SURE/COMPILE program selects all programs from the COMPILE-FAST queue and performs an integrity analysis on these programs.
- If a selected program is also linked to a user defined compile queue, then the PREVIOUS version of the source is used for compilation.
- If the selected program is NOT linked to any user defined compile queue, the newest version of the file (for that environment) is used.
- Run the RESPECT/SURE/TRANSFER("FAST-MODE, NO-DUMPJOB, THROUGH-BNA.")
- This transfers the objects to the object environment.

## 23.18.2. User Defined Compile Queue

In some cases, it is necessary to compile a very large set of programs and to deploy the compiled objects as one group. The total elapsed compile time may take several days, but in the mean time the normal compile queue has to be available for programs that are not part of the mass compile.

The above-described situation can happen because of database reorganizations, when many programs have to be recompiled with a new version of the description file. At the same time, it must still be possible to compile other programs with the old version of the description file to solve production errors.

With a special, user defined compile queue it is possible to handle this situation. The special compilations (for example, database reorganization compilations) are done through the user defined compile queue. The normal daily compilations still go through the normal compile queue.

The procedure works as follows:

- Define a compile queue through SURE function "define options" or "SURE compile options."
- Activate the compile queue with SURE command ACTIVATE <queue>.
- Link a task to the queue through task command EXT (field compile queue). All files that are adapted because of this task are linked to the task's compile queue.

Files that are saved into the repository or transferred to a next environment are automatically placed into the normal compile queue of that environment. If a file is linked to a user defined compile queue, then this file is not placed into the normal queue, but in its user defined compile queue.

- Add files manually into the user defined compile queue with SURE command COMPILE <queue>.
- Run the RESPECT/SURE/COMPILE("QUEUE=<queue>");VALUE=1 program.

The RESPECT/SURE/COMPILE program selects all files from the specified queue, and performs an integrity analysis on these programs.

The newest version of each file (in this environment) is used for the compilations. The compile timestamps of the compiled programs are not updated. This is done in a later stage.

- Run the RESPECT/SURE/TRANSFER("queue=<queue>") program

This program creates a transfer-job for the objects that are compiled through the special compile queue and are not blocked because of integrity reasons. This program should not run before all objects are compiled-ok; otherwise, an incomplete transfer-job is created. The transfer-timestamps of the transferred programs are not updated; this is done at a later stage.

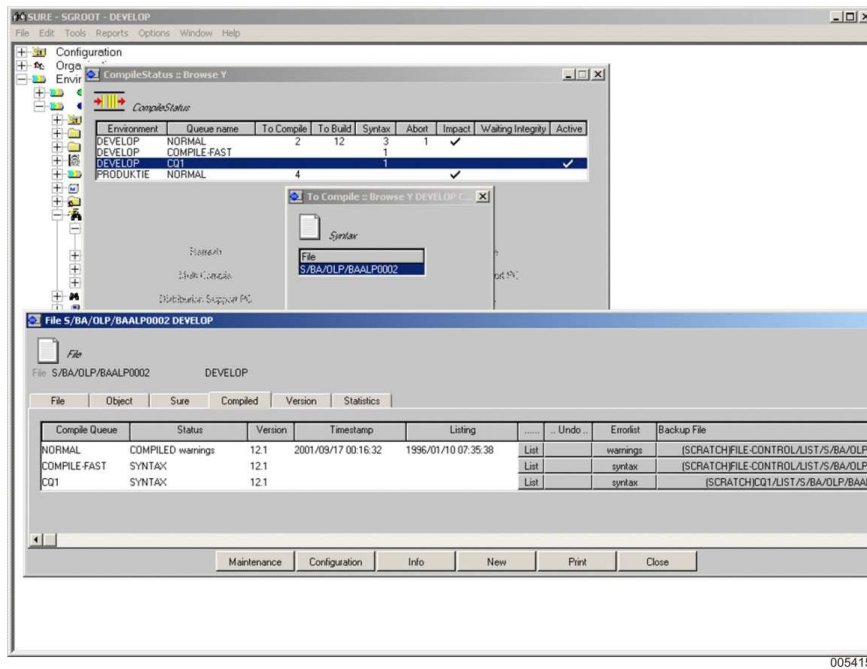
- Run the RESPECT/SURE/CLOSEQUEUE("QUEUE=<queue>") program  
This program has to run at the moment that the objects (created by the RESPECT/SURE/TRANSFER) are deployed to the run-time location. The correct compile and transfer timestamps of the compiled or transferred programs are stored in SURE, the copy files that are linked to the compile queue are copied to the work-environment (if this option is set), and the compile queue is closed.

Other characteristics are as follows:

- Programs that are linked to a user defined compile queue are normally compiled through that queue. It is still possible to compile such a program through the normal compile queue.
- If the RESPECT/SURE/COMPILE handles a user defined compile queue, the newest versions of the selected programs and copy files are used.
- If the RESPECT/SURE/COMPILE runs for the NORMAL compile queue, and a program is also linked to a user-defined queue, the previous version of that file is used for the compilation. If a program is not linked to any user-defined queue, the newest file version is used.
- If the previous version was used, then the program is automatically added again in the normal queue when the user-defined compile queue is closed. This is important when a fix is made in a copy file. The extra compilation ensures that the fix is also used in that latest compilation.
- If a quick fix is made on a file that is linked to a user defined compile queue, the quick fix will always be made on the previous version of that file. The quick fix has to be compiled through the normal compile queue. (Notice that the quick fix is made against the current run-time version of the file, which is compiled against the old version of the description file, while the newest version has to be compiled through the user defined compile queue against the newest version of the description file.)
- The RESPECT/SURE/CLOSEQUEUE program aborts if there are still programs that have to be compiled or transferred for the specified compile queue, or if there are still impact or integrity-prod indications.
- The results, the compilation listing and the error file of the compilations that are done through the user-defined queue are kept in SURE, as long as the queue is active. These compile results can be viewed through the Compile-tab on the properties screen of a source.

### Example

- Compile Interface reports syntax errors for one file in compile-queue CQ1.
- All files with syntax errors.
- Each compile queue has its own error list.
- Click this button to see the error list.



## 23.19. RESPECT/SURE/TRANSFER

If a source is compiled correctly, the RESPECT/SURE/COMPILE program changes the transfer status in the database of the source to TO-TRANSFER. Compiled programs with relation TRANSFER-STATUS (TO-TRANSFER) are transferred with the transfer the RESPECT/SURE/TRANSFER program as follows:

- Start the program to generate the transfer-jobs.
- Start the transfer-jobs. These jobs copy all new objects to the production environment.
- Start the remove-job. This job removes all the production objects from the compilation usercode.

RUN the RESPECT/SURE/TRANSFER as follows:

Input parameters :	<pre> NORMAL-MODE  THROUGH-TAPE  / FAST-MODE    / QUEUE &lt;queue&gt; / + / TRANSFERSELECTION; &lt;file name&gt; / + / OUTPUTFILE  SPILT-COPY  nnn / + / DUMPJOB     OBJECT-PREFIX &lt;PREFIX&gt; </pre>
Task string :	<p>The default mode is NORMAL-MODE</p> <p>&lt;environment&gt;.</p>
Job :	<p>SURE/TRANSFER/&lt;pack&gt;.</p> <p>This job is automatically started.</p>
Output overview :	A list of the files that are copied to the object-environment.
Example	RUN the RESPECT/SURE/TRANSFER("NORMAL-MODE");TASKSTRING = "PROD"
Where	This program runs in the SURE evening batch.
Security	MP +PU (the program uses SYSTEMSTATUS commands).
NORMAL-MODE (default)	Select files that are placed in the normal transfer queue.
FAST-MODE	Select files that are placed in the transfer fast queue.
QUEUE <queue>	Select files that are placed in the user defined transfer queue.
DUMPJOB	Enforce generation of a dump-job each object family or host. A dump-job copies all objects, jobs and data files that are defined in SURE with OBJECT-PACK = <family> and OBJECT-HOST = <host> to a backup tape.
THROUGH-TAPE	<p>The RESPECT/SURE/TRANSFER generates a deployment job each destination family and each host.</p> <p>By default, the deployment is done through tape: copy the objects from the SURE compilation directory to tape, and copy the files from tape to the object-location.</p> <p>It is possible to define each family and each host that the objects/files can be copied directly from the SURE compilation directory to the object-location (= pack to pack copy through library maintenance and through BNA if the destination pack is on another host).</p> <p>If it is defined that the objects are copied from pack-to-pack or through BNA, then that can be overruled with this parameter, so this parameter enforces that the deployment is done through tape for a single run of the RESPECT/SURE/TRANSFER.</p>
TRANSFERSELECTION <file>	<p>This option only deploys files that are in the deployment queue and are subject to a pre-defined relation. This relation must be defined in a file &lt;file&gt;. The relation must be entered as class in the first line and an asset in the second line, both terminated with an "@" sign. The transfer program transfers then only the files with this relation. This makes it possible to transfer in "phases" (for example, first the batch programs, and later the on-line programs (or the rest)). If value 2 is used and the input file is not resident, the transfer program will P-DS; if it is</p>



	<p>resident, dump jobs are not created and the transfers proceed as in value 1. This facility will not update the repository. Therefore, the transfer program must be initiated each day with a value other than 2 or 702 in order to update the production timestamps in the repository. This implies that files which were transferred previously are copied to their destination once again. The layout of the RELTRANSFER file is as follows (using the batch relation as an example):</p> <pre>1000FILE TYPE@ 2000BATCH@</pre>
SPLIT-COPY = <nnn>	<p>The RESPECT/SURE/TRANSFER generates a deployment job that copies the necessary objects to their object-locations. This is done through a library/maintenance copy command, and each object is mentioned individually in this command.</p> <p>This option can be used to limit the amount of objects each library/maintenance copy. If the amount of objects to be deployed exceeds &lt;nnn&gt;, a second library/maintenance copy is started.</p>
OBJECT-PREFIX = <prefix>	<p>All objects are copied to the object-location with the given prefix.</p> <p>This makes it possible to put the new object automatically on the run-time environment, but to control the moment that the new objects are actually activated (through a manual copy).</p>

Other characteristics:

The RESPECT/SURE/TRANSFER generates a job called the SURE/TRANSFER/<object-pack>. All objects that must be transferred to that object-pack are handled by that job. A transfer job is generated for each object-pack (if objects have to be transferred to that pack).

The actual transfer of the objects to the object-location goes through library-maintenance (a COPY statement in the job).

The original (and still default) method to transfer the objects to the object-location is through a transfer-tape. A tape is generated each destination family or host combination. The objects are copied from the SURE-pack to the transfer-tape, and the operator has to copy the objects from the transfer-tape to the object-pack (through COPY = FROM <transfer-tape> TO <object-pack>). The name of the transfer-tape is OVZ<object-pack>. OVZ is an abbreviation of the Dutch word "overset" that means "transfer."

Notice that these OVZ-tapes can be used in a backup procedure in combination with the generated dump-job for each family or host.

- Start once a week the dump-job that dumps all objects and files to tape.
- Save the daily OVZ-tape to keep the possibility to recover a family to a specific date.

### 23.19.1. Deploy for Each TASK

The default mechanism of SURE compiling and deploying software is using a batch which runs on a scheduled basis, that is once or twice a day. For development environments "instant" compile and deploy is supported. This mechanism is not flexible enough when a site wants to deploy a task at a certain date and time in production. Normally, the majority of tasks can be supported using the default mechanism; however, there are conditions where the date/time schedule for each batch is required.

#### Schedule Task

The task transfer dialog contains a button named Queue.

**Transfer task T\_0106 from environment DEVELOPMENT**

Task

Name: T\_0106      Reported By: DEVELOPER      Mastertask

Environment: DEVELOPMENT      Solved By:

Status: DEVELOPMENT/ASSIGNED

Adapt quick fix screen

Target Environment	Blocked	Overlap	Fatal	Warning
DEVELOPMENT		true	true	Can not transfer to the same environment
INTEGRATION		true		
ACCEPTANCE		true		
PRODUCTION		true		Overlapping quick-fix tasks;

Transfer Queue Block Detail Compile Impact

Linked

Environment	Changed By	Entity	Item	Link	Error
DEVELOPMENT		Task	T_0111		
DEVELOPMENT		Task	T_0113		
DEVELOPMENT		Task	T_0114		
DEVELOPMENT		Task	T_0117		

Link Move Task Detail Undo Request Undo Assign Check In

Close

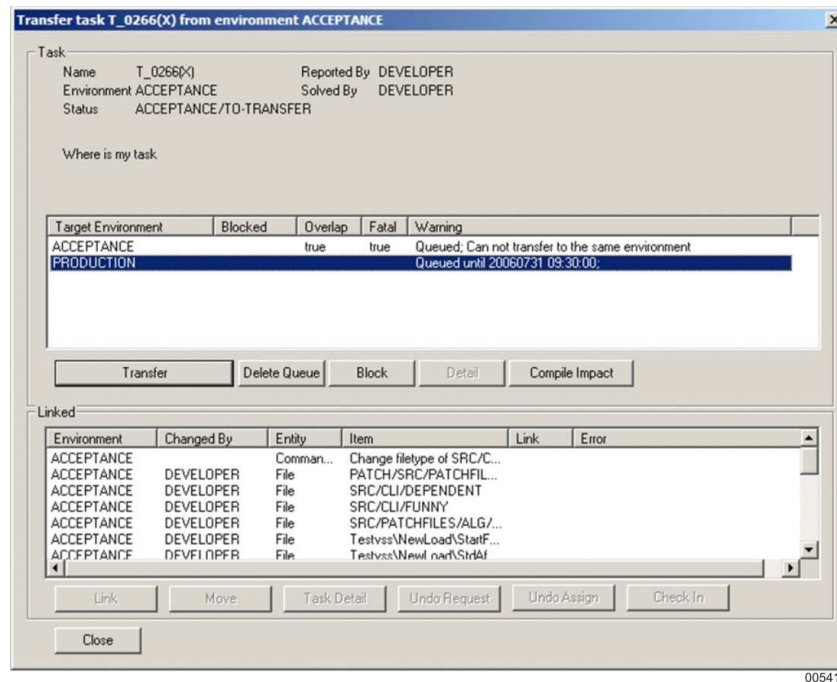
005416

The queue button allows entry of a date/time.

**Note:** You must first transfer the task to the environment prior to where the queue must be used. So, in this case, if the QUEUE functionality must be used in PRODUCTION, then this task must first be transferred to ACCEPTANCE before the QUEUE can be set.

The QUEUE button pops up a form which allows entry of date and time. Now the task is scheduled to be transferred but it is not actually transferred yet. This means that the files are not transferred yet until the time expires and they are transferred by the batch program.

After the QUEUE command, the task shows that it is queued as shown in the next screen example.



## RESPECT/TASK/TRANSFER

This program handles the queued task transfers.

Run the RESPECT/TASK/TRANSFER("DEPLOY-PER-TASK");TASKSTRING="<environment>"

This mode enforces the following:

- The RESPECT/TASK/TRANSFER does not automatically go to end-of-task.
- The program checks every minute for tasks that are placed in the transfer-queue for <environment>. The queued tasks are then transferred to <environment> one by one.
- The program starts for each transferred task a batch job and waits until that job is finished before it begins with the next task.
- The program checks every minute for tasks that are placed in the transfer-queue for <environment>, but with command HI the check is done immediately.
- The program goes to end-of-task after a HI99.
- The program gives debug displays after a HI10.
- A task cannot be transferred while the regular SURE evening job is busy. The program will then wait until that job is finished.
- The batch job will not run simultaneously with the regular SURE evening job. These two jobs will wait for each other through an exclusive synchronization file.
- The batch job does an EXAMINE, COMPILE and DEPLOY for the sources that were just transferred with the task to <environment>.
- The batch job uses the SURE/WFLINCLUDE/BATCH include-file if that include file is resident on disk or in SURE.

### 23.19.2. Pack-to-pack Copy or BNA Copy

It is possible to indicate each object-pack (and for each environment) that the library maintenance copy has to go directly from the SURE-pack to the object-pack and not through an OVZ-tape. This option must be set as follows:

- Right-click the name of the applicable SURE environment, and choose "Properties" from the sub-menu.
- Click the tab "SURE transfer."
- Enter the name of the object-pack in the column "Disk family name." If you leave the corresponding field "BNA hostname" (on the same line as the disk-family--name) empty, then SURE assumes that the <object-pack> is one of the packs of the current host; otherwise, a BNA-copy is started for each family or host combination.

#### Example

Disk Family Name	BNA Hostname	
PACK1	<empty>	All files that have <code>OBJECT-PACK(PACK1)</code> and <code>OBJECT-HOST = empty</code> are copied directly from the SURE batch pack to <code>PACK1(PACK)</code> .  In this case, it is assumed that <code>PACK1</code> belongs to <code>MyHost</code> .  This line does not select N.B. Files with an object-host.
PACK1	HOSTA	All files that have <code>OBJECT-PACK(PACK1)</code> and <code>[OBJECT-HOST(HOSTA) OR OBJECT-HOST = empty]</code> are copied directly (through BNA) to <code>PACK1(PACK,HOSTNAME=HOSTA)</code> .
PACK1	HOSTA	All files that have <code>OBJECT-PACK(PACK1)</code> and <code>OBJECT-HOST = empty</code> are copied directly (through BNA) to <code>PACK1(PACK,HOSTNAME=HOSTA)</code> and to <code>PACK1(PACK,HOSTNAME=HOSTB)</code> .  Files that have <code>OBJECT-PACK(PACK1)</code> and <code>OBJECT-HOST(HOSTA)</code> are only copied to <code>PACK1(PACK,HOSTNAME=HOSTA)</code> .  Files that have <code>OBJECT-PACK(PACK1)</code> and <code>OBJECT-HOST(HOSTB)</code> are only copied to <code>PACK1(PACK,HOSTNAME=HOSTB)</code>
PACK1	HOSTB	
<empty>	HOSTB	All files that have <code>OBJECT-HOST(HOSTB)</code> are copied directly (through BNA) to <code>&lt;object-pack&gt;(PACK,HOSTNAME=HOSTB)</code> .

The disk-to-tape-to-disk method is still the default for the following reason:

If somebody enters by accident a wrong object-pack-name then that object will not be copied automatically to the wrong place, but the transfer starts waiting for an OVZ-tape. In that case, you still have the possibility to fix the problem and to make sure that the object is copied to the correct location.

## Relations

Selection of files takes place through one of the following relations, depending on the start value:

```
FILE-CONTROL | <filename> :TRANSFER-STATUS (TO-TRANSFER)
COMPILE-FAST | <filename> :TRANSFER-FAST (TO-TRANSFER)
```

The transfer status is changed:

```
Add      FILE-CONTROL | <filename> :TRANSFER-STATUS (TRANSFERRED)
Delete    FILE-CONTROL | <filename> :TRANSFER-STATUS (TO-TRANSFER)
          COMPILE-FAST | <filename> :TRANSFER-STATUS (TO-TRANSFER)
```

The production environment for an object is determined by the relations:

```
FILE-CONTROL | <filename> :OBJECT-USERCODE (<usercode>)
FILE-CONTROL | <filename> :OBJECT-PACK (<packname>)
FILE-CONTROL | <filename> :OBJECT-HOST (<hostname>)
```

## 23.19.3. User-Hook in the Deployment Procedure

The RESPECT/SURE/TRANSFER program creates the following output file:

SURE/TRANSFER/<queue>/<pack>/DEPLOY

The layout of this file is: Recordsize = 300 characters

Pos 0 thru 17	<object-usercode>
Pos 18 thru 35	<pack>
Pos 36 thru 299	<object-name>

This deploy-summary-file contains the names of all the files that are deployed to <pack>.

Some sites use the SURE/WFLINCLUDE/TRANSFER include-file to perform some site-specific actions in the deployment job SURE/TRANSFER/<pack>.

The SURE/WFLINCLUDE/TRANSFER file is included by various jobs (startlog, transfer, secure, cleaner, delivery). Some statements in the SURE/WFLINCLUDE are only applicable for one of these jobs. In that case, job-attribute MYNAME is used to determine the name of the job, for example as follows:

```
10000200 STRING MYNAME
10000220      ,DEPLOYFILE
10000240      ;
19999999%-----
20000000 MYNAME:=MYSELF(NAME);
20002000 IF LENGTH(MYNAME) > 14 THEN
20003000     IF TAKE(MYNAME,14) = "SURE/TRANSFER/" THEN BEGIN
20004000         DEPLOYFILE:=MYNAME & "/DEPLOY";
20005000         IF FILE #DEPLOYFILE IS RESIDENT THEN BEGIN
20006000             % perform these site-specific actions for deployment jobs
20007000         END;
20008000     END;
```

This method uses the fact that the name of the deployment file is equal to the internal name of the deployment job followed by "/DEPLOY".

### 23.19.4. Creation of Transfer Jobs

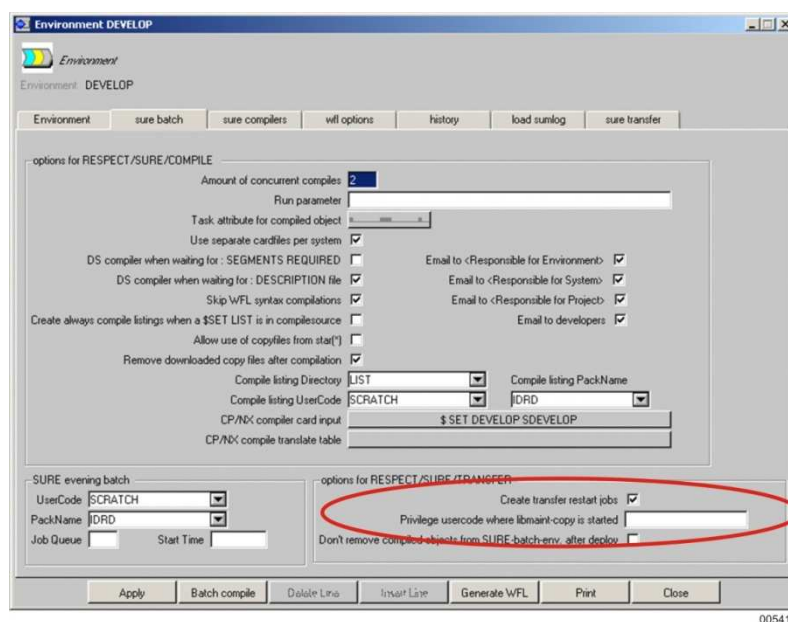
- Generate and start a job "SURE/TRANSFER/<pack>" (one transfer-job for each pack), which copies objects and jobs to a transfer-tape or directly to pack.

By default, the objects are transferred through tape, and a separate transfer job (SURE/TRANSFER/<family>) is generated for each object-family. In this situation, there is no connection between family names and host names. Each transfer job contains a library maintenance copy statement to a tape called "OVZ"<family>. This tape has to be loaded on the indicated family. If necessary, the tape can be loaded on the same family on multiple hosts.

If (besides the object-family) an object host is defined for a file, then a special transfer job is generated. This transfer job is called SURE/TRANSFER/<family>-"<host>," and contains a library maintenance copy statement to a tape called "OVZ'<family>'"<host>". This tape has to be loaded on the indicated family or host.

It is possible to indicate (for each family and environment) that the library maintenance copy to that family has to go directly from pack to pack, and not through an "OVZ"<family> tape. This option can be set through the environment properties dialog, tab "Transfer." If the family name is not linked to a host name then it is assumed that the family is one of the packs of the current host. If the family name is linked to one or more host names, then a BNA copy is started for each family or host combination.

- The transfer job executes a library maintenance copy, which may have to copy object-files to another usercode. This is only possible if the transfer job is started from a privilege usercode. But that may trigger security issues. Through an option it is possible to define a privilege usercode in SURE which is used by the transfer job to do the library-maintenance copy.



Because of this feature, it is not necessary that the SURE-batch usercodes are PU.

- Generate and start a job "SURE/TRANSFER/REMOVE\_OBJECTS," which first waits until all copies have been done, then removes all the copied objects from the compilation directory. It is possible to define that the object-file is not removed from the compilation-directory after the deployment. This can be done at three levels as follows:
  - For an individual program (Click <file> then, select properties and click tab, Sure)
  - For a file-type (Click <file-type> then, select properties and click tab, File Type)
  - For an environment (Click <environment> then, select properties and click, tab, Sure Batch)
- Generate a restart job "SURE/TRANSFER/RESTART/<pack>" if the option to create restart jobs is set.
- Generate a job that makes a dump of the production pack after the objects have been placed in the production environment. Job = "SURE/DUMP/<pack>." This job is copied to the production environment together with the new production objects.
- Create an overview listing of the files transferred and their destinations.

### Examples of generated and started jobs

Without a restart job:

```
#FILE SURE/TRANSFER/FILE-CONTROL/REMOVE_OBJECTS
10000010 BEGIN JOB SURE/TRANSFER/FILE-CONTROL/REMOVE_OBJECTS;
10000020 $ INCLUDE SURE/WFLINCLUDE 10000000 TO 19999999
10000030 SUBROUTINE DOREM0; BEGIN
10000040 REMOVE (JAN)OBJECT/RESPECT/SURE/SETCOMPILE ON IDR;
10000050 END;
10000060 WAIT(1);
70000010 WAIT(FILE SURE/TRANSFER/READY/IDRD/090993132730 IS RESIDENT);
70000020 REMOVE SURE/TRANSFER/READY/IDRD/090993132730;
99999979 DOREM0;
99999989 $ INCLUDE SURE/WFLINCLUDE 20000000 TO 29999999
99999999 END JOB;

#
#FILE SURE/TRANSFER/FILE-CONTROL/IDRD
10000010 BEGIN JOB SURE/TRANSFER/FILE-CONTROL/IDRD;
10000020 $ INCLUDE SURE/WFLINCLUDE 10000000 TO 19999999
10000030 STRING S;
10000040 FILE F(DEPENDENTSPECS=TRUE,NEWFILE=TRUE,KIND=DISK);
10000050 TASK T;
10000060 SUBROUTINE DOSEC0; BEGIN
10000070 S:= "SURE/TRANSFER/READY/IDRD/090993132730";
10000080 F(TITLE=#S);
10000090 REMOVE SURE/TRANSFER/READY/IDRD/=;
10000095 OPEN(F);
10000100 SECURITY (INFRA)SURE/DUMP/IDRD FROM IDR PUBLIC IO;
10000110 SECURITY (INFRA)OBJECT/RESPECT/SURE/SETCOMPILE
10000120 FROM IDR PUBLIC IO;
10000130 END;
10000140 DOSEC0;
25000010 BACK :
25000020 COPY (INFRA)SURE/DUMP/IDRD
25000030 ,(INFRA)OBJECT/RESPECT/SURE/SETCOMPILE
25000040 AS *OBJECT/RESPECT/SURE/SETCOMPILE
25000050 FROM IDR(PACK) TO IDR(PACK) [T];
25000370IF T ISNT COMPLETEDOK THEN BEGIN
25000380 ERRCOPY:=ERRCOPY+1;
25000390 IF ERRCOPY < 3 THEN GO BACK;
25000400 SS:=ACCEPT("SURE deploy failed (see jobsummary);"&
25000410 " <mx>DS or <mx>AX:RETRY");
25000420 GO BACK;
25000430END;
25000440IF T (VALUE) NEQ 0 THEN BEGIN
25000450 SS:=ACCEPT("SURE deploy was incomplete;"&
25000460 " <mx>DS or <mx>AX:RETRY or <mx>AX:CONTINUE");
25000470 IF SS NEQ "CONTINUE" THEN GO BACK;
25000480END;
99999910ENVIRONMENTNAME="DEVELOP-RIS";
99999920 $ INCLUDE SURE/WFLINCLUDE 20000000 TO 29999999
```



```
99999930 CRUNCH(F);
99999999 END JOB;
```

With a restart job:

```
#FILE SURE/TRANSFER/FILE-CONTROL/IDRD
10000010 BEGIN JOB SURE/TRANSFER/IDRD;
10000020 $ INCLUDE SURE/WFLINCLUDE 10000000 TO 19999999
10000030 STRING S,SS
10000040 FILE F(DEPENDENTSPECS=TRUE,NEWFILE=TRUE,KIND=DISK);
10000050 TASK T;
10000060 SUBROUTINE DOSEC0; BEGIN
10000070     S:= "SURE/TRANSFER/READY/IDRD/100693142729";
10000080     F(TITLE=#S);
10000090 REMOVE     SURE/TRANSFER/READY/IDRD/=;
10000100 SECURITY (INFRA)SURE/DUMP/IDRD FROM IDR PUBLIC IO;
10000110 SECURITY (INFRA)OBJECT/RIS/GENERATE/LFI
10000120     FROM IDR PUBLIC IO;
10000130 SECURITY (INFRA)OBJECT/RESPECT/SURE/FIND
10000140     FROM IDR PUBLIC IO;
10000210 END;
10000220 DOSEC0;
25000010 BACK :
25000020 COPY (INFRA)OBJECT/RIS/GENERATE/LFI
25000030     AS *OBJECT/RIS/GENERATE/LFI
25000040     ,(INFRA)OBJECT/RESPECT/SURE/FIND
25000050     AS *OBJECT/RESPECT/SURE/FIND
25000060 FROM IDR(PACK) TO IDR(PACK) [T];
25000370 IF T ISNT COMPLETEDOK THEN BEGIN
25000380     ERRCOPY:=ERRCOPY+1;
25000390     IF ERRCOPY < 3 THEN GO BACK;
25000400     SS:=ACCEPT("SURE deploy failed (see jobsummary);" &
25000410         " <mx>DS or <mx>AX:RETRY");
25000420     GO BACK;
25000430 END;
25000440 IF T (VALUE) NEQ 0 THEN BEGIN
25000450     SS:=ACCEPT("SURE deploy was incomplete;" &
25000460         " <mx>DS or <mx>AX:RETRY or <mx>AX:CONTINUE");
25000470     IF SS NEQ "CONTINUE" THEN GO BACK;
25000480 END;
99999969 ENVIRONMENTNAME:="DEVELOP-RIS";
99999979 $ INCLUDE SURE/WFLINCLUDE 20000000 TO 29999999
99999989 CRUNCH(F);
99999999 END JOB;
#

FILE SURE/TRANSFER/RESTART/IDRD
10000010 BEGIN JOB SURE/TRANSFER/RESTART/IDRD;
10000020 $ INCLUDE SURE/WFLINCLUDE 10000000 TO 19999999
10000030 TASK T;
10000040 BACK :
10000050 COPY (INFRA)SURESARE/OBJECT/RIS/GENERATE/LFI
```

```
10000060      AS *OBJECT/RIS/GENERATE/LFI
10000090      ,(INFRA)SURESAVE/OBJECT/RESPECT/SURE/FIND
10000100      AS *OBJECT/RESPECT/SURE/FIND
10000110      FROM IDRD(PACK) TO IDRD(PACK) [T];
10000120      IF T ISNT COMPLETEDOK THEN GO BACK;
10000130      IF T(VALUE) NEQ 0 THEN BEGIN
10000140          WHILE ACCEPT("COPY FAILED: DS OR AX:OK FOR RETRY") NEQ "OK" DO;
10000150              GO BACK;
10000160      END;
99999989 $ INCLUDE SURE/WFLINCLUDE 20000000 TO 29999999
99999999 END JOB;
#
#FILE SURE/TRANSFER/REMOVE_OBJECTS
10000010 BEGIN JOB SURE/TRANSFER/REMOVE_OBJECTS;
10000020 $ INCLUDE SURE/WFLINCLUDE 10000000 TO 19999999
10000030 SUBROUTINE DOREM0; BEGIN
10000040     CHANGE (INFRA)OBJECT/RIS/GENERATE/LFI
10000050         TO (INFRA)SURESAVE/OBJECT/RIS/GENERATE/LFI FROM IDRD;
10000080     CHANGE (INFRA)OBJECT/RESPECT/SURE/FIND
10000090         TO (INFRA)SURESAVE/OBJECT/RESPECT/SURE/FIND FROM IDRD;
10000140 END;
10000150 WAIT(1);
70000010     WAIT(FILE SURE/TRANSFER/READY/IDRD/100693142729 IS RESIDENT);
70000020         REMOVE SURE/TRANSFER/READY/IDRD/100693142729;
99999979 DOREM0;
99999989 $ INCLUDE SURE/WFLINCLUDE 20000000 TO 29999999
99999999 END JOB;
#
```

The RESPECT/SURE/TRANSFER creates jobs that copy the compiled objects (and other files) to the final run-time location, and remove the compiled objects from the SURE compile directory.

A deploy-job is generated for each pack/host combination. These deploy-jobs are automatically zipped by the RESPECT/SURE/TRANSFER, and they copy the objects through library/maintenance to the final locations. Each deploy-job creates at the end a "deploy-ready" file.

At the end, when all library/maintenance copies are done, a remove-job is zipped. This remove-job waits until all "deploy-ready" file are resident and removes then the compiled objects from the SURE compile directory. It is possible to define each individual file (file-properties) or each file-type (file-type properties) or for each environment (Select environment-options and click the tab, SURE batch) that objects must remain in the SURE-compile directory after the deploy.

The task value of each library/maintenance copy (of the deploy jobs) is checked after the library/maintenance copy is done. This task value should be zero, which means that all mentioned objects are correctly copied.

If the task value is not zero, then one or more objects were missing (this situation should not be possible in the first place, but we check it anyway). The names of missing files can be found in the job-summary of the deploy-job. The deploy-job starts waiting for operation input with the following message:

```
SURE deploy was incomplete; <mx>DS or <mx>AX:RETRY or <mx>AX:CONTINUE
```

The following operator actions are possible:

<mix>AX RETRY	The library/maintenance copy is started again. This action is only relevant if the missing object is made resident in the mean time; otherwise, the job finds the same error when the second library/maintenance copy is done.
<mix>AX CONTINUE	The missing files are checked, and you came to the conclusion that it does not matter. The process continues as if nothing had happened.
<mix>DS	The deploy-job is DS-ed so it cannot create the "deploy-ready" file. As a result the remove-job (that runs afterwards) keeps on waiting for that "deploy-ready" file. You must DS the correct RESPECT/TOOLS program (that runs in the remove-job and waits for the 'deploy-ready' file) to continue the process.

The recommended operator actions are AX:RETRY or AX:CONTINUE.

### 23.19.5. Copy or Dump Objects

If the transfer of objects is performed using tapes, then after the creation of transfer tapes the following procedures have to be carried out:

- Physically move the transfer-tape (OVZ<pack>) on to the correct production system. Then copy the files from tape to the production system with the "COPY = FROM tape TO <pack>(PACK)" command. The operator can determine the exact moment that the objects are copied from tape to the production environment.
- The tape transfer can still be used by sites that normally do pack-to-pack or BNA transfers in the event of extensive compiles (example, for a database reorganization). If the required description file is available, the necessary programs can be compiled before the actual reorganization, and copied to tape by running the transfer program with parameter THROUGH-TAPE. After the actual reorganization has taken place, the new objects need to be copied from the tape to their pack, avoiding BNA transfer delay.
- After the files are copied to the production packs, the operator can start the job "SURE/DUMP/<pack>." This job dumps the objects and workflows to a tape that can be used for pack crash recovery. In the dump job, a check is made to determine whether the production packs contain all objects according to the repository information. This allows the detection of removed objects from a production pack, while the OBJECT-PACK relation in the repository implies that the object is present on a production pack. In this situation, a discrepancy exists between the theoretical situation (according to the repository information) and the

actual situation. A “not copied” warning is issued in the job summary for this situation.

The result of the transfer/dump procedure is that the objects are on the correct packs, and that a backup tape is made of production objects. For a safe tape control, it is essential that a minimum of two versions of the backup tape be saved. The backup tape can be used when there is a pack crash, and also in the case of production problems with the new objects. In the latter situation, one of the old backup versions can be used.

### 23.19.6. Extra Object-Location to Test Quick Fixes

It is possible to fill an extra object-location to test quick fixes. This feature is only applicable for the MCP platform.

#### Detailed Information

Quick fixes are often made in the SURE production environment. The developer who works on the quick fix may require a full run-time environment (with objects and a test-database) to test his quick fixes. On the other hand: it is certainly not allowed that the quick fix is tested in the *production* run-time environment. Moreover, it is even not allowed to overwrite a production-object with a quick fixed object that the developer just compiled.

#### Solution

System option “Copy compiled objects to the work-location (environment to test quick fixes)” offers the possibility to fill an extra object-location to test quick fixes. If the option is enabled, the objects that are compiled by SURE are deployed to two locations:

- The regular object-location of each source.
- The directory that is defined as work-location to make quick fixes.

This is a system-option and can be found on the properties screen of a system, tab “SURE.”

#### Example

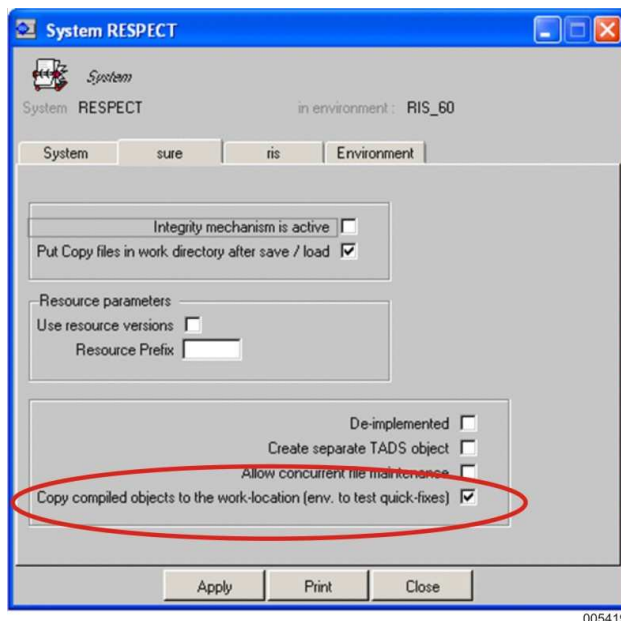
Consider system SYS1 with the following locations for environment PROD:

- Object-location = (PROD) ON PRODPK
- Work-location = \* ON PRODPK
- The SURE-batch runs under (SUREPROD) ON PRODPK

The work-location is \* ON PRODPK, and that means that each developer does his quick fixes in PROD under his private usercode on PRODPK.

If option "Copy compiled objects to the work-location (environment to test quick fixes)" is enabled, the objects that are compiled during the SURE-batch are deployed to (PROD) ON PRODPK (the regular object-location) and also to \* ON PRODPK.

The developer uses the objects in directory "\*" on PRODPK during his tests. When he starts a local compilation, the resulting object is also copied to "\*" on PRODPK. The objects in the operational object-location (PROD) ON PRODPK are not accessed by the quick fixer.



## 23.20. Manually Started Compilations by Developers

### 23.20.1. Local Compilation Started from CANDE

A standard compile job is available to compile a CANDE source and to link predefined object attributes to the resulting objects.

The name of the compile job is WFL/<environment>/COMPILE, which indicates that a separate compile job is available for each SURE environment.

The job has to be started from CANDE as follows:

```
START WFL/<environment>/COMPILE(<parameter>)
```

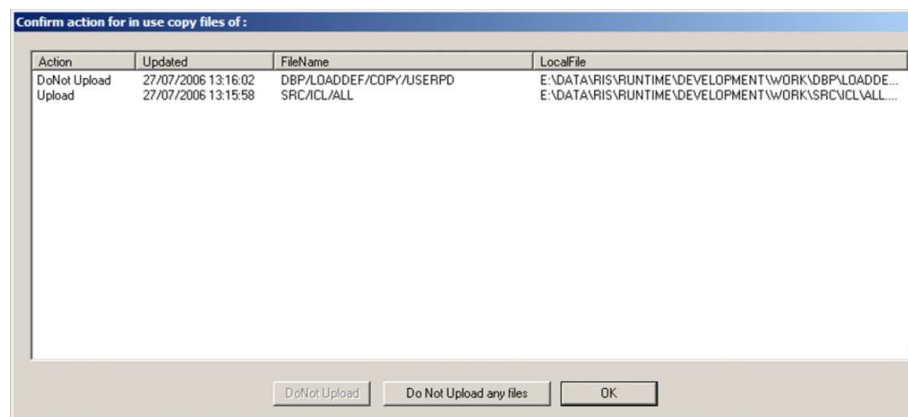
### 23.20.2. Local Compilation Started from SUREforWindows

Local compilations that are started from SUREforWindows use job WFL/<environment/PC-SESSION

SUREforWindows offers the followings methods to start a local compilation which is executed on the MCP.

- Right-click the source-name then, select Compile and click Local.
  - If the file is checked out, it is first copied to the MCP before the compilation is started.
- If you use MultiEdit as local editor then refer to Section 11, "Local Source Editing on PC" for details on how to define the local compilation method for that editor.
  - You can compile files that are checked out
  - You can also compile files that are in your download directory on the PC (See screen Options □ SURE options). These are temporary (test) programs that are not loaded in SURE. The local compile uploads the source to your CANDE directory on the MCP and compiles it over there.

The Compile Local module checks the status of the copy files of a given source when the source is compiled. If these copy files are checked out they appear in a dialog where the user can choose to upload these files too. The correct status Upload/DoNot Upload is pre-filled for each copy file according to the last upload time and the file modified timestamp, but you can manually overwrite the prefilled status.



005420

The following dependent files are mentioned in the list:

- Checked out copy files or dependent files that are referred by the source that is compiled.
- Checked out copy files or dependent files that are checked out under the same task as the source that is compiled.

The files that are changed since the previous upload are marked with "Upload."

The files that are not changed since the previous upload are marked with "Do not upload."

The files that are marked with "Upload" are uploaded to the MCP before the compilation is started.

### 23.20.3. Extra Copy of Compiled Object

It is possible to copy the compiled object (in a manually started compilation) to a second location.

#### Detailed Information

This feature is implemented as an option of the RIS/COMPLETE/OBJECT program:

```
RUN OBJECT/RIS/COMPLETE/OBJECT( "<source>--<object>" );  
TASKSTRING="<environment>,<user>,<pack>"
```

- <object> is optional. If not used, the object-name as defined in SURE is used.
- <user>,<pack> is optional. If used, the compiled object is also copied to that location.

The RIS/COMPLETE/OBJECT program runs in the compile job that is started by developers. It adds attributes that are defined in SURE to the compiled object (such as security and privilege program) and it copies the compiled object to the object-location that is defined in SURE. This feature makes it possible to copy the compiled object to a second location.





## Section 24

# Binding of Programs

SURE compiles the programs in the SURE compile queue during the evening batch. SURE can bind the compiled objects to a host. For this purpose, a number of relations are required in the database. The bind function of SURE supports in-place binding as well as total binding. The main difference is that the total bind is started up at the end of all compilations, while the in-place bind takes place directly after the compilation of a single object.

The relation that defines “which source must be bound in which driver” is:

```
FILE|<source-name>:DRIVER(<driver-name>)
```

The name of the start-job is defined by one of the following two relations:

```
FILE|<driver-name>:START-JOB(<start-job-name>)  
FILE|<source-name>:START-JOB(<start-job-name>)
```

**Note:** *These relations are created in the group FILE. Therefore, it is required to transfer the relations to the next environment if the program is transferred to the next environment.*

If a source is linked to a driver (through a DRIVER relation), and the driver has a start-job, then the start-job of the driver is zipped at the end of all compilations. This method is typically used with binding, where many objects have to be bound to a single driver-object.

If a start-job is linked to a source, then that start-job is zipped immediately after the compilation of that source (but not if the source is also linked to a driver and the driver has a start-job, because then the driver-start-job will be zipped at the end of all compilations).

The RESPECT/SURE/COMPILE adds the relation <filename>:JOB-STATUS(JOB-BUSY) for the owner of the start-job and waits until the start-job is complete.

The RESPECT/SURE/COMPILE waits until the relation JOB-STATUS(JOB-BUSY) is gone. Therefore, it is required to run the RESPECT/SURE/FINISH("<filename>") at the end of the start-job, to add the result of the job into the repository and to change relation JOB-STATUS(JOB-BUSY) to JOB-STATUS(JOB-READY), so that the RESPECT/SURE/COMPILE can finish.

Run the RESPECT/SURE/FINISH with task value = 0 if no errors are found by any task in the start-job. This adds the relation COMPILE-STATUS(COMPILED) and puts the file in the deployment queue.

Run the RESPECT/SURE/FINISH with task value = 2 if syntax errors are found by a task in the start-job. This adds the relation COMPILE-STATUS(SYNTAX) and creates a waiting integrity chain if necessary.

Run the RESPECT/SURE/FINISH with task value = 1 if another problem occurred. This adds the relation COMPILE-STATUS(ABORT) and creates a waiting integrity chain if necessary.

The following example explains the procedure in detail.

### Example of Total Bind

The programs are compiled by the RESPECT/SURE/COMPILE.

After compilation, all programs are bound at the same time (if necessary).

The program to be bound	: PROG/BIND
The host in which the program has to be bound	: BIND/HST
The final object (driver)	: OBJECT/DR/BIND1
The job in which the bind command is found	: BIND/JOB

The driver is entered by its name in the database so that relations can be attached.

Add the drivename DR/BIND1, and check that the driver has an object-location.

PROG/BIND has a "driver" relation with OBJECT/DR/BIND1 as follows:

```
FILE | PROG/BIND : DRIVER (DR/BIND1)
```

OBJECT/DR/BIND1 is created by the job BIND/JOB. Therefore, DR/BIND1 has the following "start-job" relation:

```
FILE | DR/BIND1 : START-JOB (BIND-JOB)
```

The RESPECT/SURE/COMPILE compiles PROG/BIND and determines that this has a "driver" relation with DR/BIND1. Therefore, DR/BIND1 receives the relation:

```
FILE | DR/BIND1 : JOB-STATUS (TO-START)
```

When all objects have been compiled, the RESPECT/SURE/COMPILE program searches for files with the relation:

```
FILE-CONTROL | <filename> : JOB-STATUS (TO-START)
```

For every file found, the start-job is started with START BIND/JOB ("**<environment>**,  
DR/BIND1").

Remove relation:

```
FILE-CONTROL | <filename> : JOB-STATUS ( TO-START )
```

Add relation:

```
FILE-CONTROL | <filename> : JOB-STATUS ( JOB-BUSY )
```

The last action in the start-job is a run of RESPECT/SURE/FINISH. This program sets the result of the job in the repository and informs the RESPECT/SURE/COMPILE that the start-job is complete and that the compilation process can proceed.

Remove relation:

```
FILE-CONTROL | <filename> : JOB-STATUS ( JOB-BUSY )
```

Add relations:

```
FILE-CONTROL | <filename> : JOB-STATUS ( JOB-READY )  
FILE-CONTROL | <filename> : COMPILE-STATUS ( COMPILED )  
FILE-CONTROL | <filename> : TRANSFER-STATUS ( TO-TRANSFER )
```

The RESPECT/SURE/COMPILE program adds the relation JOB-STATUS(JOB-BUSY) to a source when a start-job is zipped for that source, and the program starts waiting until the JOB-BUSY relation is gone.

At the end of the bind-job, the RESPECT/SURE/FINISH has to be started with the same parameter string as was used to start the bind job.

Run the RESPECT/SURE/FINISH with task value = 0 if no errors are found by any task in the start-job. This adds the relation COMPILE-STATUS(COMPILED) and adds the file in the deployment queue.

Run the RESPECT/SURE/FINISH with task value = 2 if syntax errors are found by a task in the start-job. This adds the relation COMPILE-STATUS(SYNTAX) and creates a waiting integrity chain if necessary.

Run the RESPECT/SURE/FINISH with task value = 1 if another problem occurred. This adds the relation COMPILE-STATUS(ABORT) and creates a waiting integrity chain if necessary.

In the case of a syntax or an abort, the program is not added to the deployment queue. If the integrity mechanism is used and the start-job did not complete correctly, then the relation ABORT(INTEGRITY-PROD) is added. The integrity mechanism adds an integrity relation to a driver if one of the driver-programs is part of an integrity chain.

The RESPECT/SURE/COMPILE goes to end-of-task when all relations JOB-STATUS (JOB-BUSY) are changed (many jobs may have been started). The next program that runs is the deployment program, RESPECT/SURE/TRANSFER.

The environment that is passed by the RESPECT/SURE/COMPILE to the bind-job is used as a task string for the RESPECT/SURE/FIND program to ensure that this program runs for the same environment as the compile program.

The started bind-jobs appear similar to the following example:

```
100 BEGIN JOB BIND/JOB (STRING VERSION, STRING TITLE);
200 TASK T;
300 T(STATUS=NEVERUSED);
400 BIND OBJECT/DR/BIND1 WITH BINDER [T] LIBRARY;
500 PL FILE HOST = OBEJCT/BIND/HST;
600
700 BINDER DATA
800 BIND A1 FROM OBJECT/PROG/BIND;
900 ?
1000 IF T IS COMPLETEDOK THEN BEGIN
1010     IF T IS COMPILEDOK THEN BEGIN
1020         RUN OBJECT/SURE/FINISH(TITLE);TASKSTRING=VERSION;
1030     END ELSE BEGIN
1100         RUN OBJECT/RESPECT/SURE/FINISH(TITLE);VALUE=2;TASKSTRING=VERSION;
1110     END;
1200 END ELSE BEGIN
1300     RUN OBJECT/RESPECT/SURE/FINISH(TITLE);VALUE=1;TASKSTRING=VERSION;
1310 END;
1400
1500 END JOB.
```

All jobs that are specified in a start-job relation must have two string parameters.

- The first string parameter identifies the environment and the compile queue. This string must be passed directly without modification to the task string of the RESPECT/SURE/FINISH (see example).
- The second string parameter identifies the FileName to which the "start-job" is related. This string must be passed without modification as a parameter to the RESPECT/SURE/FINISH (see example).

Run the RESPECT/SURE/FINISH as follows:

Input parameters	—<source name>—	
	:	Refer to Section 24, "Binding of programs," for detailed information.
Task string	:	<div> <div>—</div> <div>&lt;environment&gt;—</div> </div> <div> <div>└─&lt;queue&gt;┘</div> </div>
Task value	:	0 = Report a successful run of the start-job. 1 = Report that unknown errors were found by the start-job. 2 = Report that syntax errors were found by the start-job

### Example

```
RUN RESPECT/SURE/FINISH("DR/BIND1");TASKSTRING = "PROD"
```

Where

This program runs in the SURE evening batch.

Security

MP +PU

MP +SECADMIN (the program adds predefined securities to another object)

Parameter details:

<source-name>	The result of the start-job is linked to this source.
<queue>	If the compile-queue is not the normal queue, then the queue is passed to the RESPECT/SURE/FINISH through the task string.

## 24.1. (Binding) START-JOB and DRIVER Relations Overview

Start-job and driver relations are used in the batch compilation procedure.

A start-job relation identifies a job that has to be started after an object is compiled successfully. The start-job functionality is mostly used for binding purposes. The most common usages of the start-job are as follows:

- It controls the binding process of the newly created object that has to be bound into another object.
- It is used for XGEN sources: the XGEN generate phase and compilation phase can be handled by a separate start-job.

Notice that the functionality of start-jobs is not limited to the above-mentioned common usages. A site can define its own start-job procedures according to its own needs.

A driver relation identifies an object in which a newly compiled object has to be bound. It is possible that a program has multiple driver relations. In that case, the program will be bound into multiple driver-objects after the program is compiled.

The start-job and driver relations are dependent on each other. A program that has driver relations must be bound into driver-objects and these binding processes are controlled by start-jobs.

Start-jobs are normally started directly after the correct compilation of the object. It is also possible to define start-jobs that are started when all compilations are complete. This last function is normally used in case of binding procedures: multiple objects are recompiled and bound simultaneously into a driver object when all compilations are complete.

24.2.(Binding) START-JOB and DRIVER Relations  
Examples

The following example shows two programs, PROG/AA and PROG/BB that have to be bound into the driver object PROG/DRIVER.

Example 1

PROG/AA has to be bound into the driver object PROG/DRIVER directly after the program is compiled.

The following relations are defined.

Entity Owner:Class(Asset)	Reason
FILE   PROG / AA : DRIVER ( PROG / DRIVER )	WFL/BIND/NOW will bind the object of PROG/AA
FILE   PROG / AA : START-JOB ( WFL / BIND / NOW )	into driver object PROG/DRIVER.

**Example 2**

PROG/AA and PROG/BB have to be bound into the driver object PROG/DRIVER at the end of all compilations.

The following relations are defined.

**Entity|Owner:Class(Asset)**

FILE | PROG/AA:DRIVER (PROG/DRIVER)

FILE | PROG/BB:DRIVER (PROG/DRIVER)

FILE | PROG/DRIVER:START-  
JOB (WFL/BIND/TOTAL)

**Reason**

WFL/BIND/TOTAL binds all  
objects into

driver PROG/DRIVER at the end  
of all compilations.

### 24.2.1. (Binding) Which Start-Job Will Be Used and When Is It Started?

A start-job can be connected to a driver or to a compiled source. Which start-job will be started (the one linked to the driver or the one linked to the source)?

#### **If RESPECT/SURE/COMPILE is Started with the Parameter ALWAYS:**

- The source-start-job is used. Driver relations are ignored in this mode.

#### **If RESPECT/SURE/COMPILE is Started With the Parameter TOTAL-BIND and Not With ALWAYS:**

- If the compiled source does not have any driver relations but has a start-job relation, then this source-start-job is used.
- If the compiled source has one or more driver relations, then the driver-start-jobs are used. If one of these drivers does not have start-jobs, then the source-start-jobs are used too.

#### **If RESPECT/SURE/COMPILE is Not Started With the Parameters ALWAYS or TOTAL-BIND:**

- If the compiled source does not have any driver relations but has a start-job relation, then this source-start-job is used.
- If the compiled source has one or more driver relations but it does not have a start-job relation, then all driver-start-jobs are used.
- If the compiled source has one or more driver relations and has a start-job relation, then this source-start-job is used.

If the start-job of a compiled source is used, then this start-job will be started immediately after the compilation.

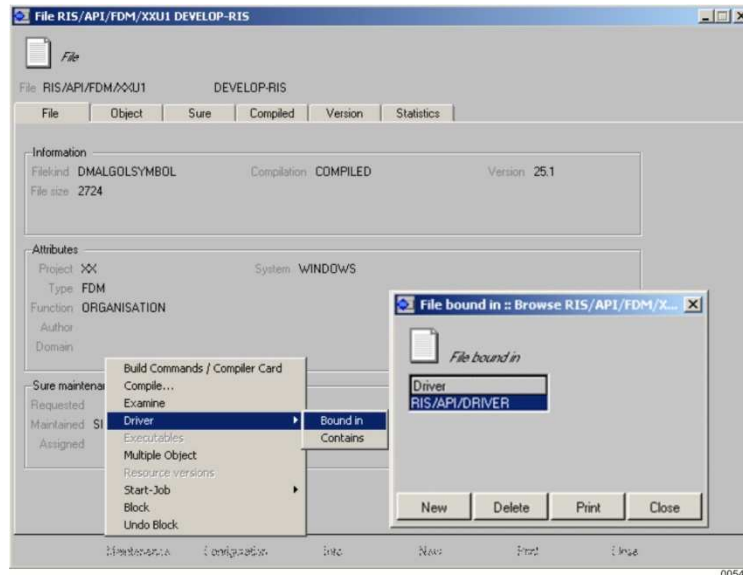
If the start-job of the driver is used, then this start-job is started when all compilations are complete.



### 24.2.2. (Binding) Maintaining DRIVER Relations

To link a driver to a source through <source-properties>, perform the following steps:

1. Click **Configuration**.
2. Click **Driver**.
3. Click **Bound in**.



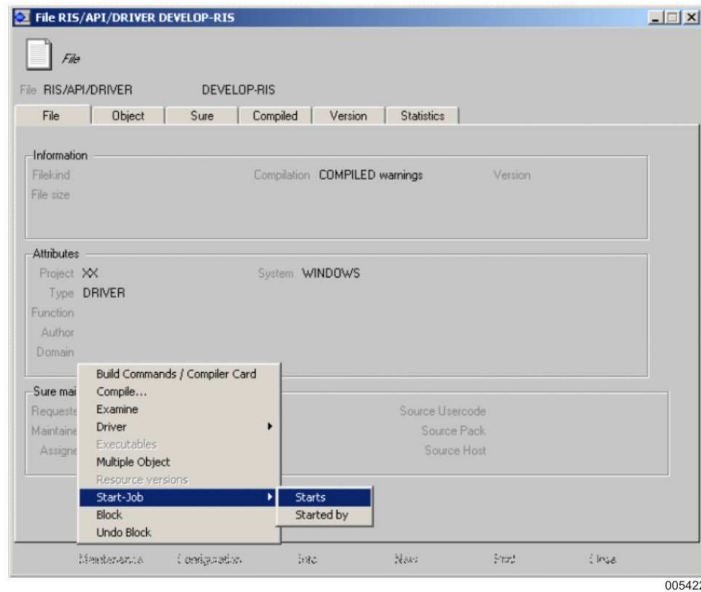
To link a source to a driver through <driver-properties>, perform the following steps:

1. Click **Configuration**.
2. Click **Driver**.
3. Click **Contains**.

### 24.2.3. (Binding) Maintaining Start-Job Relations

To link a start-job through <source-properties>:

1. Click **Configuration**.
2. Click **Start-Job**.
3. Click **Starts**.



To link a source to a start-job through <startjob-properties >:

1. Click **Configuration**.
2. Click **Start-Job**.
3. Click **StartedBy**.





## Section 25

# **(Examine) Extracting Source Information**

The sources in the repository contain the actual up-to-date information about the structure and interrelations of the written software. The SURE examine process scans the application software for language statements that refer to other files (such as copy files, libraries, ADDS, dataset, and data files). The information is stored in the repository to build a data dictionary at file level. Other online and batch processes of SURE (such as the compile process) use this data dictionary, but it is also possible that programmers use this dictionary when they make queries.

It is obvious that every item in a source can be found using the FIND function that scans a group of sources for a find target. The main difference between the results of a FIND function and the structured repository information is that the use of queries can only be applied to structured repository information. A SURE query can contain AND and OR clauses, the returned information is up-to-date and the results are returned fast (in seconds). Refer to Section 13, "Query Facility," for more information.

The data dictionary information is extracted and stored by the RESPECT/SURE/EXAMINE program. This program runs in two modes:

- Mode 1: Examine runs in the SURE batch environment and perform a total examine.

The intention of this mode is to create a data dictionary that describes the run-time application environment as good as possible. Sources that are compiled by SURE can be translated by a compilation-translate table, and they can contain dollar records that control the compiler (\$ OMIT records and \$IF, \$ELSE, \$ENDIF records in COBOL85). Therefore, the EXAMINE process translates and filters the source in exactly the same way as the compilation process does, using the same set of compiler dollar options. The second step is that the source and all its copy files are merged to one file. This file is input for the examine process.

The translate tables and dollar options can differ for each environment, and for this reason the examine output may differ in the various repository environments.

See "Examine as a Result of Transfer" in Section 16.

- Mode 2: Examine runs during the day and examine each saved file for its copy file relations.

The copy file relations are used by the RESPECT/SURE/COMPILE program that uses these relations to extract copy files and maintain application system integrity. The RESPECT/SURE/COMPILE program runs prior to the RESPECT/SURE/EXAMINE program, and this creates a chicken and egg situation. To optimize the use of system resources, this quick examine is implemented.

A “RESPECT/SURE/EXAMINE during the day” is started automatically in the background as soon as it is necessary to examine a file. If the examine program runs in this mode, all “old” copy file relations are not removed. The result of this can be that the source has too many copy file relations, because the relations to copy files that are deleted from the source do still exist. Notice that this is necessary, because the source is NOT examined in “merge-mode” (for performance reasons) and otherwise nested copy file relations (= copy files that are defined in another copy file) will be lost.

The copy file relations are important for the integrity mechanism in the RESPECT/SURE/COMPILE. This mechanism will produce correct results if there are too many copy file relations, but it can give production run-time errors if some copy file relations are missing. The changed source will be examined again during the evening batch, and at that moment, the correct set of copy file relations is linked to the source.

Every changed file will be given a relation `EXAMINE-STATUS( TO-EXAMINE )` to put that file into the examine queue. The examine queue is handled by the RESPECT/SURE/EXAMINE program. The actual examining of the files is performed in the SURE batch environment. For this reason, there is a user interface command that allows examining the files directly.

It is possible to define that a file must not be examined. This can be done for each file through:

1. Click **<file-properties>**.
2. Click the **SURE** tab.
3. Click **Skip Examine** for each file-type options.

### 25.1.(Examine) The Windows Menu Selection

To examine a single file using the `<filename>` function:

1. Right-click the **<filename>** function.
2. Click **Miscellaneous**.
3. Click **Examine**.

This function runs the RESPECT/SURE/EXAMINE the for the selected file. Note that the examining happens directly but asynchronously from the user interface. It may take some minutes before the examine process is done.

It is possible to define that a file must not be examined. This can be done for each file through:

1. Click **<file-properties>**.
2. Click the **SURE** tab.
3. Click **Skip Examine** or **per file type** (file type options).

### 25.1.1. (Examine) Recognized Statements by the Examining Process

	cobol	algol	pascal	fortran	dasdl	job	xgen	cc	delphi	RIS	opal	binder	progress
Library call	x	x	x										
import library procedure	x	x											
export library procedure		x											
copy file	x	x	x	x	x	x	x	x		x			x
Database	x	x					x						
update (open DB)	x	x											
data set	x	x			x		x						
data base items (option)	x												
Set	x	x			x		x						
Logical DB + Remap					x								
Execute						x							
External	x		x										
Public			x										
internal data file name	x	x											
external data file name	x	x											
transaction base	x	x											
Format							x						
format-file							x						
Object							x						
System							x						
Reports							x						
adds-dictionary	x	x											
adds-program	x	x											
adds-file	x	x											
adds-group	x	x											
Uses/Unit									x				
Bind, Binder, Host						x						x	
Define Situation											x		
Define OdtSequence											x		
Define Display											x		
Define Memo											x		

25.1.2. (Examine) Relation Modification

The examine process is controlled using relations.

The following relation is added when a file is changed in an environment (using check-in, load, or transfer to a next environment):

```
FILE-CONTROL | <filename> : EXAMINE-STATUS ( TO-EXAMINE )
```

The RESPECT/SURE/EXAMINE selects all sources with relation EXAMINE-STATUS ( TO-EXAMINE ) .

The following relations are modified when the file is examined:

```
Add          FILE-CONTROL | <filename> : EXAMINE-STATUS ( EXAMINED )
```

```
Delete       FILE-CONTROL | <filename> : EXAMINE-STATUS ( TO-EXAMINE )
```

25.2.(Examine) RESPECT/SURE/EXAMINE

The RESPECT/SURE/EXAMINE program runs during the SURE batch.

Input parameters :

<file>

MERGE

TOTAL

SAVE

COPYONLY

ALGOL-INTERNAL

See "RESPECT/SURE/EXAMINE" in Section 47.

Task string : <environment>.

Task value : 20 = Examine only for copy file statements.

Example

```
RUN RESPECT/SURE/EXAMINE( " " ); TASKSTRING = "PROD"
```

Where

This program runs in the SURE evening batch. The program can be started from the file-property-screen using the function EXAMINE function.

Security

```
MP +PU
```

<file>	This parameter can be used to examine a single file instead of all files. If a <filename> is not passed then option MERGE is true.
MERGE or TOTAL	A temporary merge file is created using the OMIT options in the source, the SURE compilation translate tables, and the copy file statements. This temporary merge file will be examined. The result of the merge option is that the examine process has the same input as the compilation process and the examine-result (= relations in the repository) is a correct reflection of the object environment.



COPYONLY	The examine is only done for copy file statements. Old copy file relations are not removed.
SAVE	The temporary merge file is not removed after the examine. MERGE is in this situation is always true.
ALGOL-INTERNAL	If this option is set, then internal data file names of ALGOL programs will always be examined. If this option is not set, then internal file names of data files in ALGOL programs will only be examined if an external file name is found.

## 25.3.Allow to Define a Site-Specific Examine Function

It is possible to extend the default examine process with a site-specific examine process. This is solved using a new entry point in the SURE site library.

The source is first examined in the usual way as described in the previous paragraphs. If a site-specific examine process is established, then the source is passed to that function after the default examine is done. The results of both processes are combined and modified in SURE.

The site-examine function must be defined in the SURE site library as follows:

```
PROCEDURE SITE_EXAMINE
    (D,FILENAME,FILETYPE,PROJECT,D_FILEKIND,REC_LEN,REC_OFFS,EA_RSP);
VALUE  FILENAME,FILETYPE,PROJECT,D_FILEKIND,REC_LEN,REC_OFFS;
STRING FILENAME,FILETYPE,PROJECT;INTEGER D_FILEKIND,REC_LEN,REC_OFFS;
EBCDIC ARRAY EA_RSP[0];
FILE D;
```

### Input values:

D	The input file that must be scanned. The file is opened and rewound.
FILENAME	The name of the input file.
FILETYPE	The filetype of the input file in SURE.
PROJECT	The project of the input files in SURE.
D_FILEKIND	The file kind of the input file.
REC_LEN	Record length (only the source part, no seqnr or markid).
REC_OFFS	Offset of the record.

**Return value:**

EA\_RSP                    An ebcdic array with scanned results that must be updated in the repository. The following response lines are valid:

```
RESET <class>;  
CLEAR <class>;  
ADDITEMREF <class> <itemname>;  
ADDFILEREF <class> <filename>;  
ADDFILERELATION <entity> <owner> <class> <asset>;  
ADDFILERELATION <entity> <owner> <class> <asset>;  
DELRELASSET <entity> <owner> <class>;  
DELRELOWNER <entity> <class> <asset>;
```

Each response line must be terminated with a semicolon.

**RESET <Class>**

- This command removes the old references of this class that are not used anymore (at the end of the examine process).
- <Class> can be any name less than 30 characters without intermediate spaces. If it does not exist in SURE, then it will be added.
- This command is only relevant for classes that are not maintained by the default examine process. See “Names of Classes Used By the SURE Software” in Section 13 for a list of classes that are maintained by the default Examine process. Default examine-classes cannot be initialized using the RESET function.

**CLEAR <Class>**

- This command removes all references of this class immediately.
- <Class> can be any name less than 30 characters without intermediate spaces. If it does not exist in SURE, then it will be added.
- This command can be used for all classes. If it is a default examine class, then it overrides the results of the default examine process.

**ADDITEMREF <Class> <Itemname>**

- This command adds a new reference for this class to an item-name.
- <Class> can be any name less than 30 characters without intermediate spaces. If it does not exist in SURE, then it will be added.
- <Class> must be initialized previously using the RESET or CLEAR command.
- <Itemname> can be any name less than 30 characters. If it does not exist in SURE, then it will be added as an item-name.

**ADDFILEREFS <Class> <Filename>**

- This command adds a new reference for this class to a file name.
- <Class> can be any name less than 30 characters without intermediate spaces. If it does not exist in SURE, then it will be added.
- <Class> must be initialized previously using the RESET or CLEAR command.
- <Filename> can be any name, but should be a file. If it does not exist in SURE, then it will be added as a file name.

**ADDFILERELATION <Entity> <Owner> <Class> <Asset>**

- This adds a reference from a file name to the input file or from the input file to a file name.
- <Entity> must be FILE or FILE-CONTROL.
- The <owner> or the <asset> must be "THIS," meaning the current input-file. The other one is a file name. If that file name does not exist in SURE, then it will be added.
- <Class> can be any name less than 30 characters without intermediate spaces. If it does not exist in SURE, then it will be added.

**ADDITEMRELATION <Entity> <Owner> <Class> <Asset>**

- This adds a reference from an itemname to the input file or from the input file to an itemname.
- <Entity> must be FILE or FILE-CONTROL.
- The <owner> or the <asset> must be "THIS," meaning the current input-file. The other one is an itemname. If that itemname does not exist in SURE, then it will be added.
- <Class> can be any name less than 30 characters without intermediate spaces. If it does not exist in SURE, then it will be added.

**DELRELASSET <Entity> <Owner> <class>**

- Remove all relations with this <owner> and <class> (references of this <owner>).
- <Entity> must be FILE or FILE-CONTROL.
- The <owner> can be "THIS," meaning the current input-file, an existing file name, or an existing itemname.
- <Class> can be any name less than 30 characters without intermediate spaces. If it does not exist in SURE, then it will be added.

### **DELRELOWNER <Entity> <Class> <Asset>**

- Remove all relations with this <class> and <asset> (references to this <asset>).
- <Entity> must be FILE or FILE-CONTROL.
- The <asset> can be "THIS," meaning the current input-file, an existing file name, or an existing item name.
- <Class> can be any name less than 30 characters without intermediate spaces. If it does not exist in SURE, then it will be added.

The RESPECT/LIBRARY/SITE\_FUNCTIONS source contains a full example of the usage of this new feature.

The site-examine function requires the following extra entry in RESPECT/TITLES

```
SITE-FUNCTION = EXAMINE
```

## Section 26

# Backup of Maintenance Sources

Dumping the repository will back up all information in the INFDB database, including all source files that are stored in the repository

The maintenance copies of the sources (the checked out sources) are not secured through this procedure. The RESPECT/SURE/SECURE program copies these maintenance copies to tape or to another pack.

The RESPECT/SURE/SECURE selects all files with the following relation:

```
FILE-CONTROL | <filename> : SOURCE-USERCODE ( <usercode> )
```

All selected files are copied from the workspace in CANDE to a backup-tape or to another pack.

### 26.1. Backup of Non-Repository Files

You can define a specific file type to add CANDE files to the “secured” list. The file type must have the following options set:

- Enter only a file name
- Never use object-relation

Through this method, you can also backup the files that are not stored in the repository.

### 26.2.RESPECT/SURE/SECURE

The RESPECT/SURE/SECURE program creates and zips a job, which copies all checked out sources and secured files to tape. This program runs in all the environments.

Run the RESPECT/SURE/SECURE as follows:

Input	Task string	:	This program does the secure for all environments.
	Task value	:	0 = The secured sources are copied to tape. 1 = The secured sources are copied to a dump family.
Output	Job	:	SURE/SECURE/SOURCES. This job is started automatically and copies the sources to a tape or to a backup family.
Example	RUN RESPECT/SURE/SECURE;VALUE = 1;		
Where	In the SURE evening batch.		
Security	MP +PU (the program accesses files in other directories).		

If the **Secure pack** field (on the Global Options screen, on the **Sure** tab) is entered, then the checked out sources are copied to this pack instead of using tapes. This option requires you to run the RESPECT/SURE/SECURE with task value = 1. The files are then copied as (SURESAVE)<environment>/<usercode>/<filename> on <secure-pack>.

The SURE/SECURE/SOURCES is generated. By default, this job contains a copy statement. The "Dump option" field (in the **Global Options** dialog box) might extend this copy as "copy & compare" or "copy & catalog." To achieve these extensions, enter "&compare" or "&catalog" in the **Dump option** field.

## Section 27

# Maintenance of Development Disk Space

### 27.1. Cleanup Disks

#### Function

In order to control disk space on a development system, it is necessary to remove unused files from the development disks. The RESPECT/SURE/CLEANER program is used for disks cleaning. The cleaning may be executed for each pack and host where development takes place. The clean program reads the disk directory, and all sources and data files matching the restrictions (named as defined in the following paragraph) are placed on tape (= CLEAN/<pack>) and optionally removed from disk. All unnecessary files can be removed from packs using this program.

The RESPECT/SURE/CLEANER can be executed at various time intervals (for example, daily, weekly). It is advised to save at least two versions of "clean" tapes to retrieve removed files.

The LASTACCESSDATE file attribute is used to check the "not used" status of a file and the FILEKIND file attribute is used to check the type of the file.

This program runs in all the environments.

### 27.2.RESPECT/SURE/CLEANER

The RESPECT/SURE/CLEANER ("`<directory>`") program generates a job "`SURE/CLEAN/<PACK>`" with the "copy" and the "remove" statements for the files to be affected. This generated job starts automatically. To define which files have to be handled, complete the parameter `<directory>`. To determine which options are to be used, one of the following task values has to be used.

Run the RESPECT/SURE/CLEANER as follows:

Input	Parameter	:	—<disk directory>—————
			See "RESPECT/SURE/CLEANER" in Section 47.
	Task value	:	0 = Copy the selected files to tape and remove them from disk. 1 = Copy the selected files to tape. 2 = Remove the selected files from disk.
Example	RUN RESPECT/SURE/CLEANER( "USERCODE ON PK1" );VALUE=0		
Where	This program can be used by the customer in self-written jobs.		
Security	MP +PU (the program does SYSTEMSTATUS commands).		
	<directory> examples		
	USERCODE ON PACK1		all files under any usercode on pack "PACK1" are selected.
	USERCODE/SURE ON PACK1		all files with usercode "(SURE)=" on pack "PACK1" are selected.
	USERCODE//DEBUG ON PK1		all files under any usercode with directory "DEBUG" on pack "PACK1" are selected.
	*DEBUG ON PACK1		all files in the directory "*DEBUG/=" on pack "PACK1" are selected.
	(SURE)DEBUG ON PACK1		all files in the directory "(SURE)DEBUG/=" on "PACK1" are selected.

The RESPECT/SURE/CLEANER program verifies that a file is not in use in any SURE environment.

### Restrictions

- Files in maintenance under <usercode>, <pack name> will not be removed.
- Files with "secured" status will not be removed.
- Job symbol and object files will not be removed.
- Symbol files that have been used within the last 20 days will not be removed.
- Data or seqdata files that have been used within the last 30 days will not be removed.

### Options

- To identify a usercode-directory on a family as "not-to-be-cleaned," do the following:
  1. Click **Environment**.
  2. Click **Global Options**.
  3. Click the **Sure cleaner** tab.



- To change the cleaning periods, do the following:
  1. Click **Environment**.
  2. Click **Global Options**.
  3. Click the **History** tab.



## Section 28

# SUMLOG Information

SURE reads the SUMLOG and loads the information to the repository.

The following information is loaded for each program that is known to SURE:

- From the beginning of task (BOT) records of the SUMLOG
  - The compile time of the running object is checked with the compile timestamp in the repository.
  - The Usercode that runs the program.
  - The FileVersion of the source at the moment the object was compiled.
- From the end of task (EOT) records of the SUMLOG
  - The EOT date and time of the program run.
  - The job or task numbers.
  - The system serial number of the host at which this run took place.
  - The number of the SUMLOG file that is now loaded.
  - The processor time used.
  - The IO time used.
  - The ReadyQ time
  - The elapsed time.
  - The amount of memory used.
  - The EOT status: OK for Normal EOT, DS for abnormal EOT, or STX for syntax errors.
  - The number of Lines printed with this run of the program.
  - The amount of Initial Pbits.Text

To access the loaded SUMLOG information, select **Statistics** from the <Filename> function drop-down menu.

Generally, only the SUMLOG information of the production run-time environment is loaded in SURE.

The following figure illustrates the SUMLOG information.

Statistics - Browse-RESPECT/SURE/COMPILE

Print

Close

Timestamp	Username	Mems	Lnn	Host	Object Created	File Version	Processor	I/O	Reads	Elapsed	Memory	Err	Lnn	Ina Pgs
11-2-2009 18:28:50	SIMON	477/00432	49	2222	13-6-2008 22:07:04	241.1			00:00:21	74.354	Ok	53	1059	
11-2-2009 18:59:57	SIMON	418/00432	46	2222	13-6-2008 22:07:04	241.1			00:00:21	74.330	Ok	53	1062	
27-6-2007 22:08:08	DEVELOP_RIS	297/04003	1172	1111	18-6-2007 22:00:53	237.2	00:00:02		00:00:24	65.616	Ok	46	2388	
26-6-2007 22:08:25	DEVELOP_RIS	1129/02920	1172	1111	18-6-2007 22:00:53	237.2	00:00:02		00:00:24	68.808	Ok	47	2384	
25-6-2007 22:08:57	DEVELOP_RIS	1830/0143	1171	1111	18-6-2007 22:00:53	237.2	00:00:02		00:00:23	69.367	Ok	47	2451	
24-6-2007 22:08:57	DEVELOP_RIS	9767/00183	1170	1111	18-6-2007 22:00:53	237.2	00:00:01		00:00:23	69.450	Ok	47	2431	
23-6-2007 22:08:37	DEVELOP_RIS	9487/00781	1170	1111	18-6-2007 22:00:53	237.2	00:00:01		00:00:23	69.395	Ok	47	2452	
6-11-2006 22:01:28	DEVELOP_RIS	6850/07574	1004	1111	30-10-2006 22:00:35	232.1	00:00:04	00:00:02	00:01:14	95.190	Ok	149	4431	
5-11-2006 22:01:25	DEVELOP_RIS	6561/06864	1004	1111	30-10-2006 22:00:35	232.1	00:00:03	00:00:02	00:01:09	94.601	Ok	149	4475	
30-10-2006 15:02:32	DEVELOP_RIS	615/01131	998	1111	30-10-2006 14:16:41	???	00:00:01		00:00:21	72.176	Ok	48	852	
30-10-2006 15:01:26	DEVELOP_RIS	615/01129	998	1111	30-10-2006 14:16:41	???			00:00:04	63.769	Fds	18	491	
30-10-2006 15:02:32	DEVELOP_RIS	615/01131	998	1111	30-10-2006 14:16:41	???	00:00:01		00:00:21	72.176	Ok	48	852	
30-10-2006 15:01:26	DEVELOP_RIS	615/01129	998	1111	30-10-2006 14:16:41	???			00:00:04	63.769	Fds	18	491	
12-3-2003 22:01:15		4785/05623	25	1111	5-3-2003 22:01:19	185.1	00:00:04	00:00:01	00:00:54	63.111	Ok	80	895	
12-3-2003 9:04:08		3485/04324	25	1111	5-3-2003 22:01:19	185.1	00:00:03	00:00:04	00:00:55	64.684	Ok	48	821	
10-3-2003 22:01:22		2775/03488	24	1111	5-3-2003 22:01:19	185.1	00:00:05	00:00:01	00:00:58	63.095	Ok	136	939	
9-3-2003 22:01:54		2637/02755	24	1111	5-3-2003 22:01:19	185.1	00:00:10	00:00:01	00:01:13	68.903	Ok	260	995	

Print

Close

00542

## 28.1. RESPECT/SURE/LOG

To load the SUMLOG information to the repository, it is necessary to run the RESPECT/SURE/LOG program, with a file equation of the FILE LOG for the SUMLOG file to be loaded. The current SUMLOG can be closed with ODT-command TL. After this command, a SUMLOG file is available on disk with the name "SUMLOG/<system serial number>/mmddyy/<SUMLOG version number>".

### Example

Consider a SUMLOG file named SUMLOG/32/091393/000315 ON LOGPACK.

Load this SUMLOG file as follows in the SURE environment PROD.

```
RUN OBJECT/RESPECT/SURE/LOG;
FILE LOG = SUMLOG/32/091393/000315 ON LOGPACK;
TASKSTRING = "PROD";
```

An extra RESPECT/SURE/STARTLOG program is available that performs a directory search for a specified SUMLOG directory and then generates and zips a job that runs the log-loader (RESPECT/SURE/LOG) for all the SUMLOG files in the directory. Refer to the following paragraphs for details.

If multiple production host systems are present, this procedure should be performed for the SUMLOG files of each host. It is necessary to copy the files to the host on which SURE runs. The run data of all production hosts should be stored in the RESPECT repository to keep the repository up-to-date.

The log-loader (RESPECT/SURE/LOG) checks that the correct SUMLOG file is offered.

- The log-loader registers for each host the number of the last SUMLOG file that is loaded. It is not possible to load a SUMLOG file with a number less than or equal to the last correctly loaded SUMLOG file. Therefore, it is not possible to load a SUMLOG file twice; old SUMLOG files are skipped by the log-loader.
- If the current SUMLOG file is not the expected SUMLOG file (in case there is a gap in the SUMLOG files that are offered to the log-loader), the following message will appear:

"Log sequence for HOST <system-serial number> not correct, SURE expects: <number>; current log is <number>".

The operator has to react to this message with <mix number>AX OK or <mix number>DS NONE.

- Runinfo that is spread over multiple SUMLOG files will be loaded correctly into the repository. The log-loader keeps a table with runinfo BOJ/t-records that need to be completed by a corresponding end of job (EOJ)/EOT-record. This table is stored in the repository when a SUMLOG -check is completed and reloaded from the repository when a new SUMLOG -check is started.

The log-loader produces several (control) listings as follows:

- A list of abnormal ended (DS-ed) programs.
- A list of backup file names with the number of printed lines by this program run.
- A list of compilation mismatches: programs with a compilation timestamp in SURE that differs from the creation timestamp of the object file. This overview reports the object-files that run with an invalid object date. You can also prevent objects with an invalid creation date from running.

See "RESPECT/LIBRARY/SUPERVISOR" in Section 47 for details.

- A list of object names that are not known in SURE.

Files without SURE compilation timestamp

If a file is known to SURE with an object-location, then the log-loader will try to add run information to the file. However, if the compile-date from SURE and the compile-date from the log differ, the file name will be placed on the list of mismatches. If a file does not have a SURE compile-date (because it was not compiled through SURE), it will still appear on that list, with a note that no compile-date is available. This can be the case for

- Programs that were newly entered in SURE and were not yet compiled, but an old version is still running in production.
- Programs that were entered with the SECURE function.
- Programs that were manually entered and given object-relations.

The last two options could be legitimate ways of getting run information for the programs involved (that is, programs from a package like SURE for which no sources are available). If the log-loader is started with task value = 1, 11, 21, or 31 (task value mod 10 = 1), then these programs are suppressed from the mismatch list, and the log-loader will then adjust the SURE-compile-date to the log-compile-date. Thereafter, the file will no longer appear in the mismatch list. It is reported if the log-loader appended the SURE-compile-date.

Run the RESPECT/SURE/LOG as follows:

Input	Task string	:	<environment>.		
	target	:	1 = debug mode.		
	Task value	:	value DIV 10 = 0	:	Create overviews a and c.
		:	value DIV 10 = 1	:	Create overviews a, b and c.
		:	value DIV 10 = 2	:	Create overviews a, c and d.
		:	value DIV 10 = 3	:	Create overviews a, b, c and d.
		:	value MOD 10 = 1	:	Initialize the compile-timestamp for files without a compile-timestamp.
		:	value = 99	:	Filter the SUMLOG file: read-only the beginning of job (BOJ), BOT, EOJ and EOT records and write them in a filtered SUMLOG file.
	file	:	SUMLOG file (file equate FILE LOG = <SUMLOG/...).		
	accept	:	Only if a non-expected SUMLOG file is passed to the program.		
Output	overview	:	4 different overviews.		
Example	RUN RESPECT/SURE/LOG;TASKSTRING = "PROD";VALUE = 31; FILE LOG = *SUMLOG//1111/020403/000003 ON DLPACK;				
Where	In the SURE evening batch.				
Security	MP +PU (the program accesses files in other directories).				

## 28.2. Selecting Runinfo from the SUMLOG

The log-loader (RESPECT/SURE/LOG program) can be started for any SURE-environment to load the SUMLOG information for that specific environment. On the other hand, the production environment is obviously the most relevant place to load the SUMLOG information.

The log-loader checks the following on each SUMLOG record:

- The type of the SUMLOG record.
- The runinfo environment
- The object name
- The object-location

### Check the Type of Sumlog Record

The log-loader reads the SUMLOG file and selects all BOT, BOJ, EOT, and EOJ records from that SUMLOG file. All other SUMLOG records are skipped.

### Check the Runinfo Environment

It is possible to define a runinfo environment. Objects that are NOT started from a usercode in the runinfo environment are skipped by the log-loader. This feature is handy if objects are placed under directory "\*".

A runinfo environment must be defined through dialog:

1. Click **<environment>**.
2. Click **Properties**.
3. Click the **LoadSumlog** tab.

Through this function, it is possible to add usercodes to a runinfo environment or to delete usercodes from a runinfo environment. If no runinfo environment is defined (the default situation), then all CANDE usercodes are target for the log-loader.

### Example

Suppose that OBJECT-USERCODE of all objects is "\*".

Two identical application systems are active on the host. Both of the systems are using the same set of objects (placed in the directory "\*"). Application system AA runs for customer AA under Cande usercode AA. Application system BB runs for customer BB under Cande usercode BB.

The run information of application system AA has to be logged, and the information of system BB can be skipped.

In this situation, usercode AA has to be added to the runinfo environment. Objects that are started from any other usercodes will then be ignored by the log-loader.

### Check the Object Name

Object names that are not known in SURE are written in an overview and skipped.

### Check the Object-Location in SURE

The physical location of the object or job on disk must be equal to the object-location of the source in SURE that is defined in the environment for which the log-loader is running.

### Example

Suppose that the RESPECT/SURE/LOG program has to load SUMLOG information for the PRODUCTION environment.

The PRG/AAA program has in PRODUCTION the following object-location: usercode (PROD) on PK1.

The SUMLOG contains two BOT records for the PRG/AAA program.

The first BOT record refers to the object (TST)OBJECT/PRG/AAA on TPACK. This SUMLOG record is skipped because the object-location does not match with the declaration of the object-location in SURE (in the environment PROD).

The second BOT record refers to the object (PROD)OBJECT/PRG/AAA ON PK1. This record will be selected.

Sometimes, the defined object-location in SURE is not the final place where the object is actually placed on disk (for example, if the compiled object is copied to the predefined object-location by the deployment procedure of SURE, and a site-specific job copies the object at a controlled moment from the SURE-object-location to the runtime-object-location). In this case, the SURE-object-location is not equal to the runtime-object-location, and the object will always be skipped by the log-loader.

This problem can be solved in two ways:

- For an individual file: Define the SUMLOG location for a file (the SUMLOG-run-usercode/pack) through the <file-properties> function and click the **Sure** tab. The object will be selected by the log-loader if the SUMLOG-location is equal to the runtime-object-location.
- For a group of files: Define the runtime-usercode as a synonym of the SURE-object-usercode, and the runtime-pack as a synonym of the SURE-object-pack. The objects are selected by the log-loader if the runtime-object-location is equal to the SURE-object-location or to a synonym of the SURE-object-location.

A synonym usercode or pack must be defined through one of the following relations:

```
<environment>*SYNONYM|<SURE-object-usercode>:RUN-INFORMATION(<runtime-usercode>)  
<environment>*SYNONYM|<SURE-object-pack>:RUN-INFORMATION(<runtime-pack>)
```

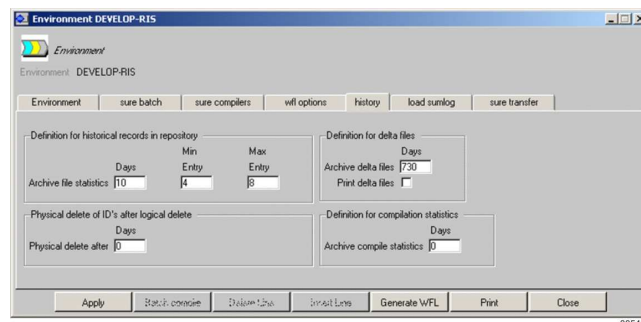
## 28.3.Cleaning the Runinfo

By default, the runinfo records that are stored in the repository will be kept forever. This is not an ideal situation that at the end will lead to limit errors. Therefore, the user has to define his cleaning criteria.

The runinfo can be loaded for each repository environment, and therefore, the “runinfo cleaning criteria” must be defined for each environment, as follows:

1. Click **<environment>**
2. Click **Properties**.
3. Click the **History** tab.

The following figure illustrates the runinfo clean options: Days = 10; Min = 4; Max = 8.





A continuation screen is presented where it is possible to define all kinds of clean options. On this screen, the cleaning criteria for runinfo are in the block "Definition for historical records in the repository."

The following runinfo clean options are available for each environment:

- days: The amount of days that runinfo has to be kept. If nothing is entered, then the runinfo will be kept forever. If a 0 (zero) is entered, then it will not be kept at all.
- min: Keep at least <min> lines with runinfo. This overrides the <days> option. This option is handy for objects that run very incidental (once a year).
- Max: Keep at the most <max> lines with runinfo. This overrides the <days> option. This option is handy for objects that run very often (multiple times on a day).

28.4. RESPECT/SURE/STARTLOG

The RESPECT/SURE/STARTLOG program performs a directory search for a specified SUMLOG directory and generates and zips a job that runs the log-loader (RESPECT/SURE/LOG) for all the SUMLOG files in the directory.

Run the RESPECT/SURE/STARTLOG as follows:

Input	Parameters	: —<directory with sumlog files> — — FILTER — — PREFIX <prefix> —
		See "RESPECT/SURE/STARTLOG" in Section 47 for details.
	Task string	: <environment>. The task string is passed to the log-loader RESPECT/SURE/LOG
	Task value	: The task value is passed to the log-loader.
Example	RUN RESPECT/SURE/STARTLOG( " *SUMLOG ON LOGPACK" ); TASKSTRING = "PROD"	
Where	This program can run in a user-written job.	
Security	MP +PU (the program accesses files in other directories).	
FILTER	The filter function reads the SUMLOG files on the production host and creates filtered SUMLOG files that contain all the required records. The size of such a filtered SUMLOG file is five to ten percent of the size of the original SUMLOG file.	
PREFIX <prefix>	The default prefix of the filtered SUMLOG files is "FILTERED/". This option overrules the default prefix.	

Details and Considerations

The RESPECT/SURE/STARTLOG program automates the process of loading the SUMLOG files. The program can be added to the SURE batch job. It is especially useful if SUMLOGs from other hosts also need to be processed. These SUMLOG files have to be copied to the host where SURE is running. This can also be done as part of a daily procedure.

The procedure is illustrated by the following examples:

1. To process all available SUMLOGs on logpack

```
RUN OBJECT/RESPECT/SURE/STARTLOG(" *SUMLOG ON LOGPACK")
```

2. To process all SUMLOGs from host 3478 and host 1111

```
RUN OBJECT/RESPECT/SURE/STARTLOG(" *SUMLOG/3478 ON LOGPACK")  
RUN OBJECT/RESPECT/SURE/STARTLOG(" *SUMLOG/1111 ON LOGPACK")
```

In both cases, a job is generated running the log-loader as described previously, for as many logs as are available. This avoids manual procedures to create jobs running the log program.

Notice that the log-loader skips the SUMLOG files that are already loaded. This makes it easier to automate the load procedure.

The name of the generated job is SURE/STARTLOG.

### Example

```
RUN RESPECT/SURE/STARTLOG(" *SUMLOG ON DLPACK");VALUE=30;
```

Job SURE/STARTLOG:

```
10000001BEGIN JOB STARTLOG;  
10000003FILE FREADY (TITLE=SURE/STARTLOG/READY,KIND=DISK,NEWFILE=TRUE);  
20000031RUN *OBJECT/RESPECT/SURE/LOG ON IDRD;  
20000032    VALUE = 30;  
20000033    FILE LOG    = *SUMLOG/1111/020403/000003 ON DLPACK;  
20000034    FILE RESPECT = (SIMON)RESPECT/TITLES ON IDRD;  
20000041RUN *OBJECT/RESPECT/SURE/LOG ON IDRD;  
20000042    VALUE = 30;  
20000043    FILE LOG    = *SUMLOG/1111/020403/000004 ON DLPACK;  
20000044    FILE RESPECT = (SIMON)RESPECT/TITLES ON IDRD;  
99999980OPEN(FREADY);  
99999981LOCK(FREADY);  
99999983END JOB;
```

### Synchronize Your Daily Batch with the SURE/STARTLOG/READY File:

The RESPECT/SURE/STARTLOG program reads an input-directory with SUMLOG-files, and then generates and zips a SURE/STARTLOG job. This job runs the log-loader for each SUMLOG-file in the input-directory. At the end of the job (when all SUMLOG-files are processed), the SURE/STARTLOG/READY file is created.

This feature can be used by the user in a site-specific job:

```
REMOVE SURE/STARTLOG/READY;  
RUN  OBJECT/RESPECT/SURE/STARTLOG("<input directory>");  
WAIT FILE SURE/STARTLOG/READY IS RESIDENT;  
<etc.>
```

### FILTER Option

It happens often that the production run-time environment is not on the same host as the development environment with SURE. In those cases, the SUMLOGs have to be file-transferred (with BNA) from the production host to the development host where log-loader can load them. This BNA file transfer may take a long time.

The filter function reads the SUMLOG files on the production host and creates filtered SUMLOG files that contain all the required records. The size of such a filtered SUMLOG file is five to ten percent of the size of the original SUMLOG file, which improves the performance of the BNA file transfer.

### Example

Consider the following directory with SUMLOG files (on a production host):

```
FILES *SUMLOG ON PRINTPK:S
```

FileName	Kind	Records	Sectors	CreationTime
*SUMLOG/4523/122902/009602	LOG	57751	57751	12/28/2002
*SUMLOG/4523/123002/009603	LOG	60496	60496	12/29/2002
*SUMLOG/4523/123002/009604	LOG	93143	93143	12/30/2002
*SUMLOG/4523/123102/009605	LOG	13505	13505	12/30/2002

Filter the SUMLOG files as follows:

```
RUN RESPECT/SURE/STARTLOG("FILTER *SUMLOG ON PRINTPK");
```

```
FILES *FILTERED ON PRINTPK:S
```

FileName	Kind	Records	Sectors	CreationTime
*FILTERED/SUMLOG/4523/122902/009602	DATA	2963	3000	01/08/2003
*FILTERED/SUMLOG/4523/123002/009603	DATA	2863	2900	01/08/2003
*FILTERED/SUMLOG/4523/123002/009604	DATA	4128	4200	01/08/2003
*FILTERED/SUMLOG/4523/123102/009605	DATA	1995	2000	01/08/2003

Load the filtered SUMLOG files as follows in SURE:

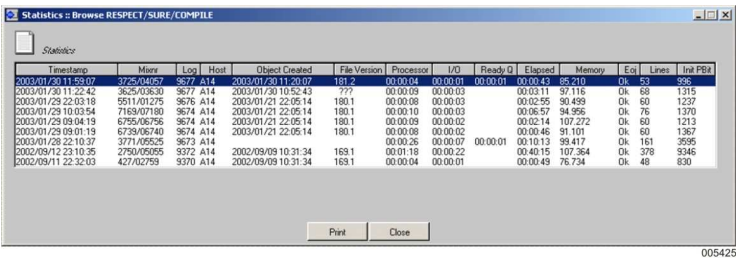
```
RUN RESPECT/SURE/STARTLOG("*FILTERED ON PRINTPK");
```

The filter option requires that the programs RESPECT/SURE/LOG and RESPECT/SURE/STARTLOG are placed on the production host because these programs filter the SUMLOG files. These programs do not use database INFDB when they are in filter mode, so it is not necessary to put an INFDB database on the production host.

# 28.5.Analyzing the SURE Run Information

## 28.5.1. Run Information of an Individual File

To analyze the runinfo of an individual file, select **Statistics** from the **<Filename>** function drop-down menu.



Timestamp	More	Log	Host	Object Created	File Version	Processor	I/O	Ready/D	Elapsed	Memory	Eq	Lines	Inst P81
2003/01/20 11:59:07	3725/04057	9677	A14	2003/01/20 11:20:07	181.2	00:00:04	00:00:01	00:00:01	00:00:43	85,210	Ok	53	996
2003/01/20 11:22:42	3625/03630	9677	A14	2003/01/20 10:52:43	777	00:00:09	00:00:03		00:02:11	57,116	Ok	68	1315
2003/01/23 22:03:19	9511/01275	9676	A14	2003/01/21 22:05:14	180.1	00:00:09	00:00:03		00:02:55	90,459	Ok	60	1227
2003/01/29 10:03:54	7169/07180	9674	A14	2003/01/21 22:05:14	180.1	00:00:10	00:00:03		00:06:57	94,956	Ok	76	1370
2003/01/29 09:04:19	6795/06795	9674	A14	2003/01/21 22:05:14	180.1	00:00:09	00:00:02		00:02:14	107,272	Ok	60	1213
2003/01/29 09:01:19	6739/06740	9674	A14	2003/01/21 22:05:14	180.1	00:00:09	00:00:02		00:00:46	91,101	Ok	60	1367
2003/01/28 22:10:37	3771/09525	9673	A14			00:00:26	00:00:07	00:00:01	00:10:13	98,417	Ok	161	3595
2002/09/12 23:10:35	2790/09055	9372	A14	2002/09/09 10:31:34	169.1	00:01:18	00:00:22		00:40:15	107,364	Ok	378	9346
2002/09/11 22:32:03	427/02759	9370	A14	2002/09/09 10:31:34	169.1	00:00:04	00:00:01		00:00:49	76,734	Ok	48	830

## 28.5.2. Run Information of a Group of Files

The run information of a selection of files can be analyzed through the RESPECT/PRINT program.

The following options are available:

- RUNINFO-LASTRUN

This function lists all programs that did not run since a date.
- RUNINFO-OVERVIEW

This function lists all run information. The overview also contains graphics about the system resources that were used by the selected programs. This overview may be helpful to discover programs that use many of the systems resources.
- RUNINFO-CHECKRUN

This function shows how many times programs have run (for each month).
- NO-RUNINFO

This overview shows programs without run-information (programs that never ran in the operational environment). This overview can be used as a help to clean up the repository.

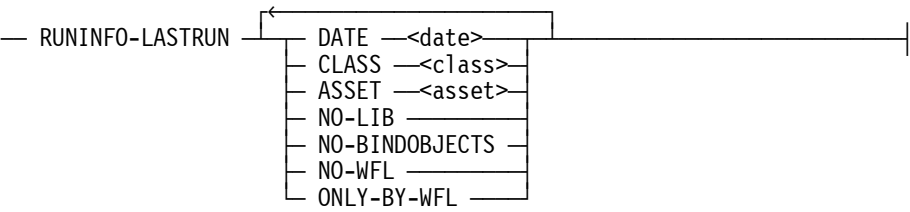
### 28.5.2.1. RUNINFO-LASTRUN

RUNINFO-LASTRUN lists all the programs that did not run since a date.

The complete railroad diagram for this option is

```
RUN RESPECT/PRINT("<parameter>");TASKSTRING = <environment>
```

<parameter> =



DATE	The selection date. This selects all the runinfo records that did not run since the date.
CLASS <class>	Select all files with a relation with class=<class> or asset=<asset>.
ASSET<asset>	If this option is not used then all files are selected.
NO-LIB	Skip the files that are known in SURE as a library.
NO-BINDOBJECTS	Skip the files that have to be bound into a driver.
NO-WFL	Skip the files that are known in SURE as a job.
ONLY-BY-WFL	Select only the files that are known in SURE as "executed in a job."

Example Output

```
Programs that did not run since 20020101    DEVELOP-RIS    page 1    20030314

Program                                         Last run
date

RESPECT/CONVERT
RESPECT/LOADDB
RESPECT/MAKE/HELP
RESPECT/MAKE/HTML                            2002-07-28
RESPECT/MAKE/XML
RESPECT/REPLACER
RESPECT/RESYDE/CHANGEDITEMS
RESPECT/RESYDE/CHKDBITEMS
End of list
```

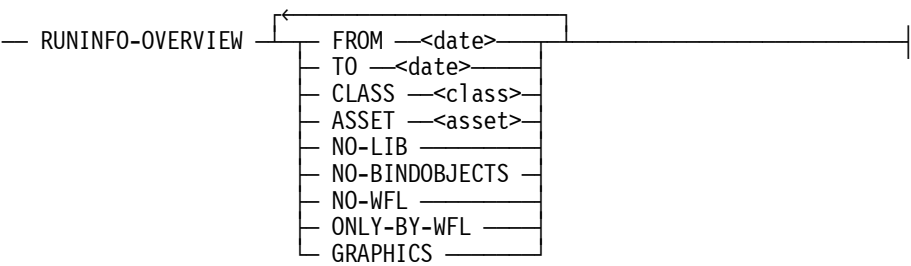
28.5.2.2. RUNINFO-OVERVIEW

RUNINFO-OVERVIEW lists all the run information of a period for the selected files. This function can be used to analyze the performance of a period.

The complete railroad diagram for this option is

```
RUN RESPECT/PRINT("<parameter>");TASKSTRING = <environment>
```

<parameter> =



## SUMLOG Information

FROM <date> TO <date>	The selection period. This selects the runinfo within the period. If option GRAPHICS is used, then the period must be one day.
CLASS <class>	Select all files with a relation with class=<class> or
ASSET <asset>	asset=<asset>.
	If this option is not used then all files are selected.
NO-LIB	Skip the files that are known in SURE as a library.
NO-BINDOBJECTS	Skip the files that have to be bound into a driver.
NO-WFL	Skip the files that are known in SURE as a job.
ONLY-BY-WFL	Select only the files that are known in SURE as "executed in a job."

### Example Output:

Run information from 20030305 to 20030305      DEVELOP-RIS   page 1      20030314												
Run date	Job/Task	Log/Host	EOT	Proctime	IO time	ReadyQ	Elapsed	InitPbit	Memory	Lines	Date	Compiled Version
RESPECT/SURE/COMPILE												
22:22:31	8567/ 9797	22/1111	OK	0:07		0:02	1:08	846	72171	235	20030305	22:01:19 185.1
22:20:42	8568/ 9787	22/1111	OK	0:08		0:03	1:10	941	69775	235	20030305	22:01:19 185.1
22:00:27	8566/ 9753	22/1111	OK	0:18	0:01	0:08	13:05	1619	93277	276	20030305	10:11:39 ???
11:06:04	9116/ 9119	22/1111	OK	0:12		0:03	1:12	932	69326	127	20030305	10:11:39 ???
10:29:20	9054/ 9056	21/1111	OK	0:31	0:05	0:27	7:10	1545	94183	77	20030305	10:11:39 ???
RESPECT/SURE/COPY												
11:22:33	9133/ 9142	22/1111	OK	0:01			0:03	265	63947		20030207	22:59:16 12.1
11:22:30	9133/ 9141	22/1111	OK	0:01			0:02	265	63247		20030207	22:59:16 12.1
11:22:24	9133/ 9140	22/1111	OK	0:02	0:01		0:06	265	66054		20030207	22:59:16 12.1
11:21:50	9133/ 9139	22/1111	OK	0:10	0:03	0:05	0:34	371	67386		20030207	22:59:16 12.1
11:21:47	9133/ 9138	22/1111	OK	0:01			0:03	302	37839		20030207	22:59:16 12.1
RESPECT/SURE/EXAMINE												
22:23:51	8567/ 9804	22/1111	OK	0:01			0:02	249	28249		20030207	23:09:00 126.1
22:22:06	8568/ 9794	22/1111	OK	0:01			0:02	249	28913		20030207	23:09:00 126.1
22:13:52	8566/ 9782	22/1111	OK	3:48	0:10	0:32	6:24	2028	83051		20030207	23:09:00 126.1
11:07:25	9116/ 9126	22/1111	OK	0:01			0:02	249	27401		20030207	23:09:00 126.1
10:42:03	9054/ 9082	22/1111	OK	2:02	0:03	2:36	6:44	7896	80690		20030207	23:09:00 126.1
RESPECT/SURE/FIND												
22:23:46	8567/ 9802	22/1111	OK	0:01			0:02	248	26058		20030207	23:12:37 43.2
22:22:02	8568/ 9792	22/1111	OK	0:01			0:02	248	29096		20030207	23:12:37 43.2
22:13:46	8566/ 9780	22/1111	OK	0:01			0:03	248	27584		20030207	23:12:37 43.2
11:07:28	9116/ 9127	22/1111	OK	0:01			0:02	248	29011		20030207	23:12:37 43.2
Worst programs, sorted on processor time.												
FileName	Processor	%	per run	runs	Proctime	IO time	ReadyQ	Elapsed	InitPbit			
RESPECT/SURE/EXAMINE	5:53	37,6	1:10	5	5:53	0:13	3:08	13:14	10671			
RESPECT/REPOSITORY	3:19	21,2	0:33	6	3:19	0:04	0:51	10:29	2081			
RESPECT/SURE/COMPILE	1:16	8,1	0:15	5	1:16	0:06	0:43	23:45	5883			

## SUMLOG Information

RIS/BACKGROUND/IMPACT	1:15	8,0	0:37	2	1:15	0:00	5:27	27:38:04	2961
RESPECT/SURE/MATCH	0:51	5,4	0:10	5	0:51	0:02	3:05	5:24	1875
RIS/BACKGROUND/EXAMINE	0:38	4,0	0:19	2	0:38	0:02	1:55	1:24:00	2651
RIS/GENERATE/ALL	0:24	2,5	0:04	5	0:24	0:00	0:05	1:44	10223
RIS/BACKGROUND/OBJLOC	0:19	2,0	0:19	1	0:19	0:00	1:00	1:19:16	789
RESPECT/SURE/COPY	0:15	1,6	0:03	5	0:15	0:04	0:05	0:48	1468
RIS/COMPLETE/OBJECT	0:09	0,9	0:01	7	0:09	0:00	0:04	0:30	2400
RIS/GENERATE/BIND	0:09	0,9	0:02	4	0:09	0:00	0:10	0:40	1635
RESPECT/SURE/TRANSFER	0:08	0,8	0:01	5	0:08	0:00	0:10	0:45	2377
RESPECT/SURE/REPLACE	0:04	0,4	0:01	4	0:04	0:00	0:00	0:10	1052
RESPECT/SURE/FIND	0:04	0,4	0:01	4	0:04	0:00	0:00	0:09	992
RESPECT/SURE/FINISH	0:03	0,3	0:03	1	0:03	0:00	0:00	0:09	372
RESPECT/TASK/TRANSFER	0:03	0,3	0:00	4	0:03	0:00	0:01	0:09	1011

-----  
Total 15:37 0:33 19:05 33:47:19 55432

Worst programs; sorted on elapsed time

FileName	Elapsed	%	per run	runs	Proctime	IO time	ReadyQ	Elapsed	InitPbit
RIS/BACKGROUND/IMPACT	27:38:04	81,7	13:49:02	2	1:15	0:00	5:27	27:38:04	2961
RIS/MENU	2:27:33	7,2	16:23	9	0:45	0:02	2:16	2:27:33	5496
RIS/BACKGROUND/EXAMINE	1:24:00	4,1	42:00	2	0:38	0:02	1:55	1:24:00	2651
RIS/BACKGROUND/OBJLOC	1:19:16	3,9	1:19:16	1	0:19	0:00	1:00	1:19:16	789
RESPECT/SURE/COMPILE	23:45	1,1	4:45	5	1:16	0:06	0:43	23:45	5883
RESPECT/SURE/EXAMINE	13:14	0,6	2:38	5	5:53	0:13	3:08	13:14	10671
RESPECT/REPOSITORY	10:29	0,5	1:44	6	3:19	0:04	0:51	10:29	2081
RESPECT/SURE/MATCH	5:24	0,2	1:04	5	0:51	0:02	3:05	5:24	1875
RIS/GENERATE/ALL	1:44	0,0	0:20	5	0:24	0:00	0:05	1:44	10223
RESPECT/SURE/COPY	0:48	0,0	0:09	5	0:15	0:04	0:05	0:48	1468
RESPECT/SURE/TRANSFER	0:45	0,0	0:09	5	0:08	0:00	0:10	0:45	2377
RIS/GENERATE/BIND	0:40	0,0	0:10	4	0:09	0:00	0:10	0:40	1635
RIS/COMPLETE/OBJECT	0:30	0,0	0:04	7	0:09	0:00	0:04	0:30	2400
RESPECT/TOOLS	0:23	0,0	0:05	4	0:02	0:00	0:04	0:23	236
RESPECT/SURE/REPLACE	0:10	0,0	0:02	4	0:04	0:00	0:00	0:10	1052
RESPECT/TASK/TRANSFER	0:09	0,0	0:02	4	0:03	0:00	0:01	0:09	1011
RESPECT/SURE/FINISH	0:09	0,0	0:09	1	0:03	0:00	0:00	0:09	372
RESPECT/SURE/FIND	0:09	0,0	0:02	4	0:04	0:00	0:00	0:09	992
RIS/PUTLINE	0:04	0,0	0:00	14	0:00	0:00	0:00	0:04	937
RIS/REFRESH	0:03	0,0	0:00	6	0:00	0:00	0:01	0:03	322

-----  
Total 15:37 0:33 19:05 33:47:19 55432

Time	Parallel processes	Processor	Readyq
9:00:00	0- 0	0- 0  *	0- 0  *
	0- 0	0- 0  *	0- 0  *
	0- 1  *	0- 6  *+	0- 3  *+
	0- 0	0- 0  *	0- 0  *
	0- 0	0- 0  *	0- 0  *
10:00:00	0- 0	0- 0  *	0- 0  *

```

0- 1|*          0- 26|*++++          0- 0|*
0- 4|****       0- 24|+****          0- 14|++*
2- 3|++*        8- 47|----+*++++     10-102|-----+*+++++
2- 3|++*        1- 32|-*+*++++       4- 43|---+*++++
11:00:00 2- 4|++* 1- 86|-*+*+++++     4- 47|---+*++++
1- 2|+*         0- 84|+*+++++        0- 58|+*+++++
1- 1|+          0- 1|*              0- 1|*
1- 1|+          0- 1|*              0- 1|*
1- 1|+          0- 1|*              0- 1|*
12:00:00 1- 1|+ 0- 1|*              0- 1|*
1- 1|+          0- 1|*              0- 1|*
1- 1|+          0- 1|*              0- 1|*
1- 1|+          0- 1|*              0- 1|*
1- 1|+          0- 1|*              0- 1|*
1- 1|+          0- 1|*              0- 1|*

```

End of list

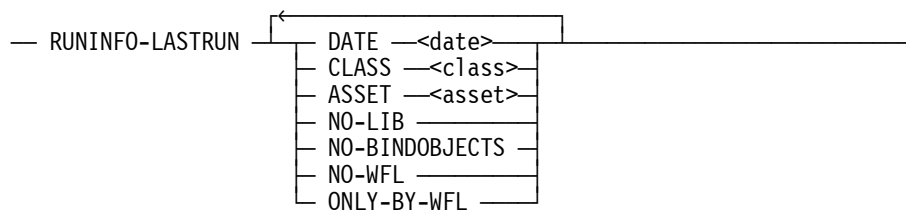
### 28.5.2.3. RUNINFO-CHECKRUN

This function shows how many times the programs have run (for each month).

The complete railroad diagram for this option is

```
RUN RESPECT/PRINT("<parameter>");TASKSTRING = <environment>
```

<parameter> =



DATE	The selection date. This selects all the runinfo records that did not run since the date.
CLASS <class>	Select all files with a relation with class=<class> or asset=<asset>.
ASSET <asset>	If this option is not used then all files are selected.
NO-LIB	Skip the files that are known in SURE as a library.
NO-BINDOBJECTS	Skip the files that have to be bound into a driver.
NO-WFL	Skip the files that are known in SURE as a job.
ONLY-BY-WFL	Select only the files that are known in SURE as "executed in a job."



```

Run info cumulation since 20020101                                DEVELOP-RIS   page 1      20030314
                2003 - > < - - - - 2002 - - - - > < - - - - 2001 - - - - > < 2000
Filename                m f j d n o s a j j m a m f j d n o s a j j m a m f j d n >

RESPECT/PRINT                1
RESPECT/SURE/BATCH           2
RESPECT/SURE/COMPILE        25 63 7           2
RESPECT/SURE/COPY            5 6
RESPECT/SURE/EXAMINE        24 60 6           2
RESPECT/SURE/FIND           26 62 5
RESPECT/SURE/FINISH          1 9 1           1
RESPECT/SURE/LOG              4
RESPECT/SURE/MATCH           24 60 5           2
RESPECT/SURE/REPLACE         23 60 5
RESPECT/SURE/SECURE           2
RESPECT/SURE/STARTLOG         1
RESPECT/SURE/TRANSFER        25 62 7           2
RESPECT/TASK/ADD_TRF_QUEUE    1
RESPECT/TASK/LIST             1
RESPECT/TASK/TRANSFER        23 61 4

End of list

```

This function selects programs that do not have run information (that never ran in the operational environment). This overview can be used as a help to clean up the repository.

```
RUN RESPECT/PRINT( "<parameter>" );TASKSTRING = <environment>
```

```

— NO-RUNINFO —┐
                  │ CLASS —<class>
                  │ ASSET —<asset>
                  │ NO-LIB —
                  │ NO-BINDOBJECTS —
                  │ NO-WFL —
                  │ ONI Y-RY-WFI —

```

ONLY-BY-WFL      Select only the files that are known in SURE as "executed in a job."

Run-information is retrieved from the system/SUMLOG by the RESPECT/SURE/LOG program and loaded in SURE.

A file without run information can mean that

- The file never ran in the operational environment.
- The system/SUMLOG is not analyzed by SURE and was never loaded in the repository.

## 28.6.Check the Creation Date of an Object

RESPECT/LIBRARY/SUPERVISOR is an interface library for Metalogic Supervisor that makes it possible to check the creation date and time of an object file with the compiled-timestamp in SURE.

Metalogic Supervisor has a function to call this library for each object that is started (on the production host). If the creation date/time of the object (on disk) is not equal to the compilation timestamp in SURE, then a warning or error is given. Notice that you require the most recent version of Metalogic Supervisor for this function. Contact Metalogic for more information about the Supervisor.

### Technical Details and Considerations

The RESPECT/LIBRARY/SUPERVISOR has the following two functions:

- Exports the FND\_REL\_TIME procedure, which is called by Metalogic Supervisor.
- Accesses the SURE repository to return the compilation timestamp.

The SURE repository is always placed on the development mainframe. The production run-time environment may be on another mainframe, or on the same mainframe as SURE. Both cases are supported through extra options in RESPECT/TITLES.

### If the Production Run-Time Environment is on Another Host as SURE

In this case, the RESPECT/LIBRARY/SUPERVISOR must be resident on both hosts. Therefore, the object must be copied from the development host to the production host, where it must be visible for Metalogic Supervisor. The copy on the development host must be resident in the same directory as the other SURE objects (such as OBJECT/RESPECT/LIBRARY).

The copy on the **production** host requires a RESPECT/TITLES file with (at least) the following options:

- SURE-HOST = <hostname>
- SURE-SUPERVISOR-PORT = <port-number>

The copy on the **development** host requires a RESPECT/TITLES file with (at least) the following options:

- SURE-SUPERVISOR-PORT = <port-number>
- INFDB-LOCATION = <location INFDB control file>
- OBJECT-LOCATION = <location SURE objects>
- No option SURE-HOST

Metalogic Supervisor calls the FND\_REL\_TIME procedure in the RESPECT/LIBRARY/SUPERVISOR on the production host. This production-copy passes the call through a port-file to the development-copy of the RESPECT/LIBRARY/SUPERVISOR. The development-copy does the access to the repository and returns the timestamp through the port-file.

The RESPECT/LIBRARY/SUPERVISOR reads RESPECT/TITLES during its initialization phase.

- If the SURE-HOST option is found, then the library will not open the repository, but a port-file (indicated by the SURE-SUPERVISOR-PORT option).
- If the SURE-HOST option is not found, then the library opens the repository. If the SURE-SUPERVISOR-PORT option is found, then it opens that port and starts waiting for input through that port to the Supervisor.

### **If the Production Run-time Environment is on the Same Host as SURE**

In this case, the RESPECT/LIBRARY/SUPERVISOR must be resident in the same directory as the other SURE objects (such as OBJECT/RESPECT/LIBRARY).

Metalogic Supervisor calls the FND\_REL\_TIME procedure in the RESPECT/LIBRARY/SUPERVISOR. This library accesses the repository and returns the compilation timestamp to the Supervisor.

The RESPECT/LIBRARY/SUPERVISOR reads RESPECT/TITLES during its initialization phase.

- If the SURE-HOST option is found, then the library will not open the repository, but a port-file (indicated by the SURE-SUPERVISOR-PORT option).
- If the SURE-HOST option is not found, then the library opens the repository. If the SURE-SUPERVISOR-PORT option is found, then it opens that port and starts waiting for input through that port; otherwise, the input is obtained through the call on FND\_REL\_TIME.

In this case, the options SURE-HOST and SURE-SUPERVISOR-PORT must not be defined in RESPECT/TITLES.

### **Select the Correct Environment in the SURE Repository**

The usercode where RESPECT/LIBRARY/SUPERVISOR is started (on the development host) determines the SURE environment. This usercode must be defined in SURE and linked to the SURE environment that must be used to check the compilation timestamps.

### Example

- Consider a repository with three environments: DEVELOP, TEST, and PRODUCTION.
- The production run-time objects are created from the PRODUCTION environment, so that the environment must be used to check the creation timestamps of those objects.
- Create a new usercode PRODSURE, define that usercode in SURE with "default environment" = PRODUCTION.
- Start the RESPECT/LIBRARY/SUPERVISOR under PRODSURE to assign it to the PRODUCTION environment.

### Automatic Start/Stop of RESPECT/LIBRARY/SUPERVISOR

The following option in RESPECT/TITLES makes it possible to run the RESPECT/LIBRARY/SUPERVISOR under the control of RESPECT/LIBRARY:

```
SUPERVISOR-LIBRARY-USERCODE = <usercode>
```

If RESPECT/LIBRARY starts and a supervisor-usercode is defined in RESPECT/TITLES, then the RESPECT/LIBRARY/SUPERVISOR is started too. If RESPECT/LIBRARY thaws, then the RESPECT/LIBRARY/SUPERVISOR is thawed too. The Supervisor library will be started under the given usercode, which assigns the library to the SURE environment of that usercode (see above).

This option is handy when the production run-time host is not the same as the development host, or when you want to close the entire repository frequently for offline dumps.

### Summarized

The following options in RESPECT/TITLES control RESPECT/LIBRARY/SUPERVISOR

- SURE-SUPERVISOR-PORT: Indicates an internal communication port-number if the production run-time host is not the same as the development host.
- SURE-HOST: If this option is found, then the library will not open the repository, but a port-file (indicated by option SURE-SUPERVISOR-PORT).
- SUPERVISOR-LIBRARY-USERCODE: Starts or stops the Supervisor library automatically when RESPECT/LIBRARY is started or stopped. Start the Supervisor library under the given usercode.

## Section 29

# Software Delivery

This function is available only in the terminal emulation interface.

The SURE/delivery option supports two different modes:

- Deliver specifications from one repository to another repository
- Deliver source files as an application system

Setting the SURE system option <deliver mechanism works in SURE style> will select delivery of source files.

### 29.1. Deliver Specifications from One Repository to Another Repository

The SURE software supports dump/load procedures between two repositories using a TASK or a RELEASE indication. On the Delivery screen, it is possible to add a single task or all tasks of a release to the delivery queue.

The SYSTEM definition for an environment contains a CURRENT RELEASE definition. If this field is set, each task gets the SOLVED-IN-RELEASE(<Current-Release>) relation as the task is transferred to the solved status. Delivery of a release will queue all tasks with the SOLVED-IN-RELEASE(<release>) attribute in the transfer queue.

Delivery can be issued for a SOLVED task from each environment that is identified as the "solved" environment.

#### Dump Procedure

```
RUN RESPECT/REPOSITORY ("DUMP-REPOSITORY")  
RUN RIS/EDITOR; VALUE=9996;
```

Both of these programs create files in the directory RESPECT/DUMP/=

#### Load Procedure

```
RUN RESPECT/REPOSITORY ("LOAD-REPOSITORY")  
IF RESPECT/DUMP/SCREENS/TRANSFER IS RESIDENT THEN  
    RUN RESPECT/REPOSITORY ("LOAD-FORMATS TRANSFER") ;
```

Note that the RESPECT/DUMP/= directory may contain hundreds of files including the source files to be loaded.

### 29.2.Deliver Source Files as an Application System

The SURE option <deliver option in SURE style> changes the delivery mode. This mode delivers source files with a license key from the repository.

On the Delivery screen, it is possible to add a single task or an entire application system to the delivery queue.

The DELIVER button starts the RESPECT/SURE/DELIVERY/40 program, which creates the delivery job. This program sets the license-key for each selected file (all files of a system or task) and starts a job to copy these files to tape.

The output of this program consists of:

SURE/DELIVERY	A job that starts and copies all files to tape.
SURE/DELIVER/INFO	A file that contains the names of all delivered files.
SURE/DELIVERY/IMPACT/ANALYSIS	A file that lists file names, projects, and versions.

## Section 30

# Initial Load Files

A special batch-program has been developed to load directories with MCP files into the database. The program is called RESPECT/SURE/LOAD.

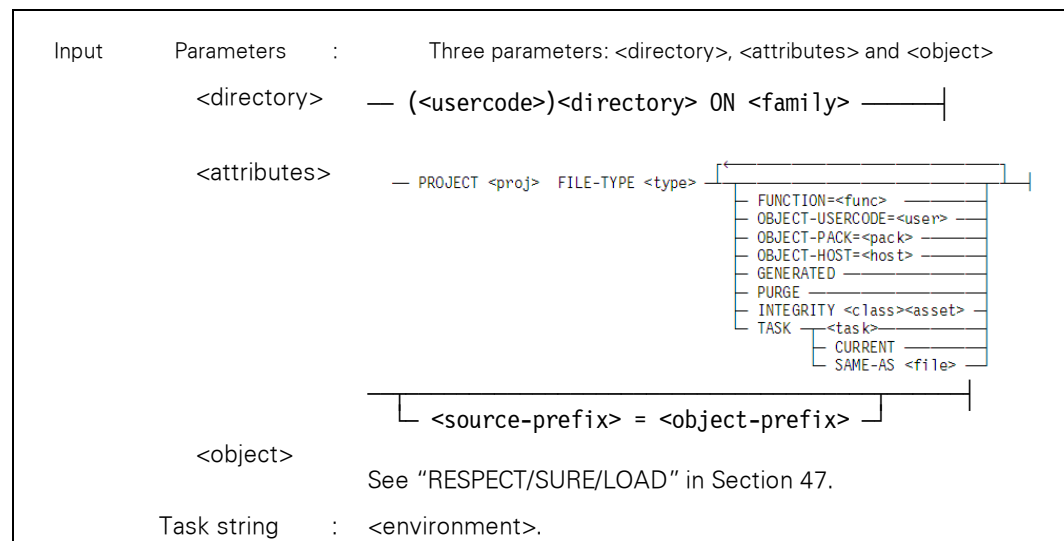
This process can be controlled from the SURE Explorer interface on the PC; however, at initial load where multiple directories are loaded, a WFL job can be more suitable. An easy way to create your own load job is as follows:

1. Start the first load-process from the GUI-interface.
2. This creates a job on the ClearPath server under the work-directory called WFL/<your-usercode>. You can see this job in the active-mix using ODT command "MX USER <usercode>".
3. Get this job and use statement "RUN RESPECT/SURE/LOAD" as an example for your own job.

### 30.1. RESPECT/SURE/LOAD

The RESPECT/SURE/LOAD program is used to load directories with sources (MCP-applications) to SURE, and must be started with three parameters:

```
RUN OBJECT/RESPECT/SURE/LOAD( "<param-1>", "<param-2>",
    "<param-3>" );VALUE = <taskvalue>;
```



Task value	: 1 = Load only new files. 2 = Reload only existing files and keep the attributes. 3 = Reload only existing files and modify the attributes. 4 = Load all files and keep attributes of existing files. 5 = Load all files and modify attributes of existing files.
Example	<code>RUN RESPECT/SURE/LOAD( "(SG)SRC ON PK1", "PROJECT=AA, FILE-TYPE=BATCH", "" ); TASKSTRING = "DEVELOPMENT"; VALUE=1</code>
Where	This program can be used by the customer in self-written jobs.
Security	MP +PU (the program uses SYSTEMSTATUS commands).
<directory>	The directory with files that have to be loaded. Only source-files will be loaded into the database. Object-files, data-files and dbdata-files that are resident in the specified directory will be skipped. The syntax is as follows:  <div> <div>USERCODE ON &lt;pack&gt;</div> <div>Select and load all files under any usercode (not *) on family &lt;pack&gt;.</div> </div> <div> <div>(&lt;uscd&gt;)&lt;dir&gt; ON &lt;pack&gt;</div> <div>Select and load all files in directory (&lt;uscd&gt;)&lt;dir&gt; ON &lt;pack&gt;.</div> </div>
PROJECT	Each loaded file gets these attributes. <Project> and <file-type> are required attributes. If <object-usercode> and <object-pack> are not entered, then they are inherited from the default object-location of the application system using the object-inheritance-option of the file-type.
FILE-TYPE	
FUNCTION	
OBJECT-USERCODE	
OBJECT-PACK	
OBJECT-HOST	
GENERATED	If this option is used, then each loaded file gets <code>STATUS( GENERATED )</code> ; otherwise, the status is set to the environment-name.  It is not possible to transfer a generated file to a next environment, because the generation is usually dependent on the current contents of the environment.
PURGE	Remove a source from disk after it was loaded in SURE.
TASK = <task>	The task is linked to each loaded file.
TASK = CURRENT	The current task of the usercode, where RESPECT/SURE/LOAD is started is linked to each loaded file.
TASK = SAME-AS <file>	Each loaded source gets the same tasks as the SAME-AS-<file>. In addition, the impact relations, the integrity-relations and the integrity-prod relations are inherited from that file. This makes it possible to load a generated file using a start-job during the batch-compile, with combination of parameters FIRST-FILE-TYPE or ALWAYS-MODE for RESPECT/SURE/COMPILE.
<source=object>	Specifies a non-standard object name for a group of files. By default, the object of source A will be titled OBJECT/A. This is done automatically by SURE. It is possible to specify a non-



standard object name for a file. However, this can only be done by using an on-line function where the non-standard object name has to be specified for each individual file. This can be a tedious task. With this option, it is possible to load the non-standard object names for all files in the specified directory.

**Note:** *This is not the recommended method. A preferred way to define non-standard object names for a group of file is through a library that determines a site-specific standard for object names. See "Object Names" in Section 23.*

Examples with source-name SOURCE/ABC:

<b>&lt;source/object&gt; conversion</b>	<b>Object name</b>
"SOURCE=OBJECT"	"OBJECT/ABC"
"SOU=X/O"	"X/ORCE/ABC"
"SOURCE="	"ABC"
"=OBJ/ "	"OBJ/SOURCE/ABC"
" "	"OBJECT/SOURCE/ABC"

#### **Notes:**

- When a file has a standard object name, the object will be titled "OBJECT/"<filename>.
- When a file has a non-standard object name that does not start with "\$", the object will be titled "OBJECT"<deviating object name>.
- When a file has a non-standard object name that starts with "\$", the object will be titled: <dev.obj.name>.

A preferred way to define non-standard object names for a group of files is through a library that determines a site-specific standard for object names. See "Object Names" in Section 23.

#### **Task Value**

The load program can skip files that are already loaded in SURE or it can load all files of the directory. The Task value controls the load mode.

- Task value : 1 = Load only new files.
- 2 = Reload only existing files and keep the attributes.
- 3 = Reload only existing files and modify the attributes.
- 4 = Load all files and keep attributes of existing files.
- 5 = Load all files and modify attributes of existing files.

### Error Report

The program creates an error report of the load. In most error situations, the file is still loaded and the error is only a warning.

In the following cases a file will not be loaded:

- The file name does not exist in SURE, and only existing files are to be loaded.
- The file name exists already in SURE, and only new files are to be loaded.
- The file is checked out by a user.

## 30.2. LOAD using GUI Interface

### 30.2.1. Load MCP Sources in SURE

Loading a directory from a ClearPath server to the SURE repository can be done in two ways:

1. Open the **Unisys CP/NX <hostname>** folder.
2. Open the **Sources <usercode>** folder and do one of the following:
  - Right-click the directory that you want to load and select **Load in SURE function**.
  - Right-click the directory that you want to load and click **Tools** menu, and then click **CP/NX environment**, and click **Load directory in SURE**.

Note that this option is greyed if no ClearPath server directory was selected.
3. Click **OK**.

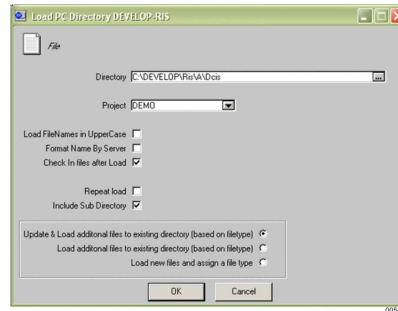
A workflow is started that run the RESPECT/SURE/LOAD program. Only one copy of the LOAD program should be active at a given time.

### 30.2.2. Load PC Directory of Files in SURE

To load a PC directory in SURE through the menu selection:

1. Click **Tools**.
2. Click **PC environment**.
3. Click **Load directory of files in SURE**.

This gives the following screen.



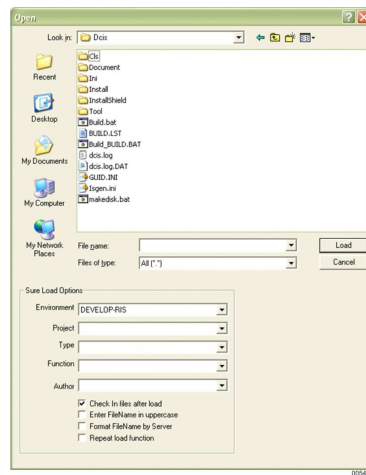
Directory	Browse to the directory with files that you want to load. It must be a sub-directory of the local work directory.
Project	Each loaded file gets this project.
Load FileNames in Uppercase	The file name is in uppercase.
Format Names By Server	Each file-node that is entirely in uppercase or lowercase will be translated to a starting uppercase and the rest of the node in lowercase. If options "Load FileNames in Uppercase" and "Format Names By Server" are both off, then no upper or lower translation is done on the file name.
Repeat load	If enabled, the loadscreen remains open to start a new load.
Update and Load additional files	To repeat a load for an existing directory. Additional files and changed files are loaded.
Load additional files	To repeat a load for an existing directory. Only additional files are loaded.
Load new files	To load a new directory.

### 30.2.3. Load files in SURE

To load an individual PC file in SURE through menu selection:

1. Click **Tools**.
2. Click **PC environment**.
3. Click **Load PC files in SURE**.

This gives the following screen.



File name	You can select up to five files to be loaded.
File of type	This is a filter to show only the files with this extension. "All" shows all files.
Project	Each loaded file gets this project.
Type	Each loaded file gets this file-type.
Function	Each loaded file gets this function.
Author	Each loaded file gets this author.
Check in after load	If disabled, the file is loaded in SURE, but it remains in checked-out state. So, you can continue working on this source. If enabled, the file is loaded and checked in.
Load FileNames in Uppercase	The file name is in uppercase.
Format Names By Server	Each file-node that is entirely in uppercase or lowercase is translated to a starting uppercase and the rest lowercase. If options "Load FileNames in Uppercase" and "Format Names By Server" are both off, then no upper or lower translation is done on the file name.
Repeat Load	If enabled, the loadscreen remains open to start a new load.

## Section 31

# Information Commands Windows Interface

### 31.1. Attachment (Properties/Info)

This command shows the attachment connected to a file. An attachment can be connected to a file using the right button down on the file or using the modify/modify attachment menu under properties.

### 31.2. Copy (popup)

Command COPY copies the file into the work-environment on ClearPath Enterprise Servers.

### 31.3. Compare (popup)

Command COMPARE allows comparison of the same file in different environments. The files are downloaded from the SURE environments to the PC, if they are not present in the defined source directories of the PC.

Other compare methods are:

- Compare the checked out version of a file with the version in SURE:  
Select the file and click **Use function Compare** in the **speed** menu.
- Compare the checked out version of a file with the version in SURE:  
Select the file and click **Use function Compare** in the **speed** menu.
- Compare the current version of the file with the version in another environment:  
Select the file and click **Use function Compare** in the **speed** menu.
- Compare the checked out version of a file with a version that is resident under CANDE:  
Select the file and click **Use function Compare** in the **speed** menu.
- Compare the current version of a file with a previous version:  
Select the delta file and click **Use function Compare** in the **speed** menu.
- Compare a previous version of the file with another previous version:  
Select both the delta files and click **Use function "Compare"** in the **speed** menu of the delta file.

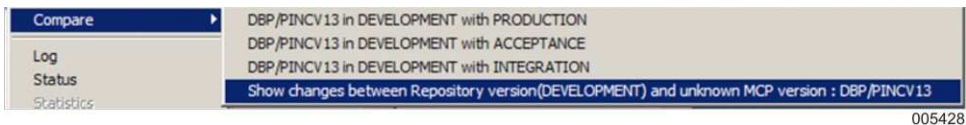
- Compare a file in SURE with another file in SURE:  
Select both the files and click **Use function “Compare”** in the **speed** menu of the file.
- Compare the current version of a file with a previous version of another file:  
Select the file and the delta file and click **Use function Compare** in the **speed** menu.
- Compare a previous version of the file with another previous version of another file:  
Select the delta files of both the files and click **Use function Compare** in the **speed** menu.
- Compare a directory in SURE with a directory on disk:  
Select the **Use function Compare** work-directory in the **speed** menu.

### 31.3.1. Compare Sources That Come from a Third-Party Vendor

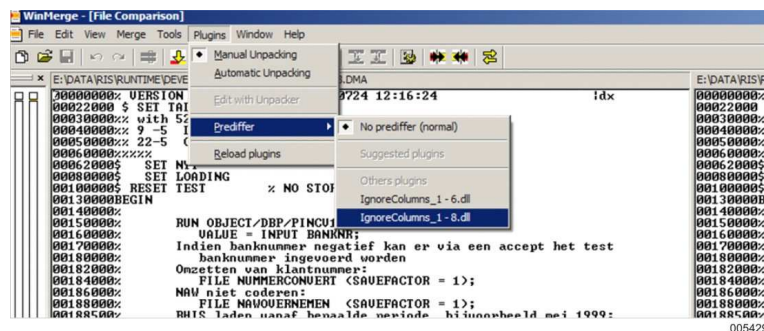
#### MCP Files

For MCP files, it is assumed that the new files are placed under a specific MCP (CANDE) usercode, the same usercode as logged on with to the SURE Explorer.

When you right-click on a file and open the Compare sub menu, there is an option which allows you to compare the MCP (CANDE) file with the repository file.

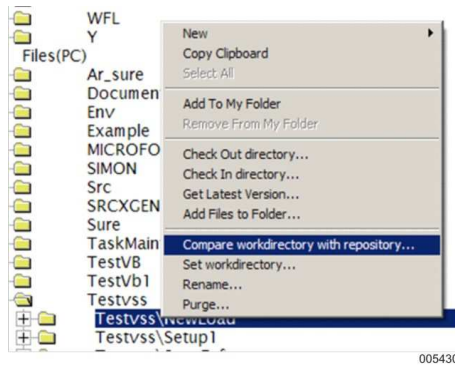


The MCP files contain sequence numbers and the files may have been re-sequenced. For that reason, the WinMerge module has been installed with the “prediffer” plug-ins. Using these plug-ins it is possible to ignore the sequence numbers.



### PC Files

For PC files, it is assumed that the new files are loaded into the work-directory of the PC developer. After doing so, right-click the directory and select **compare work directory with repository**.



## 31.4.Compile (popup, Properties/Info)

The COMPILE LOCAL command allows direct compilation of the source. If the source is in the repository, it is extracted from the repository and compiled. If the file is checked out, then the work file is compiled.

The COMPILE SURE command schedules a compilation for a file in the batch interface. Therefore, this command adds a file in the SURE compile queue.

## 31.5.Delta (Properties/Info, Expand)

The DELTA command shows an overview of the existing delta files.

## 31.6.Edit/View (popup)

The EDIT command is enabled if a file is checked out. This command invokes the defined ClearPath Enterprise Servers editor with the work file as a command line parameter.

The VIEW command is enabled if a file is checked in. This command invokes the defined ClearPath Enterprise Servers editor with the source file as a command line parameter. The source file is read-only which prohibits altering the file.

### 31.7.Errors (popup, Properties/Info)

The ERRORS command shows the syntax errors that are the result of a COMPILE LOCAL. To invoke it:

1. Click the **ERRORS** command.
2. Click **LOCAL**.

The ERRORS shows the syntax errors that are stored in the SURE repository, that are the result of the compilation by the ClearPath server batch compilation or by the PC BUILD support. To invoke it:

1. Click the **ERRORS** command.
2. Click **SURE**.

### 31.8.Information (Properties/Info)

This command shows the information attached to a file.

### 31.9.Integrity (Properties/Info)

This command shows the current integrity chain for a file.

### 31.10. List (Properties/Info)

This command shows the contents of the file.

### 31.11.Log (popup, Properties/Info)

This command shows the log entries for a file.

### 31.12. Open With

To configure the Utilities that are used to open the files using the SURE Explorer:

1. Click **Option**.
2. Click **SURE options**.
3. Click the **local options** tab.

An editor for MCP files, a default editor for PC files, and a DIFF utility are configured. For PC files, the Windows Explorer setting of the file extension has priority over the default PC file editor.

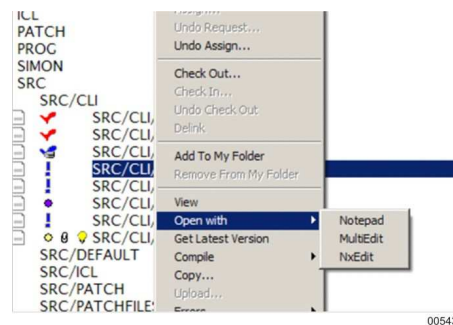


SURE allows an additional method to define extra PC editors to open files. It uses entries in the INI file according to the next example.

```
[OPENWITH]
MultiEdit=C:\MEW\MEW32.EXE
Notepad=notepad.exe
NxEdit=C:\Program Files\Unisys\MCP\ProgWrkbench\NXEdit.exe
```

- The entries must be defined in [OPENWITH] paragraph.
- Each editor is on a single line: the logical name of the editor, followed by the name of the executable.
- The list with the defined editor appears in the **Open With** menu.

The "Open with" function is located in the popup menu of a file.



### 31.13. Reference (Properties/Info, Expand)

The REFERENCES folder shows the references of a file. Using Configuration/Reference, the actual classes of the reference information are defined.

### 31.14. Reminder (Properties/Info)

This command shows the reminder information attached to a file.

### 31.15. Run (popup)

The RUN command allows execution of a file.

### 31.16. Select

Use folder “Select” to select a specific file. You can also browse to a file using folders “Files MCP” or “Files PC”, but sometimes it is faster to use the Select folder, for example in the case of a short file name.

1. Click the Select folder.

The **Select File** dialog box appears.

2. Enter the required name in the **File name** box.



It is also possible to select a file in the SURE-browser using its file nodes. The benefit is an easy method to select a file with a long name (less chance to make typos). A file name in SURE consists of file nodes that are separated by a slash or back slash.

All files that contain the node are returned to the browser in the “Select File” folder. Click the file that you need to continue.

It is also possible to enter two file nodes separated by a space. The files that have both nodes in their name will then be selected.

The complete method to select a file is as follows (the search is not case-sensitive):

- If the entered name exists as a task, give an error.
- If the entered name exists as a file, then select it.
- If the entered name exists as a file with reversed slashes ( / \ ), then select it.
- If the entered name is an existing PC file name without file-extension, then select it.
- If the entered name is used as a file-node, then select all files with that node.

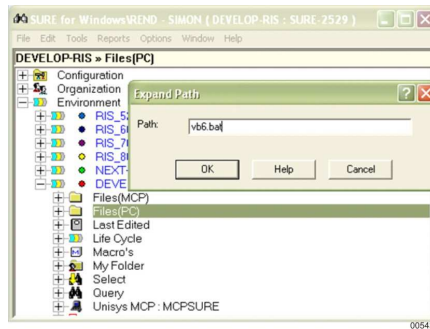
### 31.17. Expand Browser to Path

The SURE browser offers a function that expands the file tree to the first path that contains a file with a specific last node.

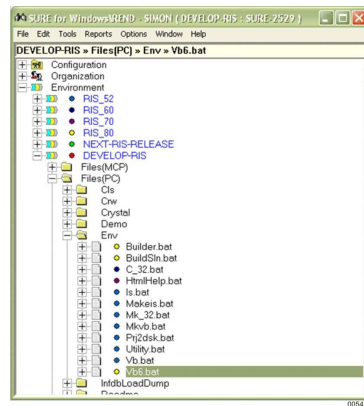
## Example 1

The following example shows the first path that contains file "vb6.bat".

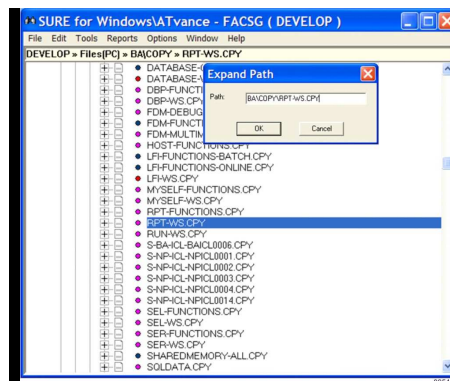
1. Right-click the "Files(MCP)" folder or "Files (PC)" folder and select "Expand to path function."



2. Enter the last node of the file and click **OK** to expand the folder to the first path that contains that file.



## Example 2



All paths in the entered name are automatically opened.

If the entered name is an existing file name then that file is made current and positioned in the middle of the SURE Explorer.

### 31.18. Status (popup, Properties/Info)

The STATUS command shows the status information over different environments.

### 31.19. Statistics (popup, Properties/Info)

The STATISTICS command shows the run-time statistics information that was loaded from the system SUMLOG.

### 31.20. Task (Properties/Info, Expand)

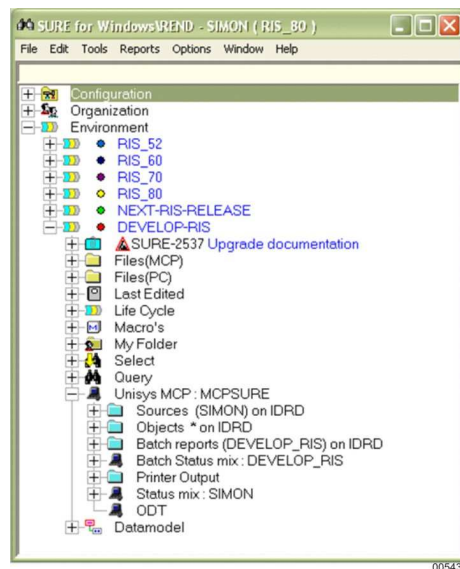
The TASK ACTUAL folder shows the actual linked tasks for a file.

The TASK HISTORY folder shows the historical connected tasks for a file.

### 31.21. Write (Properties/Info)

The WRITE command invokes the RESPECT/SURE/SOURCELIST program on ClearPath Enterprise Servers.

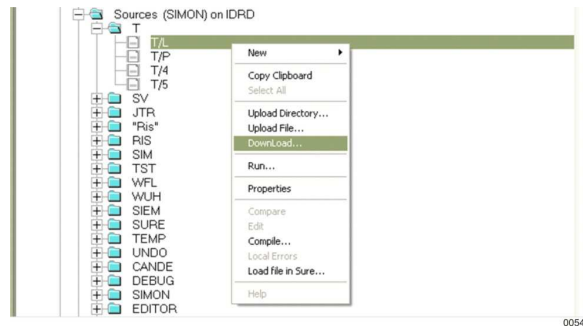
### 31.22. Unisys MCP Folder



The Unisys MCP folder is followed by the host name of the MCP system where the SURE repository is located. It has a number of subfolders as described below.

### 31.22.1.Sources (<CANDE Work Usercode>) on <Work-Pack>

This is your CANDE work location. Files are checked out using this work location and local compilations are started in this directory.

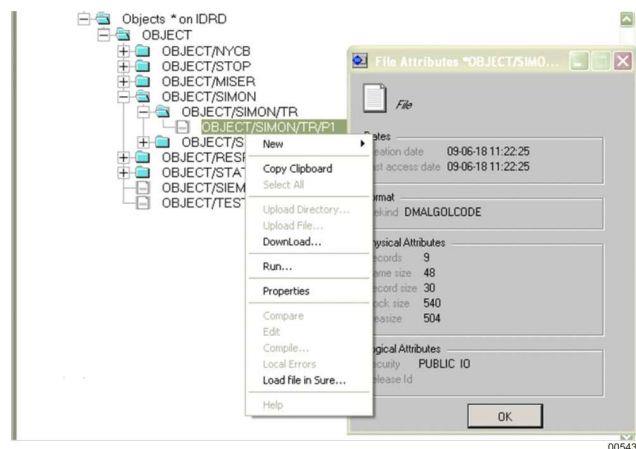


- You can download a file from your CANDE work location to your SURE-temp-directory on Windows, for example to load it in a local editor.
- You can upload files in your SURE-temp-directory to your CANDE-work-location using "Upload file."
- You can download and upload directories of files.
- You can compile source files that are located in your SURE-temp-directory using a local compilation, started from your local editor (like Net Express). See "MultiEdit" in Section 11.

### 31.22.2. Objects <Directory> on <Pack>

This is the directory that contains the objects for the system of the current task. Each time you change from current task, the object-directory is re-calculated using the system of the new current task. This may also be the location of stable copy files.

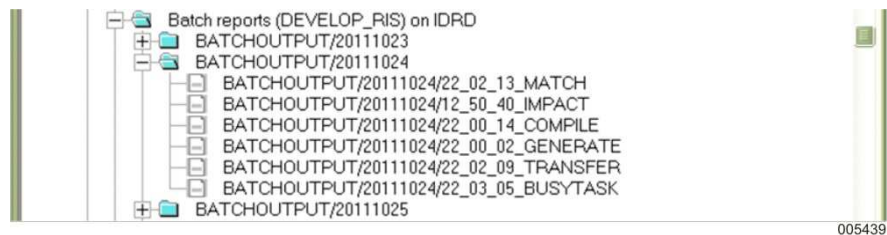
Right-click an object file and select **Properties** to check some Unisys file attributes, like creation date and last access date as shown in following example.



### 31.22.3. Batch Reports (<SURE-Batch-Usercode>) on <SURE-Batch-Pack>

This shows the overviews that are created by the SURE batch of this environment.

- The SURE-batch-overviews are grouped for each day. You can open the folder of a day to download a specific overview. It is possible that the SURE batch ran multiple times on a date, and in that case the overviews of each run are shown. The time of the run is part of the file name. The last part of the file name identifies the type of overview. For example, the BATCHOUTPUT/20111024/22\_00\_14\_COMPILE file is the compilation overview, created by RESPECT/SURE/COMPILE at 22:00:14 on October 24, 2011.



- Right-click a backup file to download it from MCP to Windows and to open it with MS Word.

### 31.22.4. Batch Status Mix: <SURE-Batch-Usercode>

This shows the activity on the MCP for the SURE-batch-usercode: active, waiting, scheduled and completed programs, displayed messages, and the content of some predefined queues.



To refresh the status-overview, press **F5**.

### 31.22.5. Printer Output

This shows the waiting print requests for your usercode.

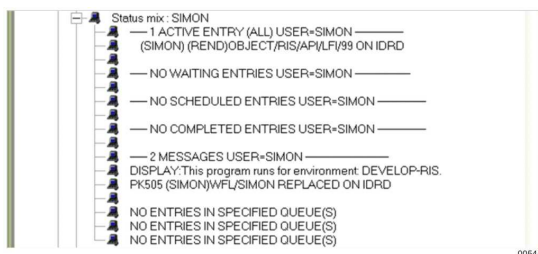


- Right-click the "Printer Output" subfolder gives you the option to clear your complete list with print requests.
- Right-click a single print request gives you the option to delete that print request.
- Open a print request folder to see the backup files that are created under that request.

Right-click the backup file to download it from MCP to Windows and to open it with MS-Word.

### 31.22.6. Status Mix: <My-Usercode>

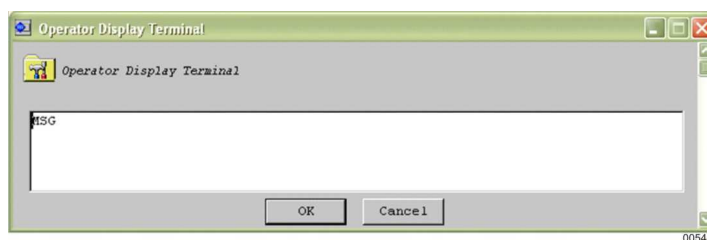
This shows the activity on the MCP for your usercode: active, waiting, scheduled and completed programs, displayed messages, and the content of some predefined queues.



To refresh the status-overview, press **F5**.

### 31.22.7. ODT

Double click the ODT to open the ODT screen, where you can enter ODT commands.



You can use the same set of ODT commands that you can also use from the MARC screen.





## Section 32

# System and Project Definition

A system is a group of files or definitions that have a strong physical (and logical) relation. In SURE, this physical relation is emphasized by the definition of object-location and work-location. All files in a system share the same object-location and work-location. In the RIS software, a system has a one to one relation with a physical database.

A project is a sub-system.

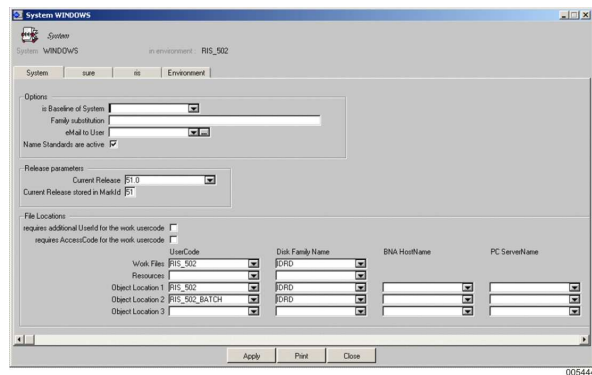
The system names or project names are not restricted by any rules. However, a name standard may use the system or project name. This causes the system or project name to be a part of a FileName. Therefore, in the case that name standards are used for files on ClearPath Enterprise Servers, the system and project names must meet the ClearPath server requirements for file names.

The configuration folder system is used for the definition of the attributes of a system or a project, respectively. Multiple projects can be connected to a system, but a project can, and must, be connected to a single system. The popup menu under system allows entering a new system name. Thereafter, you need to define the properties for every environment. Another mechanism is selecting an existing system in an environment and selecting new from the popup menu. This action copies the attributes from the selected system.

SURE requires that a file is linked to a project. This definition will implicitly connect the project's system to the file.

### 32.1. System Attributes

The attributes defined for a system are defined for each environment. Generally, this definition differs for each environment.



### <Baseline>

This option identifies that the system is an extra baseline of another system. Each baseline is tightly connected to its own disk pack. Source maintenance in a baseline is done using patch files.

For more information on Maintenance using Baseline, refer to the *SURE Assistant* (power point) from SURE Help menu.

### <Family Substitution>

This option is relevant only when you work with baselines and NX/Edit as local editor. This option substitutes the family declaration of your usercode in the userdatafile with the work family of the baseline, if you start working on a task of that baseline.

For more information on Maintenance using Baseline, refer to the *SURE Assistant* (power point) from SURE Help menu.

### <Object Location 1, 2 and 3>

Three default object-locations for each system. If you are changing these variables then the object-locations for the files that inherit these locations will also change.

See “Transfer” and “Enter a New File” in Section 10.

### <Current Release>

The current release defines the number used in the `SOLVED-IN-RELEASE(<current release>)` relation. This relation is linked to a task if it is solved in an environment.

A separate option is available to define the current release number (two digits) that will be placed in the markid of changed program lines.

It is possible to link a release number to an application system. All tasks that get status solved will get that release number, and the release number is placed in the markid of changed lined in ClearPath server sources. The benefit of a release number is that it gives a global indication when a record was changed or when a task was solved (recently or lately).

Many sites do not work with release numbers. Therefore, entering a number does not mean anything to most of the developers. A solution for this is to use the current year as the default release number. Enter “YY” as the release-id of a system to enable this function.



The “Current Release in MarkId” has value “YY” that enables the year release function.

Refer to Section 29, “Software Delivery,” for more information.

### **<Integrity Mechanism is Active>**

This option enables application system integrity control by SURE. This option should not be set for a develop environment.

See “Integrity Mechanism” in Section 23.

### **<Name Standards are Active>**

This option causes all file names to be derived from defined name standards for each FILE-TYPE.

See “Enter a New File” in Section 10.

### **<Put Copy Files in the Work-Environment After Save or Load>**

This option copies the copy file to the work-environment if the file is checked in SURE. This option should be set for a develop environment.

Refer to Section 15, “Copy Files,” for more information.

### **<Resource Environment>**

The resource environment defines the default location for copy files and other resources.

This location is only used when option <Use resource versions> is enabled.

See “(Copy Files) <Use Resource Versions>” in Section 15.

### **<Resource Prefix>**

The resource prefix will be added to qualified names of the resources, to define them uniquely across versions and systems.

See “(Copy Files) <Use Resource Versions>” in Section 15.

### **<Use Resource Versions>**

This option sets the resource versions for a system.

See "(Copy Files) <Use Resource Versions>" in Section 15.

### **<De-implemented>**

This option marks a system as de-implemented. Files of this system are skipped by the examine and find processes. All relations created by the examine process are removed for these files, so that they are not selected by the query function anymore. The object-location of these files is also removed, which enforces that they are not compiled anymore.

### **<Work Files>**

This identifies the work-environment.

The work-environment defines the usercode and family where maintenance is done for this system. If the usercode equals to "\*", all usercodes on that family are allowed to work for this system on the specified family.

The work-environment is also the location where the copy files are placed.

The SURE Explorer software creates the SURE maintenance copies in this defined environment (for the defined system of the file or task).

This location for copy files is only used when the <Use resource versions> option is disabled.

See "Check-Out Procedure" in Section 10.

### **<Check Abbreviations on New Items>**

RIS users only: this option causes RESYDE and the EDITOR to verify if new items are according to defined abbreviations.

### **<Decimal Point is Comma>**

RIS users only: an option used by the RIS generators (FCM, FDM, and RPT).

### **<Generate COBOL85 Sources>**

RIS users only: an option used by the RIS generators, (FCM, FDM, and RPT).

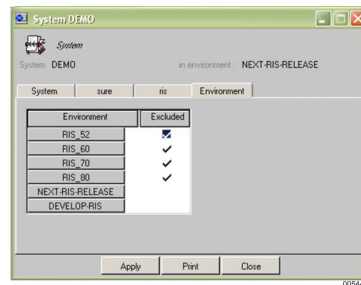
### **<New Items have Abbreviations in Fixed Pattern Txxxxyyyyzzz>**

RIS users only: a specific case of verifying item names.

### **<Exclude (On Tab Environment)>**

With this option, it is possible to exclude one or more environments for this system.

## Example



The repository of this example has six environments, but for system DEMO the top four are excluded. Therefore, DEVELOP-RIS and NEXT-RIS-RELEASE are used by system DEMO.

## Checking the (Excluded) Environments of a System

A source that belongs to a system contains excluded environments should not exist in those environments.

There is a batch function to check the content of environments as follows:

- Sources that belong to a system that has excluded environment are removed from those environments by the batch function. Delta files are also deleted from the excluded environments.
- Sources that are not available in an environment, are recovered from the next higher environment.

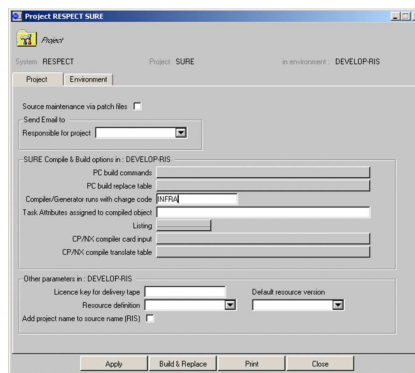
The batch function can be used when an environment is excluded or included for a system to check all files of that system.

Run the program as follows:

```
RESPECT/REPOSITORY("CHECK-SYSTEM-ENVIRONMENTS <system>");
```

## 32.2.Project Attributes

The project attributes differ for each environment.



### **<System>**

A project must be connected to one system.

Files that are connected to a project will inherit the definitions of the project's system.

### **<Source Maintenance Using Patch Files>**

This option enforces that sources of this project must be maintained using patch files.

### **<Responsible for Project>**

An email is sent to the user with this employee-function if a task of this project arrives on the environment where this option is set, or when syntax errors are found when files of this project are compiled.

### **<PC Build Commands>**

These commands are used to create the build.bat file.

### **<PC Build Replace Tables>**

Not yet implemented.

### **<Compiler Charge Code>**

This charge code is linked to the SURE batch compilations of files of this project.

### **<Task Attributes Assigned to Object>**

These task attributes are assigned to the object if a source of this project is compiled.

Refer to Section 23, "Compilation and Object Files," for more information.

### **<Listing>**

This option identifies what to do with the compilation listing

See "Compile Listings" in Section 23.

### **<CP/NX Compiler Card Input>**

This option is input for the batch compilation card file.

See "Compiler Control Cards" in Section 23.

### **<CP/NX Translate Table>**

This option is the input for the batch compilation translate table.

Refer to Section 23, "Compilation and Object Files," for more information on compiler translate tables.

### **<License Key for Delivery Tape>**

This license key is linked to the sources of this project that are delivered.

Refer to Section 29, "Software Delivery," for more information.

#### <Default Resource Version>

This defines the environment to choose for retrieving resources, if all other resource location resolution rules are exhausted.

See "(Copy Files) <Use Resource Versions>" in Section 15.

#### <Resource Definition>

This defines the name of a resource definition table to be related to the project.

See "(Copy Files) <Use Resource Versions>" in Section 15.

#### <Add Project Name to Source Name>

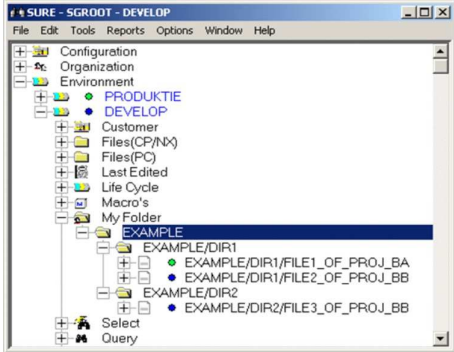
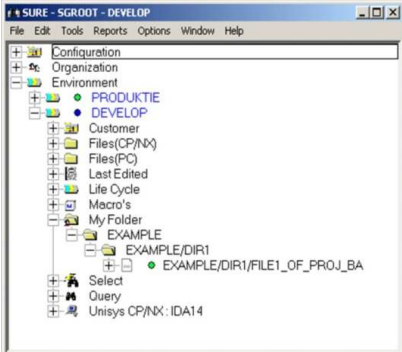
This option is only applicable for RIS users.

## 32.3.Hide Directories that Do Not Contain Any Files of My-Project-List

A user can be linked to one or more systems and projects (sub systems). This list of projects limits the visibility of files in the directory folders.

- If the user has an empty project list (not linked to any project), then the SURE browser shows all files of all systems and projects.
- If the project list of the user is not empty, then the SURE browser shows only the files that belong to systems or projects of his list.
- If a directory (in the tree directory) does not contain any files that belong to the users' project list, then that directory is hidden.

### Example

A user with an empty project list sees:	A user with project BA in his project list sees:
Directory EXAMPLE/DIR1 with files of projects BA and BB. Directory EXAMPLE/DIR2 with files of project BB.	Directory EXAMPLE/DIR1 with files of project BA.
	

If global option "user can see files of projects that are not in his project list" is set, any user can see all files of all systems and projects.

The following folders are filtered according to the users project list:

- All directory folders under "Files(CP/NX)," "Files(PC)," and "My Folder."
- Files Requested folder under Life Cycle folder.
- All folders that are the result of a query or a macro.
- The compile interface.



## Section 33

# File-Type

A FILE TYPE in SURE defines a logical name for a type of file. This type has no relation to the physical FILEKIND attribute.

The file type is used for two main purposes:

- The file type is used in SURE to create logical groups of files with the same SURE attributes.

Examples of this purpose are:

- Naming standards are defined for each file type. All files with the same file type get a similar name.
  - Inheritance of object usercode/pack attributes. All files with the same file type inherit the same object usercode/pack.
- The file type is used as a trigger for SURE to perform certain actions on the source.

The following actions depend on the file type.

<copy-file>	<p>Copy files are (optionally) placed in the CANDE work-environment. A file with a file type that is marked as "copy file" file type will be placed in the CANDE work-environment when this file is saved in SURE or transferred to a next environment (this will only be done if the corresponding option is set).</p> <p>If a copy file uses nested include files (in ALGOL or COBOL85), and the set of included copy files is changed, then all the master programs of that copy file will be re-examined.</p> <p>The integrity mechanism of the SURE compile procedure is based on copy files. If a copy file is changed, then all master programs of that copy file will be recompiled.</p>
<Use as start job (for example, XGEN)>	<p>If you have files where you need to do something for in a start-job, then these files are first compiled and thereafter the start-job is started. If the compilation phase must be skipped, use the "only use as start-job" option.</p> <p>XGEN users only: RESPECT/SURE/COMPILE will start the XGEN generator for files with this file type. The EXAMINE process will scan for specific XGEN statements.</p>
<makefile>	<p>PC-files only. A PC-file with this file type will be scanned by the examine process to create a dependency table.</p>
<C/C++>	<p>PC-files only. A PC-file with this file type will be scanned by the examine process for copy statements.</p>
<MFCobol>	<p>PC-files only. A PC-file with this file type will be scanned by the examine process for copy statements.</p>

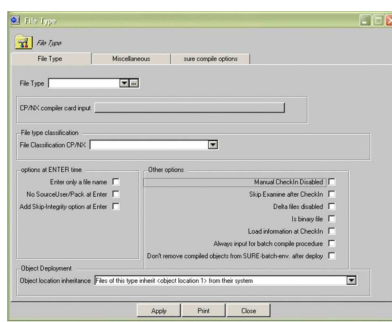
The following file types are only relevant for RIS users:

<Rule>:	<p>A source with this file type is a RULE. If such a file is changed then the necessary DIM generations will be started during the evening batch.</p> <p>The EXAMINE process will scan for specific Rule statements.</p> <p>Files with this file-type cannot have an object usercode/pack.</p> <p>Files with this file-type are used in various overviews about rules.</p>
<Rpt>:	<p>A source with this file type is an RPT script file. If such a file is changed, then the RPT generation will be started for that script-file during the evening batch.</p> <p>The EXAMINE process will scans for specific RPT statements.</p> <p>Files with this file type cannot have an object-usercode/pack.</p>
<Fcm>: FCT:	<p>A source with this file type is an FCM script file. If such a file is changed, then the FCM generation will be started for that script file during the evening batch.</p> <p>The EXAMINE process will scan for specific FCM statements.</p> <p>Files with this file type cannot have an object usercode/pack.</p>
FDM:	<p>A source with this file type will be bound into a host (HST) if this source belongs to the same project of that host (or an included project).</p> <p>Objects of this file type can be recompiled automatically during the "RIS-load-database" process.</p>

HST:	Objects of this file type can be recompiled automatically during the "RIS-load-database" process.
LFI:	Objects of this file type can be started using the RUN LFI screen in RIS.
DODDLE:	A file with is file type contains PC specs (transaction definitions and formats) that are input for a corresponding online LFI.
<Letter>	A source with this file type is a RIS letter file. A letter generation will be started during the evening batch if such a file is changed.

To define file type:

1. Click Configuration folder.
2. Click **Type**.



The following paragraphs describe the attributes that can be declared for each file type.

### 33.1. File Type Classification

It is possible to split a logical file type into multiple physical file types. The logical file type is used as a trigger for SURE to perform certain actions on a source, and the physical file type can be used for grouping purposes.

#### Example

Consider two types of copy files:

- Copy files that contain declarations (COBOL data division).
- Copy files that contain statements (COBOL procedure division).

The copy files of type 1 (declarations) must have a name that starts with WS/COPY/... and the integrity mechanism must be enabled for this type of copy files.

The copy files of type 2 (statements) must have a name that starts with PD/COPY/.... and the integrity mechanism must be disabled for this type of copy files

Defining two file types can create these situations:

1. File type WS-COPY for copy files that contain declarations.
  - The “Use never object relations” option is set.
  - The file type has a name standard, which enforces names that start with WS/COPY/.
  - The “add skip-integrity at enter” option is NOT set.
  - The file type is classified as “COPY-FILE” type.
2. File type PD-COPY for copy files that contain statements.
  - The “Use never object relations” option is set.
  - The file type has a name standard, which enforces names that start with PD/COPY/.
  - The “add skip-integrity at enter” option is set.
  - The file type is classified as “COPY-FILE type.”

The following file type classes are available:

### **<Copy-file> Type**

A file with this file type is considered as a copy file. Refer to the table above for more information about copy files.

### **<XGEN> Type**

A file with this file type will be treated as an XGEN file. Refer to the table above for more information about XGEN files.

### **<RPT> Type**

A file with this file type will be treated as a RPT script file. Refer to the table above for more information about RPT script files.

## **33.2.Object-Usercode/Pack Options**

The object-usercode/pack of a source determines where the object of that source will be placed after compilation. It is obvious that the object-usercode/pack combination of a source differs for each environment.

Default object-usercode/pack combinations can be declared for each application-system and for each environment. If a source arrives newly in an environment (using the function New/File, or when it is transferred for the first time to an environment), the objects-attributes are inherited from its system defaults for that environment.

It is possible to declare three different object-attribute combinations for each application system and for environment: object-location-1, object-location-2 and object-location-3.

Each of these three can be the default object-location in any file type declaration.

The object-location of a file is re-evaluated in the following cases:

- When the system or file type of that file is changed.
- When object-location of the file system is changed.
- When the object-location-inheritance option of the file's file type is changed.

#### <Inherit Object-Location 1 From the System>

A file with this file type will inherit the object usercode/pack from object-location 1 that is declared for the application system of the file.

#### <Inherit Object-Location 2 From the System>

A file with this file type will inherit the object usercode/pack from object-location 2 that is declared for the application system of the file.

#### <Inherit Object-Location 3 From the System>

A file with this file type will inherit the object usercode/pack from object-location 3 that is declared for the application system of the file.

#### <No Object-Location>

A file with this file type will not inherit any object-location. It is not possible to update the object-location of the file.

#### <Manual Defined Object-Location>

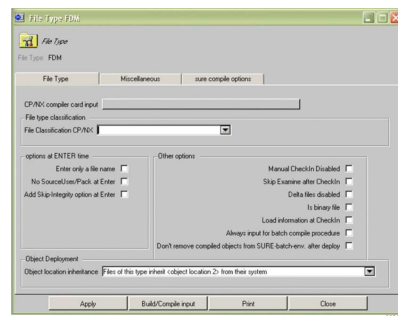
A file with this file type does not inherit the object-location, but it is manually defined (It is possible to leave the object-location empty).

#### Example:

Consider the following System definition.

The screenshot shows the 'System WINDOWS' dialog box with the 'System' tab selected. The 'in environment' field is set to 'RIS\_502'. The 'Options' section includes 'is Baseline of System' (unchecked), 'Family substitution' (empty), 'eMail to User' (empty), and 'Name Standards are active' (checked). The 'Release parameters' section shows 'Current Release' as 'R1 0' and 'Current Release stored in MarkId' as 'R1'. The 'File Locations' section has checkboxes for 'requires additional Userid for the work usercode' (unchecked) and 'requires AccessCode for the work usercode' (unchecked). Below these are fields for 'Work Files', 'Resources', 'Object Location 1', 'Object Location 2', and 'Object Location 3', each with a 'UserCode' dropdown and a 'Disk Family Name' dropdown. The 'UserCode' dropdowns are set to 'RIS\_502', 'RIS\_502', 'RIS\_502\_BATCH', and 'RIS\_502' respectively. The 'Disk Family Name' dropdowns are set to 'ICPD', 'ICPD', 'ICPD', and 'ICPD' respectively. The 'Apply', 'Print', and 'Close' buttons are at the bottom.

Consider the following File Type definition.



The result of these two definitions is that all source of system WINDOWS with file type = FDM inherit object-location-2. This object-location is (RIS\_502\_BATCH) ON IDR. If one of these sources is compiled by SURE, then the object is deployed to (RIS\_502\_BATCH) ON IDR.

### 33.3. Other File Type Options

#### <Enter Only a FileName>

When a file with this file type is added in SURE, then only the name of the file is added. SURE does not expect an actual source for such a file name.

See “Enter a New File” in Section 10.

#### <No Source-User/Pack at ENTER>

This option informs SURE not to register a file as “checked out” when the file is created in SURE. This option is often used to register data files in the repository.

See “Enter a New File” in Section 10.

#### <Add Skip Integrity at ENTER>

A copy file with this file type option will not create integrity chains during the SURE compile process. A file with this file type option will not be part of an integrity chain during the SURE compile process.

See “Integrity Mechanism” in Section 23.

#### <Manual Check In Disabled>

A file with this file type cannot be checked out manually and can only be loaded in SURE using automated procedures.

#### <Load Information at Check In>

Instructs SURE to store comment-lines in the source as textual information during the save of the source in the repository. This information can be inquired using the button INFO.

Refer to Section 14, “Text as Documentation/Information,” for more information.

**<Delta Files Disabled>**

No delta files will be stored for files with this file type. This option can be used for binary PC files or for generated files.

See "Binary Files" in Section 12.

**<Is Binary File>**

This option can be used for binary files like executables, or binary PC files.

See "Binary Files" in Section 12.

**<Skip Examine After Check In>**

Files of this file type will not be examined by RESPECT/SURE/EXAMINE.

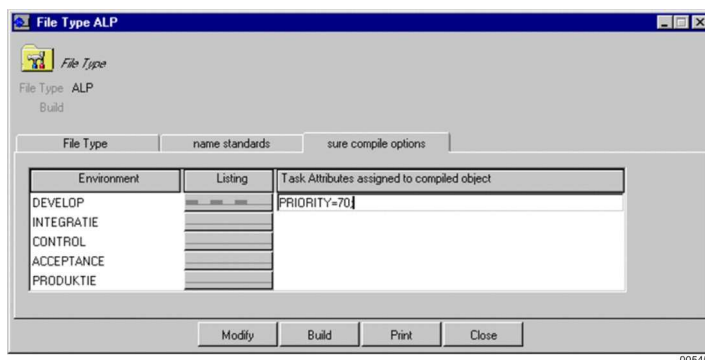
See "RESPECT/SURE/EXAMINE" in Section 25.

**<Last Node is ID For Copy-Files>**

This option is used by the examine process to determine the correct name of the include files that are mentioned in a PC file. The examine process uses the last node of a copy file name to obtain the correct and full name of the copy file (without unnecessary directory levels).

**<Always Input For Batch Compilation Procedure>**

Files with this file type are always copied from SURE to the disk during the initialization phase of RESPECT/SURE/COMPILE. If a wrong version of such a file is already resident, then it will be overwritten with the correct version. The files are not removed when the RESPECT/SURE/COMPILE goes to end-of-task. The result of this is that the correct versions of these files are always resident under the SURE batch usercode, and available for other compilation or generation processes that are not controlled by SURE.

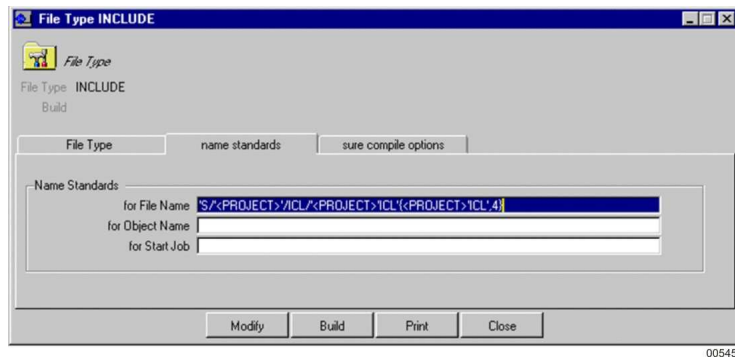
**<Define Extra Task Attributes For the Compiled Objects>**

See "SURE Compile Options" in Section 35.

**<Define Compile Listing Options For Each File type>**

See "Compile listings" in Section 23.

### 33.3.1. Other File Type Options for Mainframe Files



#### <Name Standard for File>

If a file with this file type option is entered (new), then the name of the file is determined by SURE according to a name standard formula.

#### <Name Standard to Create a Start Job Name at ENTER>

If a file with this file type option is entered, then a corresponding start job will be created too. The start job will get a name according to the formula of the given name-standard.

Refer to Section 39, "Name Standards," and "24.1 (Binding) START-JOB and DRIVER Relations."

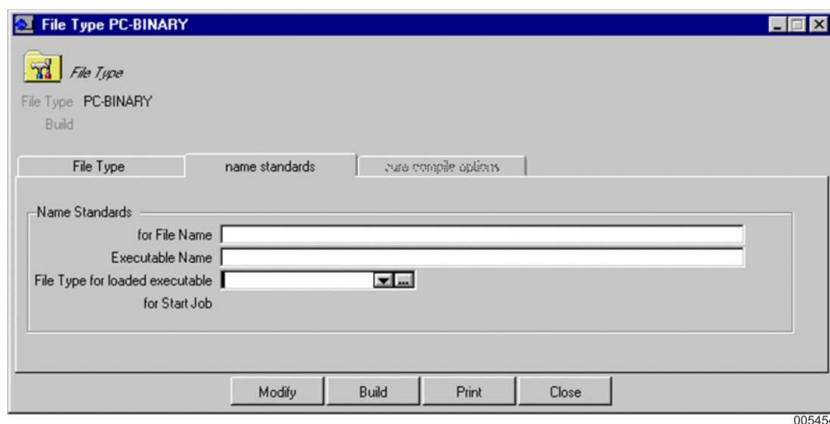
#### <Name Standard to Create Deviating Object Name at ENTER>

If a file with this file type option is entered, then the source gets a deviating object name. This object name is determined according to the formula of the given name standard.

Refer to Section 39, "Name Standards "and "23.10 Object Names."



### 33.3.2. Other File Type Options for PC Files



#### <Name Standard for FileName>

If a file with this file type option is entered (new), then the name of the file is determined by SURE according to a name standard formula.

#### <Name Standard to Create a Start Job Name at ENTER>

If a file with this file type option is entered, then a corresponding start job will be created at the same time. The start job will get a name according to the formula of the given name standard. The start job will be executed automatically each time when the file is checked in to SURE.

Refer to the Section 39, "Name Standards," for more information.

#### <Name Standard to Create an Executable Name at ENTER>

If a file with this file type option is entered, then an executable name is determined by according to the name standard formula. The executable name is linked to the file name.

Refer to the Section 39, "Name Standards "and "Object Names" in Section 23.

#### <File Type for Loaded Executable at ENTER>

If a file with this file type option is entered, and an executable name is determined too (see previous option), then the executable name will get this file type.

#### <Software Distribution>

This option is only applicable for PC files. A PC file with this file type will inherit a server location (default, bind or alternate according to the other options) of the application system. This file will be placed on the target server where the PC application runs.



## Section 34

# Authorization Mechanism and Log On

Access to the SURE system is only possible if a user is logged on with a usercode that is known in the userdatafile. This implies that the security in SURE is based on two levels:

- MCP or operating system security.
- SURE security mechanism.

The MCP security level provides a mechanism where usercodes can be restricted to access files or to perform functions. With the SURE security mechanism, it is possible to authorize users for specific SURE functions.

### 34.1.Unique Identification in SURE For Each User

It is important that each person who works with SURE system can be recognized by the system using a unique usercode. This is necessary because the usercode of a person is used for routing purposes (this task has to be solved by this usercode), for logging purposes (who adapted this source and when), authorization purposes (is this user allowed to perform this function), and other identification purposes.

Some sites have chosen to work with multiple programmers under the same (project) usercode. The common reason for this choice was to keep files that belong to different projects in separate directories on disk, and to give visibility and access to a set of files to the programmers that have to work with these files.

In the SURE Explorer interface, each user has a personal and unique usercode.

SURE makes a distinction between the usercode under which the user is logged on, and the usercode of the work-location.

### 34.2.The Logon Screen

A user must log on to the SURE Explorer using the following screen.

The screenshot shows the SURE Logon window. It includes fields for Usercode (SGROOT), Password, Accesscode, Accesscode pwd, and Chargecode, each with a Change button. There is a checkbox for 'Save these credentials'. The 'Current Environment' section has dropdowns for Environment and Task, and labels for Host, Pack, and Printer. The 'modify SURE explorer layout according to role' section has radio buttons for various roles and their corresponding dropdown menus. Buttons for Logon, Start SURE Explorer, and Cancel are present. A small number 005455 is at the bottom right.

A user must enter a valid usercode, and optionally the password, accesscode, accesscode-password, and chargecode. The usercode must be known in the userdatafile.

Other fields on the Logon screen:

Save credentials	If the credentials are saved then SURE will logon automatically.
Environment	The SURE environment where the user logged on the previous time.
Task	The users current task.
Role	The layout of the SURE Explorer is customized according to the chosen role. If no role is chosen, then all possible options and folders are presented.
Host	The name of the host.
Pack	The current workspace family.
Printer	The mainframe printer for this user.
News	The last system news

#### 34.2.1. Make the Logon Credentials Customizable

It is possible to customize the logon credentials. By default, the Logon screen contains fields for usercode, usercode-password, accesscode, accesscode-password and chargecode. It is possible to hide one or more of these fields and to change the names of these fields.

The default Logon screen is as follows.

005456

### Example 1: Hide Fields

It is possible to hide the Password, Accesscode, Accesscode pwd, and Chargecode fields using the following settings in the AW\_OBJ.INI file:

[ CUSTOMLOGON ]	
HIDEPASSWORD=TRUE	Hide the password field.
HIDEACCESSCODE=TRUE	Hide the accesscode field.
HIDEACCESSCODEPWD=TRUE	Hide the accesscode password field.
HIDECHARGECODE=TRUE	Hide the chargecode field.

Example of a possible definition in the AW\_OBJ.INI file and the corresponding Logon screen.

```
[ CUSTOMLOGON ]
HIDEPASSWORD=FALSE
HIDEACCESSCODE=TRUE
HIDEACCESSCODEPWD=TRUE
HIDECHARGECODE=FALSE
```

005457

### Example 2: Customize Field Names

It is possible to define your own log on credentials using the following settings in the AW\_OBJ.INI file:

[CUSTOMLOGON]	
VAR1=<name>	Override the default name of the first logon field. This key is mandatory.
VAR2=<name>[ , TRUE]	Override the default names of the other four fields in custom logon. If a key is not used then the corresponding field is hidden. The "TRUE" option indicates that the field is a password.
VAR3=<name>[ , TRUE]	
VAR4=<name>[ , TRUE]	
VAR5=<name>[ , TRUE]	

Example of a possible definition in the AW\_OBJ.INI file and the corresponding Logon screen:

```
[CUSTOMLOGON]
VAR1=UWNetId
VAR2=UWNetId Passwd, TRUE
VAR4=SecurId
```

The screenshot shows a logon dialog box with a title bar. Inside, there are three input fields: 'UWNetId', 'UWNetId Passwd', and 'SecurId'. The 'UWNetId Passwd' field has a 'Change' button next to it. Below the fields is a checkbox labeled 'Save these credentials'. To the right of the input fields are three buttons: 'Logon', 'Start SURE Explorer', and 'Cancel'. The text '005458' is visible in the bottom right corner of the dialog box.

Option TRUE for the second field indicates that the field is a password a "change-password" button is placed behind the field, and the entered characters are secured.

Each field has a maximum length of 18 characters.

The default logon authentication validation is done against the userdatafile on ClearPath Enterprise Servers. The following checks are done:

- Does the usercode exist in the userdatafile?
- Is the usercode/password combination valid?
- Is an accesscode required?
- Is the accesscode/password combination valid?
- Is a chargecode required?
- Is the chargecode valid?

It is possible to customize the authentication validation routine according to the requirement, using a site-specific authentication procedure in the SURE site library. An example site library is placed in the SURE installation directory on the mainframe.

The name of this example site library is RELEASE512/RESPECT/LIBRARY/SITE\_FUNCTIONS/EXAMPLE. The authentication validation routine in the example site library is identical to the default authentication validation routine (which is compiled into the software). You can extend or modify this example to your own needs.

It is mandatory to create a customized authentication routine if you customized the fieldnames on the Logon screen (see "Example 2" illustrated earlier in this section). This is obvious, because the default authentication routine would not support that site-specific functionality.

You can also create a customized authentication routine if you want to extend the default authentication routine. For example, if the site has password aging enabled and you want to send a message that the password is almost expired. In that case, you can use the authentication validation routine in the example site library as a quick start.

The authentication validation procedure must be defined as follows in the SURE site library:

```

BOOLEAN PROCEDURE SITE_USERDATA
    (FUNC,USER,S2,S3,S4,S5,S6,FLD,ERRTEXT);
VALUE    FUNC,USER,S2,S3,S4,S5,S6,FLD;
INTEGER FUNC,FLD;STRING USER,S2,S3,S4,S5,S6;EBCDIC ARRAY ERRTEXT[0];
BEGIN
    LABEL EOP
        ;
    DEFINE    FUNC_LOGON                = 1 #
            ,FUNC_VALIDATE_PASSWORD    = 2 #
            ,FUNC_VALIDATE_ACCESSCODE  = 3 #
            ,FUNC_CHANGE_PASSWORD      = 4 #
        ;
    PROCEDURE ERR(S);
    VALUE S;STRING S;
    BEGIN
        REPLACE ERRTEXT BY S,48"00";
        SITE_USERDATA:=TRUE;
        GO TO EOP;
    END;

    CASE FUNC OF BEGIN
    FUNC_LOGON:
        ; %% Write your SURE logon authentication validation at this place
    FUNC_VALIDATE_PASSWORD:
        ; %% A simple usercode/password check for logon via RIS/MENU
    FUNC_VALIDATE_ACCESSCODE:
        ; %% A simple accesscode/password check for logon via RIS/MENU
    FUNC_CHANGE_PASSWORD:
        ; %% A routine to change a password
    ELSE: ERR("Wrong function : "!!STRING(FUNC,*));
    EOP:
    END;
    EXPORT SITE_USERDATA;

```

### Input values:

FUNC	The function: 1 = Logon using SURE Explorer. 2 = Extra logon using RIS/MENU. 3 = Validate accesscode using RIS/MENU. 4 = Change password of the usercode.
USER	The first logon variable of the Logon screen. (default: usercode).
S2	The second logon variable of the Logon screen. (default: password).
S3	The third logon variable of the Logon screen. (default: accesscode).
S4	The fourth logon variable of the Logon screen. (default: accesscode-pwd).
S5	The fifth logon variable of the Logon screen. (default: chargecode).
S6	The new password (only used if FUNC = 4 <change password>).
FLD	The field that contains the old password that must be changed. (Only used if FUNC = 4 <change password>).

### Return values:

ERRTEXT	The reason of the error.
SITE_USERDATA	The procedure must return TRUE if an error was encountered.

A customized authentication routine must support the following functions.

Function	When	Input fields
Logon using the SURE Explorer	Logon using SURE Explorer	FUNC = 1 S2...S5 = All input fields of the Logon screen are passed to the authentication routine using these strings
Extra logon using RIS/MENU.	This function is only called if a user logs on to RIS/MENU and option "extra usercode required" is enabled.  This function is not called from the SURE Explorer.	FUNC = 2 USER = the usercode S2 = the password



Accesscode logon (RIS/MENU)	<p>This function is only called if a user logs on to RIS/MENU using COMS (for example, ?ON RISMENU) and an accesscode is required for the CANDE work usercode.</p> <p>This function is not called from the SURE Explorer.</p>	<p>FUNC = 3</p> <p>USER = the usercode</p> <p>S3 = the accesscode</p> <p>S4 = the accesscode password</p>
Change password	<p>Change password using Logon screen in SURE Explorer.</p>	<p>FUNC = 4</p> <p>S2...S5 = All input fields of the Logon screen are passed to the authentication routine using these strings.</p> <p>S6 = The new password</p> <p>FLD= The number identifies the string (S2 thru S4) with the old password.</p>

The example site-library contains an example for each function.

### 34.2.2. Disable “Save these credentials” at Logon

The Logon screen offers option “Save these credential.” It is possible to disable this option using a Global Option in the repository or using a setting in the AW\_OBJ.INI file.

If the function is disabled using the INI-file, then only the workstations using that INI-file are affected. If the SURE client software is installed using a server-only installation, then all workstations that refer to that server are affected. If the SURE-client software is installed locally on an individual PC, then only that workstation is affected. This makes it possible to enable the function on some workstations and to disable it on other workstations.

If the function is disabled using the Global Option in the repository, then all workstations are affected.

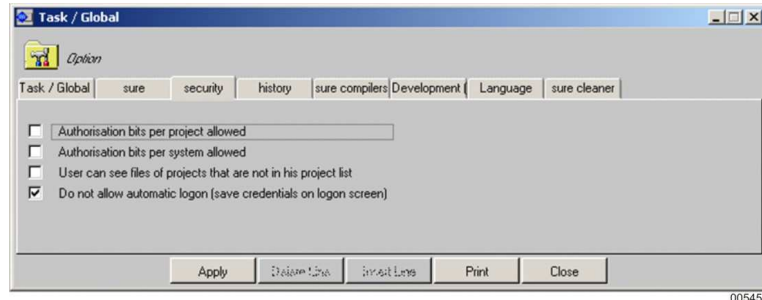
#### Example

Disable the function using the INI-file as follows:

```
[CUSTOMLOGON]
SAVEALLOWED=TRUE      (display logon-button "Save credentials")
SAVEALLOWED=FALSE (hide logon-button "Save credentials")
```

To disable the function using the option in the repository as follows:

1. Right-click the **Environment**.
2. Click **Global Options**.
3. Click the **security** tab.
4. Check the **Do not allow automatic logon (save credentials on logon screen)** check box.



### Considerations

Some types of workstations may require the automatic logon:

- Workstations, defined as build-server for a PC-application, where the building process must be started automatically (without user interruptions).
- Workstations, defined as SURE-email server.

### 34.2.3. Authentication Based On the Windows Account

A user can log on to SURE using his windows account without any password verification in the MCP userdatafile.

#### Detailed Information

The SURE logon authentication using the MCP usercode works fine for MCP developers. But for PC and UNIX developers it is cumbersome. If a PC and UNIX developer is using SURE, his usercode must be defined in the MCP userdatafile. This is very unhandy when password aging is used on the MCP, or when the MCP usercode differs from the Windows account.

This new facility allows a user to log on to SURE using his windows account without any password verification. It assumes that the windows security has validated the password and that the screen saver locks the computer when the user is temporarily absent.

The Windows account must be added as a user ID in SURE so that the appropriate authorization bits can be enabled for that account.

If the Windows account is not known as a user ID in SURE then it must be known in the userdatafile.

If the windows account is defined in the userdatafile, then the old log on method is used: the usercode + password will validated in the userdatafile.

### Example

The dialog shows the logon format. If the entered usercode is equal to the user logged on to the windows system, then the password is not required and the user does not need to be defined in the MCP user data file.

The user definition in SURE requires a special definition setting this windows account check box according to the next example.

The user definition contains an additional attribute named Is Windows Account. If this attribute is set for a user, the usercode may be equal to the windows account logged on with. In this case the password is not validated at the MCP system and therefore the usercode does not need to be entered in the user data file.

**Note:** The default user validation rules apply if the usercode is filled with any other value than the local windows account.

### 34.2.3.1. Anonymous Logon

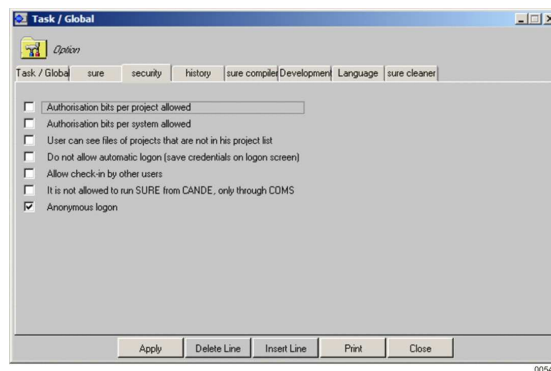
The SURE security is based on a number of different settings:

- The MCP userdata file.
- The WINDOWS logged on account name.
- The SURE user definition "is windows account."

By default, the MCP usercode and credentials are used in the SURE logon dialog. Except for PC developers which may use their WINDOWS account and credentials (if the usercode is marked in SURE as "is windows account").

This mechanism requires that everyone must be defined as SURE user; otherwise, they cannot log on.

The **Global Options** security tab now contains an option named "anonymous logon". If set, every user (in the MCP user data file) can logon to SURE, so it is not necessary to define a usercode in SURE too. These users can only view the contents of the explorer and cannot do any functions because they do not have any employee role.



### **34.2.4. Authentication Based On Kerberos**

SURE supports the automatic logon authentication using Kerberos.

Kerberos is a network authentication system. It is of particular use in securing client-server communications over a network where messages might be intercepted, altered, or replayed. In the case of SURE, we use Kerberos as a secured authentication and logon method where passwords are not sent across the network.

#### **Detailed Information**

SURE Kerberos authentication requires that the users Windows account is mapped to an MCP usercode (this is done in the userdatafile).

If SURE Kerberos authentication is enabled the following happens:

- The logon and authentication is done based on the users Windows account, so the usercode does not have to enter a usercode and password. The SURE logon screen only appears for a few seconds, but the logon itself is done automatically. The Kerberos method enforces that no passwords are sent across the network.
- After the logon, SURE automatically connects the user to his MCP usercode. (the same usercode that the user used for the logon when SURE Kerberos was not enabled).

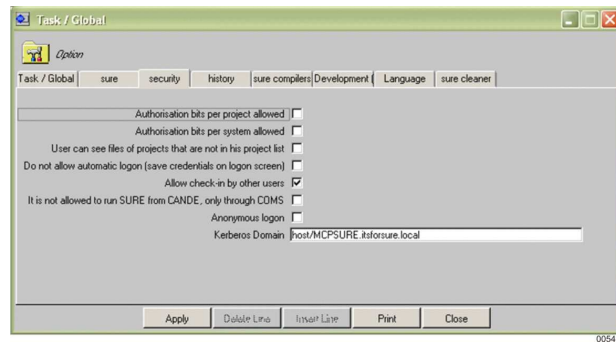
#### **Technical Details and Considerations**

Refer to the *ClearPath Kerberos Security Configuration and Administration Guide* for information on Kerberos security, and how to configure an MCP system to use Kerberos.

As soon as the MCP system is well defined to use Kerberos (you can check that with some example program that are supplied by Unisys using the Security SDK), it is possible to configure Kerberos authentication and logon in SURE.

The SURE Kerberos logon and authentication method is enabled if the Kerberos Domain is defined in SURE. To define Kerberos Domain on the Global Options screen:

1. Right-click the Environment folder.
2. Click **Global Options**.
3. Click **Security** tab.



The Kerberos Domain must be the fully qualified hostname of the MCP host in the realm, preceded with the service-id.

In this example:

- Service-id = host
- Fully qualified hostname = MCPSURE.itsforsure.local

### Reset Kerberos in SURE

It may happen that the current setting of the Kerberos Domain is invalid. In that case nobody will be able to logon to SURE for Windows anymore, because if a Kerberos Domain is defined in SURE (even if it is a wrong name), then the logon must go using Kerberos, but the logon will then fail because of the invalid Kerberos Domain.

Using a batch function it is possible to clear the setting of the Kerberos Domain that allows a regular logon using MCP usercode and password.

- RUN RESPECT/REPOSITORY("RESET-KERBEROS")  
This displays the current setting of the Kerberos Domain.
- RUN RESPECT/REPOSITORY("RESET-KERBEROS DELETE")  
This clears the current setting of the Kerberos Domain.

## 34.3. Employee Functions

It is possible to define employee functions in SURE. Employee functions are linked to usercodes, and are used for routing purposes (this task has to be solved by a user with this employee function) and authorization purposes (users with this employee function are allowed to perform this function).

A usercode can be linked to multiple employee functions, and will inherit all authorizations that are assigned to its employee functions. Defining authorizations for employee functions simplifies the task of assigning a fixed set of securities to a group of users.

**Example**

Consider the following employee functions with their authorization schemes.

<b>Employee-Function</b>	<b>Authorization Scheme</b>
PROGRAMMER	REQUEST
	CHECKOUT
	CHECKIN
	EDIT
PROJECT LEADER	PROBLEMS
	MAINTAIN
	LIST
	DELETE
CHANGE-CONTROL	COMPILE
	RELATE
	LIST

Consider usercode SIMON, which is assigned to two employee functions:

SIMON:	Employee function	PROGRAMMER
	Employee function	CHANGE-CONTROL

User SIMON inherits the authorizations that are assigned to his employee functions and therefore SIMON has the following authorization scheme:

SIMON	CHECKOUT
	CHECKIN
	EDIT
	REQUEST
	RELATE
	LIST
	COMPILE

Changing the authorization map of an employee function will automatically change the authorization maps of all usercodes that are linked to that employee function.

Authorizations can be defined for each usercode or employee function. If authorizations are defined for a usercode, then this usercode will not inherit the authorizations of its employee functions anymore. The users authorizations will take precedence until they are reset. The user will then inherit the authorizations from the employee function.

Usercode INFRA has always all authorizations in the SURE system. It is advisable to delete usercode INFRA from the user data-file or to make this usercode available only to authorized staff.

### 34.3.1. Employee Function for Routing Purposes

The employee function is also used for routing purposes. It is possible to define that a task has to be solved by an employee function. In that case, the task will appear in the outstanding tasks list of all users with that employee function, and any of these users is allowed to make changes for this task.

**Note:** *An employee function can be defined for routing purposes only. In that case, some global (site-level) employee functions are defined for authorization purposes, and local (a project/system level) employee functions are defined for routing purposes.*

#### Example

Consider "programmer" employee function with a security map and "programmer-NP", "programmer-GL" employee functions without security maps.

- User AA is linked to employee functions "programmer" and "programmer-NP."
- User BB is linked to employee functions "programmer" and "programmer-GL."
- Both users have the same authorizations.
- Tasks of project NP have "to handle by" "programmer-NP" and appear in the list of user AA (and not in the list of user BB).
- Tasks of project GL have "to handle by" "programmer-GL" and appear in the list of user BB (and not in the list of user AA).

A user can be linked to one or more teams, and a team can be used for routing purposes. If a task is assigned to a team, then that task appears in the To Do list of all members of that team.

## 34.4. Authorization Scheme

Authorizations can be defined for each usercode or employee function and for each environment. If a user does not have any authorizations, then this user cannot do any update actions in the repository. A usercode inherits the authorizations that are defined for its employee functions for each environment, unless authorizations are defined directly for the usercode on any environment.

#### Example

Consider usercode U1 with employee functions "programmer" and "project-leader", and a repository with three environments: DEVELOP, ACCEPT, and PRODUCTION.



The following authorizations are defined for "programmer":

DEVELOP	SURE/CHECKOUT and SURE/CHECKIN
ACCEPT	NONE
PRODUCTION	NONE

The following authorizations are defined for "project-leader":

DEVELOP	TRANSFER/FROM
ACCEPT	TRANSFER/FROM and TRANSFER/TO
PRODUCTION	TRANSFER/TO

This results in the following authorizations for usercode U1:

DEVELOP:	SURE/CHECKOUT and SURE/CHECKIN, TRANSFER/FROM
ACCEPT:	TRANSFER/FROM and TRANSFER/TO
PRODUCTION:	TRANSFER/TO

If any authorization is defined for usercode U1 itself (regardless of the environment), then none of the authorizations of its employee function ("programmer" or "project-leader") would be inherited.

### **34.4.1. Default Authorization**

It is possible to define default authorizations for each usercode or employee function. These default authorizations are used in the environments where no specific authorizations are defined for that usercode or employee function.

#### **Example**

Consider a repository with three environments DEVELOP, ACCEPT, and PRODUCTION.

Employee function "programmer" is defined with the following authorizations:

DEFAULT	Use SURE/CHECKOUT and SURE/CHECKIN
DEVELOP	Nothing defined
ACCEPT	Use SURE/LIST
PRODUCTION	Use NONE

This results in the following authorizations:

Environment DEVELOP	SURE functions CHECKOUT and CHECKIN (inherited from default level)
Environment ACCEPT	SURE function LIST
Environment PRODUCTION	No functions are allowed.

**Note:** *NONE is not the same as "nothing defined."*

The default authorizations are defined for each employee function. If a usercode is linked to two or more employee functions, then the default securities of an employee function are not inherited by another employee function if nothing is defined for that other employee function on a specific environment.

### Example

Consider a repository with three environments DEVELOP, ACCEPT, or PRODUCTION.

Usercode U1 is linked to three employee functions EMP1, EMP2, and EMP3.

The following authorizations are defined for EMP1, EMP2, and EMP3.

	DEFAULT	DEVELOP	ACCEPT	PRODUCTION
EMP1	AUT1, AUT2	-	AUT3	NONE
EMP2	AUT4	-	NONE	NONE
EMP3	-	AUT5	-	-

This results in the following authorizations for usercode U1.

Environment DEVELOP	AUT1 and AUT2	(default of EMP1)
	AUT4	(default of EMP2)
	AUT5	(specific for EMP3)
Environment ACCEPT	AUT3	(specific for EMP1)
Environment PRODUCTION	NONE	(specific for EMP1 and EMP2, no default for EMP3)

## 34.5. Temporary Authorizations

It is possible to link authorizations to a task using "security-function" task field. An employee function can be entered in this field and the task will then inherit the authorizations of that employee function. When a user defines the task as "current task", then the extra task authorizations are added to the user's normal authorization scheme. When the user changes again from the current task, the user's original authorization scheme will be re-established.

The task authorization gives the user extra a facility in the SURE system, as long as this user is working for that task. The task assignment is an authorized function, and therefore only an authorized person can assign temporary authorizations using a task to a user.

### Example

- Consider Employee function "project-leader" with the following authorizations:

DEFAULT: TASK/APPROVEMENT

- Consider Employee function "programmer" with the following authorizations:

DEVELOP: SURE/CHECKOUT and CHECKIN

ACCEPT: NONE

PRODUCTION: NONE

- Employee function "quick-fixer" with authorizations:

DEVELOP: NONE

ACCEPT: SURE/CHECKOUT and CHECKIN

- Task T1 is defined with security-function "quick-fixer."
- Usercode U1 is linked to employee function "programmer."
- Usercode PL1 is linked to employee function "project-leader."

Project leader PL1 approves task T1 and assigns this task to user U1. The normal authorization scheme of user U1 is:

DEVELOP SURE/CHECKOUT and CHECKIN

ACCEPT NONE

PRODUCTION NONE

If task T1 is defined as “current task” of user U1 then this user inherits the extra task authorization too:

DEVELOP	SURE/CHECKOUT and CHECKIN (from employee function “programmer”)
ACCEPT	SURE/CHECKOUT and CHECKIN (from employee function “quick fixer”)
PRODUCTION	NONE

If user U1 changes again from current task, then his normal authorization scheme re-established.

### 34.5.1. Authorizations for Each System/Project

If a user is authorized to perform a function, then he can use this command for all systems and projects. For example, if a user is allowed to use SURE command check-in, then he can check in files of all projects into SURE. Especially for larger sites, this can be a problem. It often happens that a project leader of project AA does not want the programmers of another project to access the programmers’ files.

The authorization mechanism gives the possibility to define authorizations for each system/project.

Each authorization screen contains an extra field where the system or project name can be entered. If no system or project name is entered, then the defined authorizations are valid for all systems/projects.

#### Example

Consider

- Employee function “programmer-AA” with the following authorizations for project AA: SURE/COMPILE and SURE/CHECKIN.
- Employee function “programmer-BB” with the following authorizations for project BB: SURE/CHECK-OUT, SURE/CHECKIN.

Usercode U1 is linked to both employee functions. This means that he can do the following actions:

- Check in files that belong to projects AA and BB.
- Check out files that belong to project BB.
- Compile files that belong to project AA.

The “multi compile”, “multi relate” and “multi delete” SURE functions are only possible if the corresponding options “multi compile”, “multi relate” and “multi delete” are set with the system/project field empty (thus, for all systems). These authorizations give only access to these commands. A file will only be placed in the compile queue (or related or deleted) if the user is allowed to use function compile for the project of that file.

### Example

Consider:

- Employee function “change-control-AA” with authorization SURE/COMPILE for project AA.
- Employee function “change-control-BB” with authorization SURE/COMPILE for project BB.
- Employee function “change-control-all” with authorization SURE/MULTI-COMPILE (for all systems/projects).

User U1 is linked to employee functions “change-control-AA” and “change-control-all”.

User U1 is allowed to use SURE function multi-compile (inherited from employee function “change-control all.”

In the query function, multiple files are selected to be placed in the compile queue. An extra authorization validation is made for each selected file to check if this file belongs to a project/system for which the user has authorization “compile.” In this example, the files of project AA will pass that check, all other files will not, and are not placed into the compile queue.

Authorizations for each system/project can also be defined for each environment, with a default environment, and for each usercode or employee function.

If system/project authorizations are defined for the usercode itself then this user will inherit the system/project authorizations of his employee functions.

If global authorizations are defined for the usercode itself (global = for all systems/projects), then this user will not inherit the global authorizations of his employee functions but the user will inherit the system/project authorizations of his employee functions.

### 34.6. Defining Users, Employee Functions, and Authorizations

Authorizations can be defined for users and for employee functions. A user will inherit the authorizations of its employee functions if he does not have own authorizations.

#### 34.6.1. User Attributes

The screenshot shows a window titled 'User SIMON' with a tabbed interface. The 'User' tab is selected, displaying a form for user attributes. The form includes fields for Name (Simon Groot), Email Address (SGroot@intaDesign.nl), Department (R&D-TEAM), Initials (SG), Default Environment (DEVELOP-RIS), Profile Destination (NONE), Default Language (English), Employee Function (RIS-SPECIALIST), Team Name (R&D-TEAM), Security Administrator (unchecked), and Role (empty). Below these fields, it shows 'Emp. Function: RIS-SPECIALIST, RELEASE-MANAGER, TESTER', 'Team Name: R&D-TEAM, TestTeam', and 'Projects'. At the bottom, there are buttons for 'Apply', 'Team', 'Employee Function', 'Projects', 'Default Language', 'Default Env', and 'Print'. A small number '005464' is visible in the bottom right corner of the window.

- The **User** tab shows the user attributes.
- The **Security map** tab must be used to define authorizations for this user.
- The **System/Project** tab must be used to inquire for authorization maps for each system/project.
- The **Inherited security** tab shows the actual (inherited) security map of the user.

#### User ID

The usercode that has to be inquired or updated.

#### Name

The full name of the user.

#### Email Address

It is possible to inform the user by email that a new task is added to his To Do list.

#### Department

The department of the user. This department is used in the <define task> function if a user reported a task. The users department is then linked to the task, and various overviews will cumulate amounts of tasks for each department.

#### Initials

This field identifies the user's initials. These initials will appear in the mark-id of source lines that were changed or created by the user. The initials are also placed in the release-id of a source, if the user was the last person who updated that source.

### Default Environment

The environment that is made current when a user starts a SURE program. This default environment can be overridden if the program is started with a task string that identifies another environment.

### Example

Consider usercode U1 with default environment DEVELOP.

If online program RIS/MENU is started without a task string, then the program will be initialized in environment DEVELOP.

If program RIS/MENU is started with task string ACCEPT, then it will be initialized in environment ACCEPT. (For example, RUN RIS/MENU;TASKSTRING="ACCEPT")

Notice that a task string can also be defined for a "COMS program definition."

### Printer Destination

This identifies a station name or system printer name that is associated with the user. All SURE print output that is created by the user will be printed on the defined remote- or line printer. If an online function creates output, then an intermediate screen will be presented where the destination printer can be entered. On that screen, the user's remote printer is prefilled as destination printer.

### Default Language

A user inherits normally the SURE system language (English). It is possible to change the default language for a specific user (if there are other languages available).

### This Usercode is A SECADMIN User

If this option is set for one or more users, then only those users can update the authorization maps of employee functions/usercodes and links employee functions to usercodes.

A user that is defined as SECADMIN user, cannot have an authorization map and cannot be linked to any employee function.

### Summarized

SECADMIN users can define authorizations but cannot perform other functions.

Non-SECADMIN users cannot define authorizations but can perform other functions. (If they are allowed.)

If no user is defined as SECADMIN user, then each user who has authorization-bit "Securities" set can update authorizations.

### Employee Function

These employee functions are linked to the user. The **Employee Function** button must be used to change the employee function list of the user.

### Team

These teams are linked to the user. The **team** button must be used to change the team list of the user.

### Project

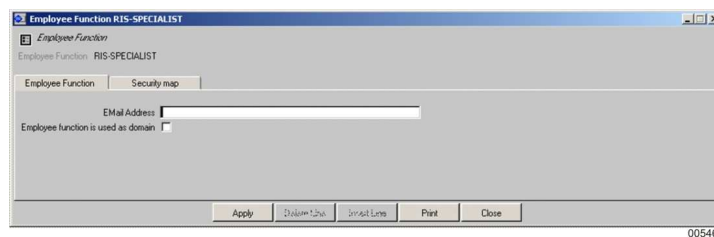
These projects are linked to the user. The Project button must be used to change the team list of the user. The project list identifies the task projects or task systems for which the user can work. The user cannot work on tasks or files that belong to other System/Project. If the user is not linked to any task System/Project, then he can work on all tasks or files.

If one or more projects are linked to any of the user's teams then that team-project-list overrules the individual project list: all members of the same team have the same project list.

### Role

The role for this user, which customizes folders and menus in the SURE Explorer.

## 34.6.2. Employee Function Attributes



The **Security map** tab is used to define authorizations for this employee function.

### Email Address

It is possible to inform the user by email that a new task is added to his To Do list (if the new task is assigned to one of the employee-functions of the user).

### Employee function is used as domain

It is possible to link a file to a domain. In that case, only the users that are linked to that domain (employee function) are allowed to modify the file.

## 34.6.3. Define Authorizations

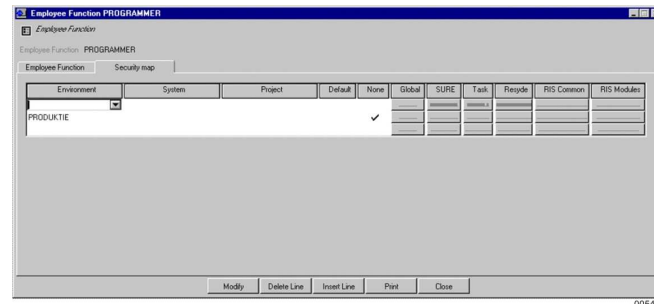
It is possible to define an authorization scheme for each usercode or for each employee function. In both cases, this must be done using **Security map** tab (On the User dialog and on the Employee Function dialog).

It is possible to define authorizations for each environment and for each system or project



This example screen shows two sets with authorizations bits:

- The first line defines the Default authorizations bits for this employee function. The "Environment" field is left empty to address the default environment.
- The second line defines authorization bits for this employee function for environment PRODUKTIE. The "None" option is set to indicate that this employee function is not allowed to do anything in environment PRODUKTIE. All other environments inherit the default authorization bits.



### Explanation of the Columns

#### Environment + System + Project

These three columns identify together a set with authorization bits.

#### Default

If this option is set, then all authorization bits on that line are switched off, and the authorization bits of the default environment will be used (the line will be deleted).

#### None

If this option is set, then all authorization bits will be switched off, but the default authorizations are not used (no authorization bits on this environment).

#### Global/SURE/Task/Resyde/RIS Common/Ris Modules

The authorization bits have to be maintained using these buttons.

## 34.7. Options of the Authorization System

### <Requires Additional UserId for the Work Space>

This option is only applicable for the terminal emulation interface.

This option should be set if multiple users are working under the same CANDE usercode. This can be the case for a specific application system, or for all systems.

This option is not used by the SURE Explorer interface, because in that case, every user has his own private usercode (for identification purposes), and the user will be routed to the correct CANDE work-environment (CANDE usercode/pack) for the application system on which the user is working.

It is possible to set this option for each application system or for each environment:

- Additional UserId for one application system: Dialog System/Properties.
- Additional UserId for all workspaces: Dialog Environment/Properties.

If this option is set for an environment or system, and RIS/MENU is started for that environment or system, then an extra logon screen will appear where the user can enter his individual usercode.

This option should be set for systems or environments where the work usercode is not equal to "\*".

If a user logs on to SURE at an environment where the "additional usercode" option is not set, then the CANDE usercode under which the user logged on is also used as personal usercode.

### Example

Consider a repository with three environments: DEVELOP, TEST and PRODUCTION. The work usercode for TEST is TESTUSCD, and for PRODUCTION it is PRODUSCD. The work usercode for DEVELOP is "\*". The "additional UserId" option is set for environments TEST (all usercodes), and PRODUCTION (all usercodes).

If a user is logged on under usercode TESTUSCD and starts the on-line program, then an extra logon screen will appear. He can enter his individual usercode and password at this screen. For example, BILL/PASS.

When the user passes the logon screen, then the online session works under CANDE usercode TESTUSCD and with SURE usercode BILL.

If the same person is logged on under his personal CANDE usercode (BILL), and he starts the online program for environment DEVELOP, then the extra logon screen is NOT presented. In this case the online sessions works under CANDE usercode BILL with SURE usercode BILL.

### <Requires ACCESSCODE for the Work Space>

This option is only applicable for the terminal emulation interface.

This option can be used if multiple users work under the same CANDE usercode with a personal CANDE accesscode. The accesscode is used by the SURE software for a unique identification.

Notice that the unique identification can also be achieved using option <use extra usercode for the authorizations>.

It is possible to set this option for each application system or environment:

- Requires an accesscode for one application system: Dialog System/Properties.
- Requires an accesscode for all workspaces: Dialog Environment/Properties.

If this option is set for an environment or system, and RIS/MENU is started for that environment or system, and an accesscode is available for the work-usercode, then the individual SURE user ID will be set equal to that accesscode. If no accesscode is available (this can happen if RIS/MENU is started using COMS), then an extra logon screen will appear where the user has to enter an accesscode that belongs to his current work usercode. This usercode and accesscode combination is validated against the user data file.

**<Authorizations Through System>.**

**<Authorizations Through Project>.**

By default, the authorization mechanism works for all projects/systems. It is possible to define SURE authorizations for each system or project.

## **34.8. Who is Allowed to Maintain Usercodes and Authorization Maps?**

If one or more users are defined as SECADMIN user, then only those users can modify authorizations maps and usercodes.

If no user is defined as SECADMIN user then the following rules apply:

- A user with authorization-bit "global: securities" can modify all authorization maps and usercodes.
- A user with authorization-bit "global: securities per team" can do the following:
  - Add a usercode (all security-bits of the user must be reset).
  - Add a team.
  - Modify and delete a team (only if it is one of the team members).
  - Link his usercode to an empty team (a team without any users).
  - Link/Delink other users to one of his teams.
  - Update a usercode (but not the security bits of the usercode and only if the user is one of his team members).
  - Delete a usercode (only if the user is one of his team members).
  - Add an employee function (but the new employee function cannot have security-bit "global: securities").
  - Update an employee function (but the employee function cannot have security-bit "global: securities", and only if that employee function is exclusively in use by members of his own team).
  - Delete an employee function (only if that employee-function is exclusively in use by members of his own team).
  - Link/Delink a team member to an employee-function (only if the employee function does not have security-bit "global: securities").
  - Link/Delink a team member to a project (only if the user did not inherit any project using his teams).

- A user with authorization-bit “global: organization” can do the following:
  - Add a usercode (all security-bits of the user must be reset).
  - Add a team.
  - Modify and delete a team (only if it is one of his teams).
  - Link his own usercode to an empty team (but only if he also has authorization-bit “global: Link Team”).
  - Link/Delink other users to one of his own teams (but only if he also has authorization-bit “global: Link Team”).
  - Update a usercode (but not the security-bits of the usercode and only if the user is one of his team members)
  - Link/Delink a team member to an employee function (only if the employee function does not have any authorizations).
  - Link/Delink a team member to a project (only if the user did not inherit any project using his teams).

### 34.9. Authorization Bits and the Offered Privileges

Column:	Bit + bit number	Subject	Function
Global:	01.Compile	File:compile	Compile a file (local compilation).
Global:	02.Create overlap	File	Allow to create task overlap when a source or file-attributes are modified.
Global:	03.Data Control		Not used in SURE Explorer.
Global:	04.Deliver final		Not used in SURE Explorer.
Global:	05.Delivery		Not used in SURE Explorer.
Global:	06.Domain	Task:transfer	If option “use domains” is set: allow to transfer files of all domains.
Global:	07.File properties		Not used in SURE Explorer.
Global:	08.Link team	Usercode:team	Link or delink a usercode to a team.
Global:	09.Load dasdl		Not used in SURE Explorer.
Global:	10.Monitor	ODT	Allow to give ODT commands
Global:	11.Options (all)	Task-type	Add, update, or delete a task-type.
Global:	11.Options (all)	File-type	Add, update, or delete a file-type.
Global:	11.Options (all)	Printer	Add, update, or delete a printer.
Global:	11.Options (all)	Drop-down box	Add or delete a value to a drop-down box.

<b>Column:</b>	<b>Bit + bit number</b>	<b>Subject</b>	<b>Function</b>
Global:	11.Options (all)	Reference-class	Update the list with reference classes.
Global:	11.Options (all)	Drop-down box	Add, update, or delete a drop-down box.
Global:	11.Options (all)	Macro	Add, update, or delete a macro.
Global:	11.Options (all)	News	Update the news.
Global:	11.Options (all)	Logon	Make folder "Configuration" visible in SURE Explorer.
Global:	11.Options (all)	E-mail	Delete one e-mail from the e-mail queue or purge the total e-mail queue.
Global:	11.Options (all)	System	Add, update, or delete a system.
Global:	11.Options (all)	Project	Add, update, or delete a project (sub-system).
Global:	11.Options (all)	Environment	Add or delete an environment or update the environment properties.
Global:	11.Options (all)	Global Options	Update the Global Options.
Global:	11.Options (all)	Customer	Add, update, or delete a customer.
Global:	12.Organization	Team	Add, update, or delete a team.
Global:	12.Organization	Team:project	Link or delink a team to a project.
Global:	12.Organization	Usercode	Update the usercode attributes (except the authorization map).
Global:	12.Organization	Usercode:empl.function	Link or delink a user to an emp-func (emp-func may not have an auth.map).
Global:	12.Organization	Usercode:project	Link or delink a usercode to a project.
Global:	13.Purge	File:purge	Purge a file or a directory with files out of SURE (physical delete).
Global:	14.Quick fix	File:quick Fix	Apply the quick fix patches or copy the quick fixed file.
Global:	15.Recover	File:undo all changes	Recover (copy) the file from the next environment this current environment.
Global:	16.Reports		Not used in SURE Explorer.
Global:	17.Securities	Usercode	Add, update, or delete a usercode.

## Authorization Mechanism and Log On

---

Column:	Bit + bit number	Subject	Function
Global:	17.Securities	Employee function	Add, update, or delete an employee function.
Global:	17.Securities	Usercode	Make a report of the defined authorization scheme
Global:	17.Securities	Usercode:team	Link or delink a usercode to a team.
Global:	17.Securities	Usercode:empl.function	Link or delink a usercode to an employee-function.
Global:	17.Securities	Usercode:project	Link or delink a usercode to a project.
Global:	17.Securities	Team	Add, update, or delete a team.
Global:	17.Securities	Team:project	Link or delink a team to a project.
Global:	18 Securities for each team	Securities	See bit 17, but for my team members only.
Global:	19.Select macro		Not used in SURE Explorer.
Global:	20.Site function		Not used in SURE Explorer.
Global:	21.Status	Usercode:status	Show the mix-status of another usercode.
Global:	22 System and Project	System	Change the properties of my systems and my projects
Global:	23.Transfer block	Task:transfer block	Block or unblock a task-transfer.
Global:	24.Transfer delink	File:delink from task	Delink a file from a task.
Global:	25.Transfer from	Task:transfer	Transfer a task from an environment.
Global:	26.Transfer from	Task:transfer queue	Add or delete a task to the transfer queue.
Global:	26.Transfer move	File:move to other task	Move a file from a task to another task.
Global:	27.Transfer non domain	Task:transfer	If option "use domains" is set, transfer files that do not have a domain.
Global:	28.Transfer to	Task:transfer	Transfer a task to an environment.
Global:	29.Utility editor		Not used in SURE Explorer.
Global:	30.Work for other projects	Task or File	Allow to work on a file or task of a project that is not in my project list.
Resyde:			Not used in SURE Explorer.

<b>Column:</b>	<b>Bit + bit number</b>	<b>Subject</b>	<b>Function</b>
Ris common:			Not used in SURE Explorer.
Ris modules:			Not used in SURE Explorer.
Sure:	01.Activate queue	Compile-queue	Activate a user defined compile queue, not a baseline queue.
Sure:	02.Assign	File:assignment/checkout	Assign a file to the user that requested the file.
Sure:	02.Assign	File:assignment/checkout	Request a file for a user and assign it to that user.
Sure:	02.Assign	File:assignment/checkout	Undo checkout/assign/request.
Sure:	03 Check In	File:check-in	Check a file in.
Sure:	03 Check In	File:modify	Change the filekind of a source (cobol to cobol74, and so on).
Sure:	04 Check Out	File:assignment/checkout	Request a file for maintenance, assign it to myself, and check it out.
Sure:	04 Check Out	File:assignment/checkout	Check a file out of the repository (the file must be assigned to the user).
Sure:	05.Compile	File:compile	Add or delete a file in a batch compile queue.
Sure:	05.Compile	File:compile block	Block or unblock all batch compilations of the file.
Sure:	06.Copy	File:view	View a file or copy a file from the repository to disk.
Sure:	07.Count		Not used in SURE Explorer.
Sure:	08.Create deltafiles		Not used in SURE Explorer.
Sure:	09.Customer		Not used in SURE Explorer.
Sure:	10.De-activate queue	Compile-queue	De-activate a user defined compile queue, not a baseline queue.
Sure:	11.Delete	File:remove	Delete a file logically (put in recycle bin).
Sure:	11.Delete	File:undo remove	Reactivate the logical deleted file (from recycle bin).
Sure:	12.Driver	File:driver	Link or delink a file to a driver.
Sure:	13.Edit		Not used in SURE Explorer.
Sure:	14.Examine	File:examine	Examine the file immediately for references to other files.

## Authorization Mechanism and Log On

---

Column:	Bit + bit number	Subject	Function
Sure:	15.Find	File:find	Start the find function.
Sure:	16.Guard		Not used in SURE Explorer.
Sure:	17.List	File:list	List a file (using file-properties).
Sure:	18.Load	File:load	Load a file or a directory with files in SURE.
Sure:	19.Multi compile	File:compile	Put a file in the batch compile queue (using multi compile).
Sure:	20.Multi delete	Delete/Rename	Delete or Rename a directory.
Sure:	21.Multi relate	Relate	Add, update, or delete multiple relations.
Sure:	22.New File	File:new	Add (enter) a new file name into SURE.
Sure:	22.New File	File:executable	Add or delete an executable definition for a PC file.
Sure:	23.Object security	File:modify object	Add, update, or delete a multi-objectfile definition to a source.
Sure:	23.Object security	File:modify object	Update the object location or object name of a file.
Sure:	24.Options		Not used in SURE Explorer.
Sure:	25.Patch	File:patch files	Merge a patch-file into the source.
Sure:	26.Print relation info		Not used in SURE Explorer.
Sure:	27.Purge queue	Compile-queue	Purge a compile-queue.
Sure:	28.Relate	Relate	Add, update, or delete a relation.
Sure:	29.Rename	File:rename	Rename a single file.
Sure:	30.Replace	File:replace	Start the replace function.
Sure:	31.Request	File:assignment/checkout	Request a file for a user.
Sure:	31.Request	File:assignment/checkout	Request a file for maintenance, assign it to myself, and check it out.
Sure:	31.Request	File:assignment/checkout	Request a file for a user and assign it to that user.
Sure:	32.Resequence	File:resequence	Resequence a file.
Sure:	33.Reset	File:assignment/checkout	Undo checkout/assign/request.
Sure:	33.Reset	File:assignment/checkout	Undo assign.
Sure:	33.Reset	File:assignment/checkout	Undo request.



<b>Column:</b>	<b>Bit + bit number</b>	<b>Subject</b>	<b>Function</b>
Sure:	33.Reset	File:compile	Remove a file from a batch compile queue.
Sure:	34.Save again		Not used in SURE Explorer.
Sure:	35.Start compile batch	Compile-queue	Start the compile batch for a specific compile queue.
Sure:	36.Start fast batch	Compile-queue	Start the compile fast batch.
Sure:	37.Start PC build	File:set build result	Set the result of the build of the PC file.
Sure:	38.Startjob	File:start-job	Link or delink a file to a start-job.
Sure:	39.Syntax		Not used in SURE Explorer.
Sure:	40.Update	File:modify	Update the file attributes but only with existing values (for example, a known author).
Sure:	40.Update	File:link to task	Link a file to a task.
Sure:	40.Update	File:attachment	Add or delete a file attachment.
Sure:	41.Update object fields	File:modify object	Update the object securities and object privileges of a file.
Sure:	42.UpdateOK	File:modify	Update the file attributes and allow new values (for example, an unknown author).
Sure:	43.View		Not used in SURE Explorer.
Sure:	44.Write	File:write	Write the file (this creates a backup file on the mainframe).
Task:	01.Add a task with self chosen task nr	Task:new	Add a task.
Task:	02.Add a task	Task:new	Add a task.
Task:	02.Add a task	Task:modify	Update task attributes (except solution, doc.impact, power, security-func).
Task:	02.Add a task	Task:assignment	Assign a task to a developer (if option "approve tasks" is not set).
Task:	03.Approve or deny a task	Task:close	Close a task that is not yet approved (if option "approve tasks" set).
Task:	03.Approve or deny a task	Task:deny	Deny a task that is not yet approved (if option "approve tasks" set).

Column:	Bit + bit number	Subject	Function
Task:	03.Approve or deny a task	Task:assignment	Assign a task that is not yet approved (if option "approve tasks" set).
Task:	04.Change the task status to solved	Task:close	Close a task.
Task:	04.Change the task status to solved	Task:deny	Deny a task.
Task:	05.Define and update task groups	Task-group	Add, update, or delete a task-group.
Task:	06.Link temporary privileges to a task	Task:modify	Update task attribute "Security func" (if you are also allowed to add a task).
Task:	07.Delete a task	Task:delete	Delete a task.
Task:	08.Make task dependent on other tasks	Task:dependency	Make a task dependent on another task or delete a task dependency.
Task:	09.Re-activate a task	Task:reactivate	Reactivate a task.
Task:	10.Update fields Solution, Doc.impact	Task:modify	Update the solution or documentation impact of a task.
Task:	10.Update fields Solution, Doc.impact	Task:assignment	Assign a task to a developer (if option "approve tasks" is not set).
Task:	11.Update the task's power attribute	Task:modify	Update the power attribute of a task.

### 34.10. Example Authorization Scheme

A new repository is installed with the following employee functions and authorization schemes:

#### **Employee Function: Sure Administrator**

A SURE administrator is allowed to perform all SURE functions, except to change the authorization maps.

Authorizations: All authorization bits are set, except Global security bit.

#### **Employee Function: Security Administrator**

A security administrator is allowed to change the authorization maps.

Authorizations: Global security.

### **Employee Function: Operator**

An operator starts the evening batch.

Authorizations:

- Global: monitor, status.
- Sure: start-compile-batch, start-fast-batch, start-PC-build.

### **Employee Function: Developer**

Developers are allowed to check out, modify and check in files, and so on.

Authorizations:

- Global: compile, quick fix, status, transfer from.
- Sure: copy, examine, find, get, list, request, re-sequence, reset, save, update, write.
- Task: reactivate, update field solution.

### **Employee Function SENIOR-DEVELOPER**

A senior developer is responsible for the total set of sources of the application, and the links to drivers and start jobs.

Authorizations:

- Global: options, purge, recover.
- Sure: activate-queue, compile, deactivate-queue, delete, driver, enter, guard, load, multi-compile, object-security, patch, purge-queue, rename, replace, startjob, updateOK, update-object-fields, start-fast-batch, relate, multi-relate.

### **Employee Function: Analyst**

An analyst must be able to look into files and to add and approve tasks.

Authorizations:

- Sure: copy, find, list, view.
- Task: add, approve/deny, change-to-solved, delete, reactivate, task-dependencies, update solution.

### Employee function PROJECT-LEADER.

A project leader is responsible for the status of the tasks.

Authorizations:

- Global: create-overlap, link-team, organization, transfer-block, -from, -to, -delink, -move, options, securities-per-team, system-project.
- Sure: assign, request.
- Task: variable-authorizations, power-attribute, task-groups.

**Note:** *This is only an example. You can change the authorization schemes to your own needs, and add or delete employee functions. It is possible (and often necessary) to link multiple employee functions to a usercode.*

In the above example:

- A senior developer must also have Developer employee function; otherwise, the user cannot check out and check in any files.
- A project-leader may also have Analyst employee function.

In many organizations a strict separation of responsibilities is a requirement. SURE meets this requirement because it is possible to define authorizations for each employee function, as shown in the example. On the other hand, if all above mentioned employee functions are linked to one usercode, then that person is allowed to perform all functions in SURE.

## 34.11. Overviews

### 34.11.1.Authorization Overview

This overview is created using Toolbar function:

1. Click **Reports**.
2. Click **Security Overview**.

This overview shows the (combined) authorization schemes that are in use, and the usercodes with their authorization schemes.

The overview consists of three parts:

Part 1: The description of the authorization bits:

- The first page of the overview describes the authorization bits. The bit-numbers are used in the tables of part 2.

Part 2: The authorization schemes that are in use:

- The tables on these pages give the authorization schemes with the authorization bits that are enabled for an employee-function.
- The enabled bits are represented by a star, and the position-number of the star is described in part 1.
- There are also tables for combinations of employee functions, but only if such a combination is in use (assigned to one or more users). The overview labels such a combination as AUTHORIZATION-GROUP-xxx. For example, some users have RELEASE-MANAGER and RIS-SPECIALIST role, and AUTHORIZATION-GROUP-1 table is the combination of those two roles. Only the combinations that are really used are mentioned.

Part 3: The last page gives a list of users and the authorization-tables (of part 2) and vice versa.

- The left-side gives a list of users and for each user the authorization-table.
  - The table has the name of an employee-function if the user is connected to one employee function).
  - The table has the name of a combined table (AUTHORIZATION-GROUP-xxx) if the user is connected to multiple employee functions.
- The right-side gives a list of authorization-tables and for each table the users that have that table.

### Example of Part 3

Overview of authorizations and employee functions.

UserID	EmployeeFunction	EmployeeFunction	UserID
DEVELOP_RIS	COMPILE-BATCH	COMPILE-BATCH	DEVELOP_RIS
ELLEN	TASK-MANAGER	COMPILE-BATCH	NEXTRISRELEASE
FACSG	RIS-SPECIALIST	COMPILE-BATCH	RIS_52
FRANK	AUTHORIZATION-GROUP-1	COMPILE-BATCH	RIS_60
GERARDB	AUTHORIZATION-GROUP-1	COMPILE-BATCH	RIS_70
JAN	AUTHORIZATION-GROUP-1	COMPILE-BATCH	RIS_80
NEXTRISRELEASE	COMPILE-BATCH	RIS-SPECIALIST	FACSG
RIS_52	COMPILE-BATCH	TASK-MANAGER	ELLEN
RIS_60	COMPILE-BATCH	AUTHORIZATION-GROUP-1	FRANK
RIS_70	COMPILE-BATCH	AUTHORIZATION-GROUP-1	GERARDB
RIS_80	COMPILE-BATCH	AUTHORIZATION-GROUP-1	JAN
SIMON	AUTHORIZATION-GROUP-1	AUTHORIZATION-GROUP-1	SIMON

### 34.11.2. Overview of Users, Teams, and Projects

This overview must be created using batch program

```
RUN RESPECT/PRINT( "TEAM-OVERVIEW" );
```

The overview consists of two parts:

- Teams, with their projects and users
- Users, with their projects and teams

#### Example

Overview of teams, with their users and projects.

```
Team    Quality&Support
        No projects linked to this team
User     FACSL           Sandra Lablans
          FACGD           Gijs den Dulk
          FACMS           Marielle scheffer
```

```
Team    HELPDESK
        No projects linked to this team
User     FACMS           Marielle scheffer
```

Overview of users, with their projects and teams.

```
User     FACGD           Gijs den Dulk
Project  BASE,SB,NP,GL,IS,BA,TM,BB,PS,BC,DOC,IF,FI,PH,CV,FB,GN,RA,ST,JX
          JZ,HD,EU,GUI,TA,ATINT,ATVANCE
Team     Quality&Support
```

```
User     FACMS           Marielle scheffer
        No projects linked to this user
Team     HELPDESK,Quality&Support
```

```
User     FACSL           Sandra Lablans
        No projects linked to this user
Team     Quality&Support
```

### **34.11.3. Overview of All the Defined User IDs Plus Attributes**

This overview must be created using batch program

RESPECT/USERATTRIBUTES

The overview can be customized to your own needs. We supply the source plus instructions how you can change the overview.

### Example

UserName:

-----  
ELLEN

                  Name: Ellen van Neste  
          Email address:  
          Department:  
          Initials: EvN  
      User environment: DEVELOP-RIS  
Printer Destination:  
          Language: English  
          Role:  
Security Administrator: No  
      Windows Account: No  
Employee functions: TASK-MANAGER  
          Teams:  
          Project list:

-----  
FRANK

                  Name: Frank van Tiel  
          Email address:  
          Department: R&D-TEAM  
          Initials: FvT  
      User environment: DEVELOP-RIS  
Printer Destination: REND  
          Language: English  
          Role:  
Security Administrator: No  
      Windows Account: No  
Employee functions: RIS-SPECIALIST  
                    ,RELEASE-MANAGER  
          Teams: R&D-TEAM  
          Project list:



## Section 35

# SURE Options

### 35.1. Global Options

To open the Global Options screen, perform the following steps:

1. Right-click **Environment**.
2. Select **Global Options**.

#### 35.1.1. Global Options, Tab: Task/Global

**Task / Global**

Option

Task / Global   sure   security   history   sure compilers   Development   Language   sure cleaner

Name standard for newly entered tasks:

Define tasks per task-group: ☐

Name standard for newly entered tasks per task-group:

Validate newly entered tasks: ☒

Task verification at each update: ☐

Also approve tasks that are readied by a task-approver: ☐

Approve of mastertask also approves subtasks: ☐

Allow to assign a task to multiple individual users: ☒

Lock attachment before modify: ☐

Timestamp layout is YYYYMMDD HH:MM:SS instead of YY-MM-DD HH:MM:SS: ☒

eMail server definition

Can use email: ☒

eMail server IP address:

Port:

SURE installation

Functions valid in test period:

Installation location:

Apply   Delete Line   Insert Line   Print   Close

005467

### **Name Standard for New Tasks**

The Global Option defines a name standard formula that is used to create new task names.

If a new task is entered, then the following checks are done:

- If the "Task Name" field is entered by the user and the user is allowed to choose a new task name, then that will be the name of the new task.
- If the "Task Name" field is entered by the user and the user is NOT allowed to choose new task names, then an error is given.
- If the "Task Name" field is blank and the "Task Group" field is entered, then the new task name is created according to "name standard formula for new tasks for each task-group."
- If the "Task Name" "Task Group" fields both are blank, then the new task name is created according to the "name standard formula for new tasks."
- If it is NOT possible to create a name, then an error is given.

### **Define Tasks for Each Task-Group**

This option is applicable only for the terminal emulation interface.

If this option is set, then the "Task Group" field appears on the task-screen on the terminal emulation interface. In that case, a task must be defined for each task group through the "name standard formula new tasks for each task-group."

This option is not applicable for the PC interface of SURE: if the "Task Group" field is entered on the task-new dialog, then the new task name is determined according to the "name standard formulate for new task for each task-group"; otherwise, the new task name is determined according to the "name standard formula for new tasks."

### **Name-Standard for Newly Entered Tasks For Each Task-Group**

See "Define Tasks for Each Task-Group."

### **Approve Newly Entered Tasks**

A task is approved when it is assigned to a user, a team, or an employee-function through the "to handle by" box. By default, a user who is allowed to add a task or update a task is also allowed to assign a task. If this option is set, then the user must have the authorization "approve or deny a task" to assign a task.

Authorization bit "approve or deny a task" is applicable for functions assign-task and-task, only if the option (approve newly entered tasks) is set.

**Task Verification at Each Update**

This option is applicable only for the terminal emulation interface.

When a file is checked out, the task verification function is started. The user must verify the current task, which he or she is working on, because that current task will be linked to the file that the user is going to check out.

Task verification is normally done once for each session, or when the user changed from the current task, or when the user does not have a current task. If the user works the whole day on the same task and never leaves the RIS/MENU session, then the task verification screen will appear only once.

This option enforces that the task verification screen is given for each update action (function UPD, GETIT, Check-out).

**Also Approve Tasks That are Readied by the Task Approver**

The default method is: if a task is made ready (to approve), and the person who did that is also the task approver, then the task is approved at the same time (with the ready).

If this option is set, the task is not automatically approved and it is a separate action.

**Approve of Master Task Also Approve Its Sub-Tasks**

If the option is disabled (default), then each subtask must be made ready and approved individually. This makes it possible to check the attributes of each subtask before approving is done.

If the option is enabled, then each subtask must be made ready individually, but they can be approved together through the master task. If a subtask is reactivated and again transferred, then it must be approved explicitly.

**Allow to Assign a Task to Multiple Individual Users**

By default, a task can be assigned to an individual user, an employee-function, or a team. In case of an employee-function or a team, all members of that function or team can work on the task. This option allows a task to be assigned to multiple individual users, which is more accurate than assigning a task to an entire team or employee-function.

**Lock Attachment Before Modify**

This option prevents two users from modifying the same attachment (of a task or file) simultaneously.

### Timestamp Layout YYMMDDDD

This option is applicable only for the terminal emulation interface.

The default timestamp layout is: YY/MM/DD hh:mm:ss. For example, 00/12/31 12:34:56.

This option sets the lay-out to: YYYYMMDD hh:mm:ss. For example, 20011231 12:34:56.

### Email Server Definition

This option is applicable only if emails are sent through the default SURE method. If emails are sent through a site-specific library, then this option is not relevant.

Email server is the IP address of the server that is going to handle the emails plus the port that must be used to send the emails from the mainframe to that server.

SURE can send emails to users, customers, or teams about the status of a task. SURE creates the email on the mainframe and sends it to the email server. The email server forwards the email to the correct email address.

The preferred method to send emails is through Unisys OBJECT/EMAIL tool.

See "SURE Site-library" for the site-specific functions.

Using the Unisys OBJECT/EMAIL function requires that the following file is resident on disk:

```
#FILE *INSTALLATION/OPTIONS ON DISK
00000100 EMAIL MAILSERVER=<email server>
```

<email server> is the URL of the email server in your network; for example,  
00000100 EMAIL MAILSERVER=EMAIL.INFRA.NL

### Test Functions

SURE works in three modes:

- Source-maintenance: The stable version of each source is stored in the SURE repository. A developer uses the "check-out" function if he or she wants to modify a source and uses the "check-in" function if the modifications are done. SURE keeps a physical history of each source.
- Source-maintenance + tasks & environments: All functions of "source-maintenance" plus an integrated task tracking mechanism. A task describes the reason why a source has to be modified, and this task is linked to the source when that source is checked out. The (source-modifications because of a) task can be tested in different environments (TEST, ACCEPTANCE). SURE also keeps a logical history of each source. A task is used to transfer changed files to the next environment.

- Source-maintenance + tasks & environments + application deployment: All functions of “source-maintenance + tasks & environments” plus an integrated application deployment mechanism for PC files and ClearPath Enterprise Server files.
  - PC files: Changed files are placed in a “to-build” queue. SURE offers functions to start the correct build procedures and check the result of the build.
  - ClearPath Enterprise Server files: Changed files are compiled during an evening batch. If a copy file is changed, then the calling programs are compiled. The location of the compiled object can be defined in SURE that copies the compiled objects to the correct object-locations. SURE guards the integrity of the total object-environment.

Each function requires an extra license key, but during the temporary test-period of SURE, the customer is free to enable and disable each function through this option.

### Installation Location

This defines the location of the available client software in SURE. It triggers an automatic installation if the available client version differs from the installed client version.

## 35.1.2. Global Options, Tab: SURE

The screenshot shows the 'Task / Global' dialog box with the 'Option' tab selected. The 'sure' sub-tab is active, displaying various configuration options for SURE. The options are organized into sections: 'Printer Destination', 'SURE site library', 'Start screen for terminal emulation interface', 'Save source directory', 'Securitymode for compiled objects (default = PUBLIC IO)', 'ReleaseID prefix', 'Options for RESPECT/SURE/SECURE', 'Secure pack', 'Dump option', and a list of checkboxes for various SURE features.

Fields and options visible:

- Printer Destination: [Dropdown menu]
- SURE site library: [Text field: OBJECT/RESPECT/LIBRARY/SITE\_FUNCTIONS ON IDR]
- Start screen for terminal emulation interface: [Text field]
- Save source directory: [Text field]
- Securitymode for compiled objects (default = PUBLIC IO): [Text field]
- ReleaseID prefix: [Text field: ITSforSURE]
- Options for RESPECT/SURE/SECURE:
  - Secure pack: [Dropdown menu: IDR]
  - Dump option: [Text field]
- Checkboxes:
  - The test object will be removed after saving the source in SURE ☒
  - Version line will be added on the first line of each source ☒
  - SURE also handles UNISYS cataloging ☐
  - SAVE the source in SURE after an EDIT which placed the source on disk ☒
  - Delivery mechanism works in SURE style ☐
  - Use file system work environment (instead of task system) ☐
  - Use task-field 'next-release' to update the release number of its system ☐
  - Compile queues are semi-permanent with dynamic usage ☒
  - A quick-fixed source is also placed in the compile-fast queue ☐
  - Put task name in Markid field in changed lines of source ☐
  - Use SURE method to merge patchfiles (in stead of system/patch) ☐

Buttons at the bottom: Apply, Delete Line, Insert Line, Print, Close.

005468

### Printer Destination

This option specifies the default remote printer (Unisys PRINT/REPRINTS) to be used for the SURE output. This default remote printer can be overruled for each usercode.

### SURE Site Library

This library overrides the default SURE function with site-specific functions. The following functions can be customized to your own needs:

- Site-specific procedures to determine the object-name from a source-name and vice-versa. The default method in SURE is equal to the CANDE method: the object-name = OBJECT/<source name>. Refer to the chapter "Object Names."
- Site-specific usercode and password logon validation. The default SURE method uses the userdatafile.
- Send SURE emails through a site-specific email handler.
- Site-specific procedure to scan (examine) sources for references to other files with the purpose to create a data dictionary. Refer to Section 25, "(Examine) Extracting Source Information," for more information.
- Site actions for each event: The library contains an event procedure that is called when a log-entry is added. A site can customize this event procedure so that specific actions are executed for each event. By default no specific actions are done.

An example site library is delivered with each release. The name of this example library is EXAMPLE/RESPECT/LIBRARY/SITE\_FUNCTIONS and it contains example procedures for each of the above-mentioned functions. You can use these example procedures as a starting point. They work identical to the default functions in the SURE software itself.

Use the following steps to enable a site-function:

- Change the name of the example site library to your own needs, so that it will not be overwritten when you install a future SURE release.
- By default, the example procedures in the example site library are not exported. You must export the procedures that belong to the site-function that you want to use, according to the following table.

Function	EXPORT Procedures
Determine object names, only for source files	GIVE_SOURCE_NAME GIVE_OBJECT_NAME
Determine object names for all file kinds	GIVE_SOURCE_NAME_ALWAYS GIVE_OBJECT_NAME_ALWAYS
Usercode and password validation	SITE_USERDATA
Send emails	SITE_SEND_EMAIL
Event trigger	SITE_EVENT
Scan (examine) sources	SITE_EXAMINE

- Modify the procedures in the site library to your own needs.
- Compile the site library and deploy it to the correct location where it is visible for the SURE software.
- Enter the full name of the site-library-object (with usercode and pack) in the SURE site library option as illustrated in the above screen.
- Stop the entire SURE system (check that database INFDB is stopped).
- Restart SURE.

The names of the site-functions that you want to use can be defined in the RESPECT/TITLES configuration file to improve the startup performance of the SURE system. Refer to “(Installation) RESPECT/TITLES” in Section 8 for details.

### **Start Screen for Terminal Emulation Interface**

This option is applicable only for the terminal emulation interface.

The RIS/MENU screen can be customized to your own needs. Non-RIS users have a customized RIS/MENU screen, which hides all the items that are not relevant and only confusing.

### **Save Source Directory**

If this option is set, then a file is not removed from the disk after check in.

- If the value of this option is “USERCODE,” then the file is left under the usercode where it was modified.
- If the value of this option is a directory, then the file will be changed to that directory.
- If this option is empty, then the file will be removed after checked in.
- In all cases, the new version of the file is saved in SURE.

Refer to the chapter “Check-In.”

### **Security Mode for Compiled Objects**

The default security mode for compiled objects is PUBLIC IO. This option overrides the default with a site-specific value.

### **ReleaseID Prefix**

SURE puts information in the releaseid of the compiled objects, so that it is easier to determine how an object is created. The releaseID prefix is the first part of the releaseid.

### Example releaseID

```
lfile *object/ris/menu :releaseid
#RUNNING 6528
#?
ON IDRD
*OBJECT : DIRECTORY
. RIS : DIRECTORY
. . MENU : DMALGOLCODE
        RELEASEID="ITSforSURE: DEVELOP-RIS 133.1;
                Descriptionfile: INFDB_2_DMS491; Compiler: ALGOL-050.1A.3"
#
```

The releaseID prefix of this example is "ITSforSURE."

### Secure Pack

The mainframe batch procedure "backup files in maintenance" copies all files that are checked out by a developer to tape. This procedure should run on a daily basis, and it protects the developer's last changes.

This procedure is optional: many sites have another backup procedure to protect the work of the developers.

By default, the checked out files are copied to tape, but this option enforces that the files are copied to another pack.

Notice that a developer, who maintains a file through the SURE Explorer, always has the last changes locally available on his or her PC. Only when a file is maintained through CANDE on ClearPath Enterprise Servers, it is recommended to backup the source on a daily basis.

Notice that it is possible to backup files or directories with files that are not known in SURE. This can be done by using the file-type option, "Enter only a file-name," where files-to-be-secured are made known to SURE.

See "RESPECT/SURE/SECURE" in Section 26, "Backup of Maintenance Sources."

### Dump Option

By default, the secure job uses the COPY statement.

Use this option for extensions, such as "& COMPARE" or "& CATALOG."

See "RESPECT/SURE/SECURE" in Section 26, "Backup of Maintenance Sources."



**The Test Object will be Removed After Saving the Source in SURE**

If this option is set, and an object of this source is available under the usercode where the source is checked out, then that object will be removed as the source is checked in.

Refer to the chapter "Check-In."

**Version Line will be Added on the First Line of Each Source**

If this option is set then an extra comment line with version information is added to the source as the file is checked in. The new version-line becomes the first record of the source and gets sequence number zero.

If the first source record has sequence number zero, and that record is not a version comment line, then the source will not get a version comment line.

This option is not applicable for PC files.

The layout of a version comment line is:

```
% VERSION nnn.nn;    SAVED yyyyymmdd hh:mm:ss
    nnn.nn = the file-version number of the source.
    yyyyymmdd hh:mm:ss = the moment the file was checked-in.
```

Refer to the chapter "Check-In."

**SURE Handles Unisys Cataloging**

Set this option if Unisys catalog system is used.

**SAVE the Source in SURE after an EDIT That Placed the Source on Disk**

This option is applicable only for the terminal emulation interface.

The EDIT command may copy the maintenance copy out of the repository. Setting this option will cause a SAVE in this situation. Note that this option may cause a considerable amount of created match files.

Refer to the chapter "Check-In."

**Delivery Mechanism Works in SURE Style**

This option is applicable only for the terminal emulation interface.

This option allows source files to be delivered by SURE. By default, delivery creates repository dump files (source files) that are loaded in SURE repositories.

### **Use Work Environment of File-System (Instead of Task-System)**

The work-environment identifies the usercode and family where maintenance is done for a system. If the usercode equals to "\*", then the work-environment consists of all usercodes on that family. The work-environment also defines the location of copy files.

Each system has its own work-location, but in many cases, the work-locations of two different systems are equal.

Each file is linked to a system, and each task is also linked to a system (through the task project). By default, the system of the task is used to determine the work-location. All files that are maintained because of the task are checked out in a CANDE directory that meets the work-location of the task's system. This is also the case for files that have another system with another work-location.

If this option is set, then each file is checked out in a CANDE directory that meets the work-location of its own system. If a file of another system is checked out, then that file will be placed in another CANDE directory (if the work-location of that file differs from the work-location of other files).

This option can be used in combination with the "Use resource versions" option.

### **Use Task-Field "next-release" to Update the Release Number of its System**

If this option is set, then the release number of a system can be updated through the "next-release" field on the task screen. The release number on the higher environments is updated when the task is transferred.

### **Compile Queues are Semi-Permanent with Dynamic Usage**

Set this option when the compile queue does not have to be active on each environment before a task of that queue is transferred.

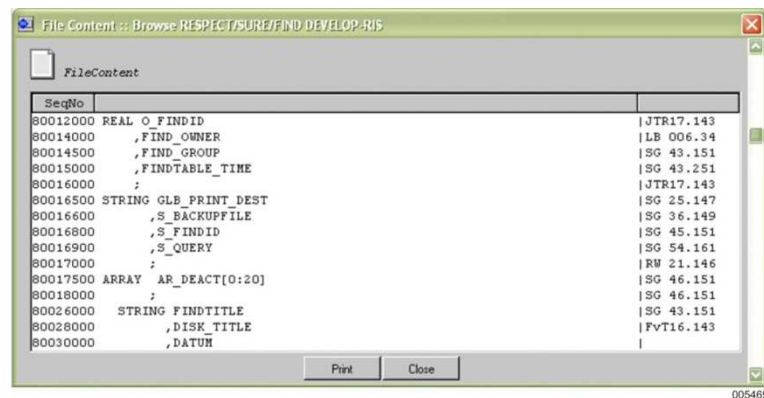
### **A Quick-Fix Source is Also Placed in the Compile-Fast Queue**

A quick fixed source is also placed in the compile-fast queue, if this option is set.

### **Put Task Name in Markid Field in Changed Lines of Source**

This option is relevant only for MCP-sources that support a markid field, such as COBOL and ALGOL sources.

When a source is checked in, the markid of the changed records is updated with the initials of the developer, the version-id of the source, and (optionally) the system-release-number or the year.

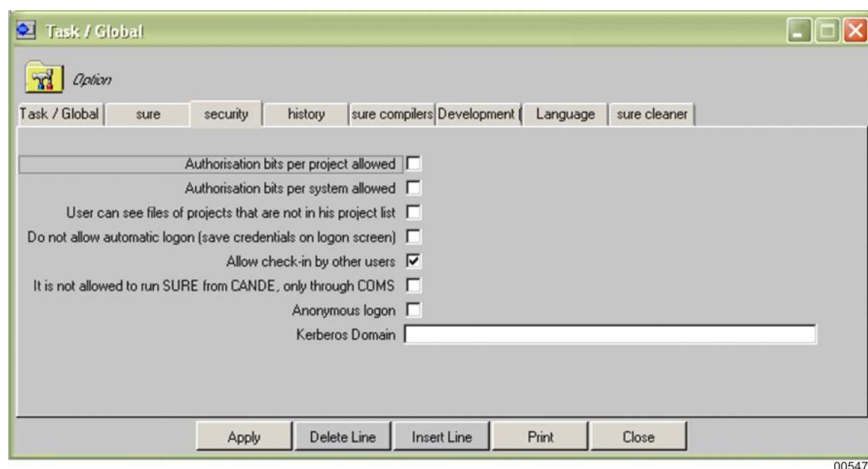


This option puts the task name in the markid of the changed lines.

### Use SURE Method to Merge Patch Files (instead of system/patch)

Refer to Section 21, "Patch Files," for more information.

## 35.1.3. Global Options, Tab: Security



### Authorization Bits For Each Project Allowed

This enables the possibility to define authorization bits for each project.

Refer to Section 34, "Authorization Mechanism and Log On," for more information.

### Authorization Bits For Each System Allowed

This enables the possibility to define authorization bits for each system.

Refer to Section 34, "Authorization Mechanism and Log On," for more information.

### **User Can See Files of Projects That are Not in His Project List**

If this option is enabled, then all file names of all projects are visible in the SURE browser.

If this option is disabled, then the user's project-list works as a filter: only file names of projects that are in the project-list are visible.

### **Do Not Allow Automatic Logon (Save Credentials on Logon Screen)**

This option disables the possibility to save credentials at logon. Notice that some workstations require an automatic logon:

- Workstations, defined as build-server for a PC-application, where the building process must be started automatically (without user interruptions).
- Workstations, defined as SURE-email server.

### **Allow Check-In By Other User**

This option allows the check in to be done by another usercode than the user who did the check out.

Refer to the chapter "Check-In."

### **It is Not Allowed to Run SURE From CANDE, Only From COMS**

This option is applicable only for the terminal emulation interface.

This option prohibits running SURE from CANDE. In this case, a COMS window has to be defined that runs the online program.

### **Anonymous Logon**

By default, each usercode that is used to log on to SURE must be defined in the Unisys userdatafile as well as in SURE itself. If this option is set, it is not necessary that the user is defined in SURE. Obviously, such a user cannot do any updates in the repository, as he is not authorized (because the user ID is not defined).

Refer to "Anonymous Logon" in Section 34 for details.

### **Kerberos Domain**

This enables the Kerberos log-on and authentication method.

Refer to "Authentication Based on Kerberos" in Section 34 for details.

### 35.1.4. Global Options, Tab: History

Task / Global

Option

Task / Global sure security history sure compilers Development Language sure cleaner

Definition for historical records in repository

	Days	Min Entry	Max Entry
Archive log information	50	20	40
Archive file task history			
Archive solved tasks			

Parameters for RESPECT/SURE/CLEANER

	Days
Remove datafiles after	
Remove sources files after	
Remove object files after	

Apply Default Install Print Close

005471

#### Archive Log Information

This defines the amount of days before the log-information is expired. At least <min entries> are kept, but at the most <max entries>. If nothing is entered, then the info expires. Expired information is removed at the next run of RESPECT/SURE/MATCH.

#### Archive File-Task History

This option defines the amount of days before a solved task indication in the file-history is expired. At least <min entries> are kept, but at the most <max entries>. If nothing is entered, then the info expires. Expired information is removed at the next run of RESPECT/SURE/MATCH.

#### Archive Solved Tasks

This option defines the amount of days before a solved task is expired. If nothing is entered, then the task expires. Expired information is removed at the next run of RESPECT/SURE/MATCH.

#### Parameters for RESPECT/SURE/CLEANER

These options define how long files will be kept on disk while they are not accessed by any program or any user.

### 35.1.5. Global Options, Tab: SURE compilers

Task / Global

Option

Task / Global sure security history sure compilers Development Language sure cleaner

Algol

DCAlgol

DMAlgol

Cobol

Cobol74

Cobol85

Sort

Fortran77

Pascal

C

Apply Delete Line Insert Line Print Close

005472

#### SURE compiler Names

SURE provides an option to substitute the default used compiler names. Deviating compiler names can be defined at a global level (for all environments in the repository) or for each environment. This makes it possible to install new Unisys DMS releases for each environment.

Refer to "23.9 Compiler Names" for more information.

### 35.1.6. Global Options, Tab: SURE cleaner

Task / Global

Option

Task / Global sure security history sure compilers Development Language sure cleaner

UserCode	PackName
INFRA	IDRD

Apply Delete Line Insert Line Print Close

005473

#### SURE cleaner

This option allows multiple usercode/pack combinations to be excluded from the cleaning mechanism.

See "RESPECT/SURE/CLEANER" in Section 27.

## 35.2.Maintain Printer Destinations

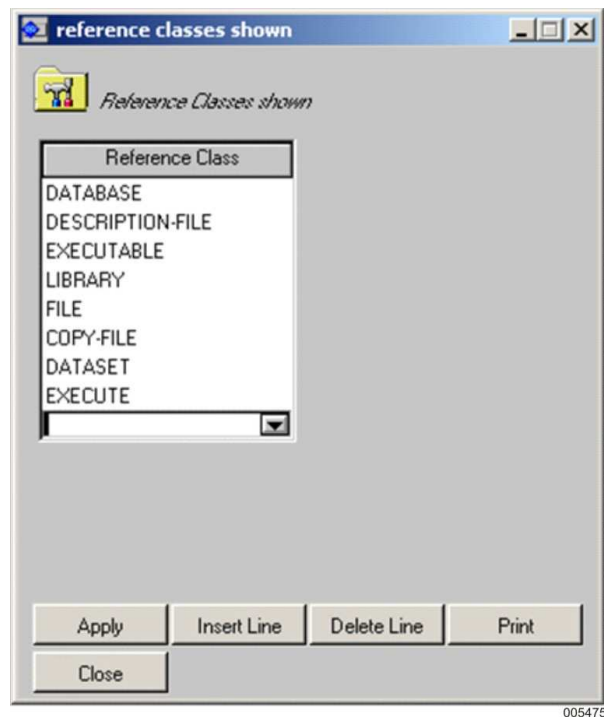


This option defines remote printers and optionally the transform name, which may be used in a transform library.

For each printer type, it is possible to define a page length and a page margin.

(Reference "Unisys PRINT/REPRINT")

## 35.3.Maintain Reference Classes



This option defines a set of relation classes that are used in the "References" and "Used By" folders to show a list of file references.

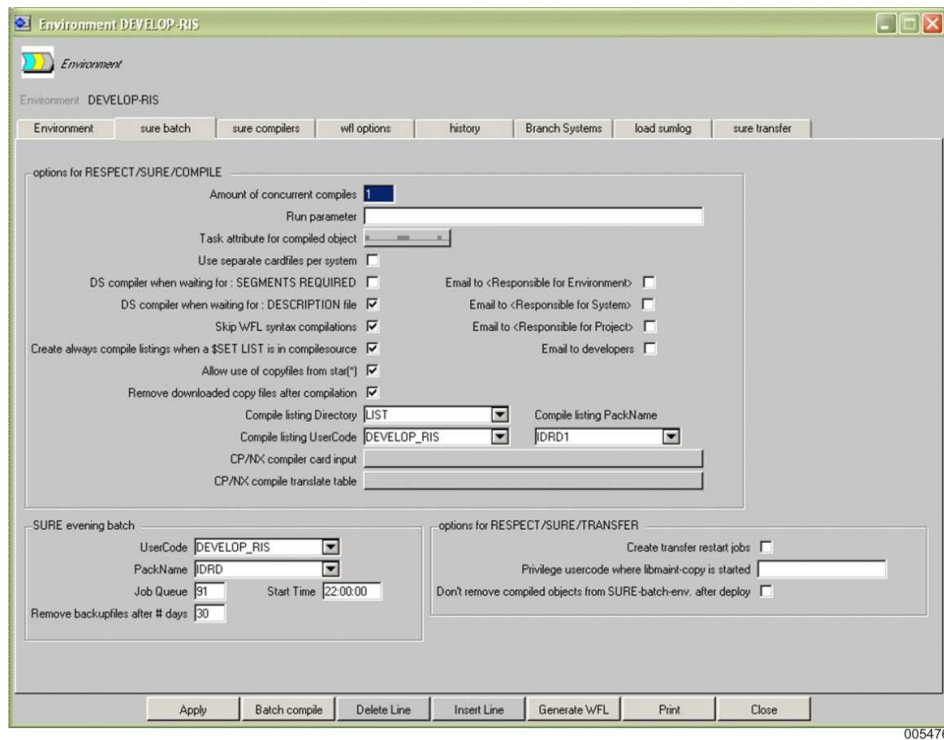
The order in which the classes are entered here defines the order in which they will be displayed in the "References" and "Used By" folders.

Refer to "Reference (Properties/Info, Expand)" in Section 31.



## 35.4. SURE Compile Options

The SURE compile options are defined for each environment. This allows using different options for development and production.



### Amount of Concurrent Compiles

Changed files are added to the compile queue and are compiled during the evening batch. The RESPECT/SURE/COMPILE program controls these compilations. This program can control up to 64 concurrent compilations, but the preferred setting is less than 10.

The default value is 1.

The amount of concurrent compilations can be changed while RESPECT/SURE/COMPILE is running: Give ODT command HI <number> to the mix-number of RESPECT/SURE/COMPILE, and extra compilations are started. The compilation selection process can be stopped through HI 1000.

### Run Parameter

Extra run parameters for RESPECT/SURE/COMPILE.

Refer to Section 47, "Program index," for a complete list of run parameters.

The queue that RESPECT/SURE/COMPILE has to process must NOT be entered in this run parameter.

### Example

Suppose that the compile program must be started as follows:

```
RUN RESPECT/SURE/COMPILE( "NORMAL-MODE NO-LISTINGS" )
```

In that case, the run parameter must be NO-LISTINGS.

### Task Attribute for Compiled Object

This option sets task attribute "OPTION" for the compiled object.

See "RESPECT/SURE/COMPILE" in Section 47.

### Use Separatecard Files for each System

If this option is set, then a separate card file is required for each system.

### DS Compiler When Waiting for Segments Required

Setting this option will DS the compiler if no disk space is available. Note that the compilations will be aborted in that situation. This option is RESET by default, because if there is not enough disk space, then all following compilations are going to be DS-ed too. In this case, it is often better to wait and solve the problem.

See "RESPECT/SURE/COMPILE" in Section 47.

### DS Compiler When Waiting For No File Description

Setting this option will DS the compiler if it is waiting for any type of description file. Note that the compilations will be aborted in that situation. This option is SET by default. Perhaps a developer made a mistake and misspelled the name of the database in his program. If the compiler was not DS-ed then the evening batch will keep waiting for this single compilation. In this case, it is often better to DS the compilation. The DS will be mentioned in the compilation overview, and the source gets compile status abort.

See "RESPECT/SURE/COMPILE" in Section 47.

### Skip WFL Syntaxerrors

The RESPECT/SURE/COMPILE program initiates and controls the compilations of the evening batch.

When a file is changed in an environment, it is automatically placed in the compile queue of that environment. This rule counts also for job symbols (WFL's).

RESPECT/SURE/COMPILE can check the syntax of the jobs that are in the compile queue. At this moment, it is not possible to check all WFL statements correctly. If a job contains accesscode assignments, then it is not possible to compile such a job without errors. This may trigger all kinds of waiting integrity chains.

If this option is set, then the syntax of a WFL is not checked.

**Always Create Compile Listings When a \$SET LIST is in Compiled Source**

The compile control card \$ SET LIST instructs the compiler to create listings. If this option is disabled, all backup files will be removed. Note that this \$ SET LIST option can be set not only by the user program but also in the compiler CARD file.

See "RESPECT/SURE/COMPILE" in Section 47.

**Allow Use of Copy Files From Star (\*)**

This option instructs RESPECT/SURE/COMPILE to use copy files visible from the \* directory. Resetting this option instructs RESPECT/SURE/COMPILE to download copy files from the repository into its running directory.

See "RESPECT/SURE/COMPILE" in Section 47.

**Remove Downloaded Copy Files After Compilation**

This option instructs RESPECT/SURE/COMPILE to remove the previously downloaded copy files at completion.

See "RESPECT/SURE/COMPILE" in Section 47.

**Compile Listing Directory**

This option works in combination with the listing-option. Through the listing-option, it is possible to define that the compilation listing must be saved on disk. The directory where the compilation listing is saved must be defined through this option. The compile listing directory consists of three parts: a usercode, a prefix, and a pack-name. The prefix will always get an extra prefix: the name of the compile-queue or FILE-CONTROL. The name of the compilation listing will be:  
(<usercode><prefix><file-name> ON <pack>.

Refer to "23.11 Compile Listings" for more information.

**ClearPath Enterprise Server Compiler Card Input**

RESPECT/SURE/COMPILE starts the compiler with a primary and secondary input file (the "card-file" and the "tape-file"). The tape-file is file-equated to the source that has to be compiled and this tape-file is merged with the card-file. The card-file contains dollar options that are used to control the compiler. The card-file must contain at least the \$ SET MERGE option. The organization is free to declare its own dollar options in the card-file. Refer to Section 23, "Compilation and Object Files," for more information.

**ClearPath Enterprise Server Translate Table**

Many sites have different names for databases, libraries, and files in production and in development. A source in development contains the develop names, and if the source is released for production, then the develop names are translated to production names. This concept is supported by SURE through "compile translation tables." The source loaded in SURE used develop names. If the source must be compiled for production, then that source is automatically translated through a predefined compile translate table before it is offered to the compiler.

### Create Transfer Restart Jobs

Setting this option causes the RESPECT/SURE/TRANSFER to create a restart job and save the objects in a separate directory. This option allows recreation of the transfer tape.

See "RESPECT/SURE/TRANSFER" in Section 47.

### Privilege Usercode Where Library Maintenance Copy is Started

The RESPECT/SURE/TRANSFER creates a deploy-job that copies the compiled objects to their final object-locations. This is done through a library maintenance copy, and that requires a privilege usercode, because the usercode of the object-location differs most of the times from the SURE batch usercode.

If this option is empty, then the SURE batch usercode must be a privilege user (PU); otherwise, the library maintenance copy will fail. You can define a privilege usercode in this field to avoid that all SURE-batch usercode have to be PU. The RESPECT/SURE/TRANSFER will assume the entered usercode, so it is not necessary to define a password.

Box: email

With the options in this box, you can define who must obtain emails in the case of syntaxerrors.

<responsible for environment> must be defined by clicking the Environment properties and then clicking the Environment tab.

<responsible for system> must be defined through System-project properties.

<responsible for project> must be defined through Project properties.

Box: SURE evening batch

This box defines the usercode and pack where the SURE-evening batch must run. If you want to schedule the evening batch automatically for the next day, you have to define the job-queue and start-time as well.

Remove backup files as # days, defines the period how long the overviews that are created by the SURE batch have to be kept on disk (in directory (<batch-usercode>)BATCHOUTPUT/=).

## 35.5.Compile Options for each Project or System

These options are defined for each environment and for each project.

The screenshot shows a Windows-style dialog box titled "Project RESPECT RESPECT". It has a "Project" icon and two tabs: "Project" and "Environment". The "Environment" tab is selected. At the top, it displays "System RESPECT", "Project RESPECT", and "in environment : RIS\_491". The main area contains several sections of options:

- PC build commands:** A text input field.
- PC build replace table:** A text input field.
- eMail to User:** A dropdown menu with a button to open the list.
- Add project name to source name:** A checkbox (unchecked).
- Source maintenance via patch files:** A checkbox (unchecked).
- Resource parameters:**
  - Resource definition:** A dropdown menu.
  - Default resource version:** A dropdown menu.
- SURE Compile options in : RIS\_491:**
  - Compiler/Generator runs with charge code:** A text input field.
  - Task Attributes assigned to compiled object:** A text input field.
  - Listing:** A text input field.
  - CP/NX compiler card input:** A text input field.
  - CP/NX compile translate table:** A text input field.
- Delivery parameters in : RIS\_491:**
  - Licence key for delivery tape:** A text input field.

At the bottom, there are four buttons: "Apply", "Build & Replace", "Print", and "Close". The ID "005477" is visible in the bottom right corner.

### Compiler/Generator Runs with Charge Code

Unisys charge code is used for the compiler. Use this option, if the compiler output must be routed using this charge code.

See "RESPECT/SURE/COMPILE" in Section 47.

### Task Attributes Assigned to the Compiled Object

This definition allows defining task attributes for the compiled object or for the compiler at the project or system level.

Refer to "23.4 Compile Options."

### Listing

This definition allows defining extended compile listing options at the project or system level.

Refer to "23.11 Compile Listings."

### **ClearPath Enterprise Server Compiler Card Input**

The RESPECT/SURE/COMPILE starts the compiler with a primary and secondary input file (the "card-file" and the "tape-file"). The tape-file is file-equated to the source that has to be compiled and this tape-file is merged with the card-file. The card-file contains dollar options that are used to control the compiler. The card-file must contain at least the \$ SET MERGE option. The organization is free to declare its own dollar options in the card-file. Refer to Section 23, "Compilation and Object Files," for more information.

### **ClearPath Enterprise Server Translate Table**

Many sites have different names for databases, libraries, and files in production and in develop. A source in development contains the develop names and if the source is released for production then the develop names are translated to production names. This concept is supported by SURE through "compile translation tables." The source loaded in SURE used develop-names. If the source must be compiled for production, then that source is automatically translated through a predefined compile translate table before it is offered to the compiler.

## **35.6.Compile Options for each File Type**

These options are defined for each environment and for each file-type.

### **Define Extra Attributes for the Compiled Object**

This definition allows defining task attributes for the compiled object or for the compiler at the file-type level.

Refer to "23.4 Compile Options."

### **Define Compile-Listing Options for the File Type**

This definition allows defining extended compile-listing options at the file-type level.

Refer to "23.11 Compile Listings."

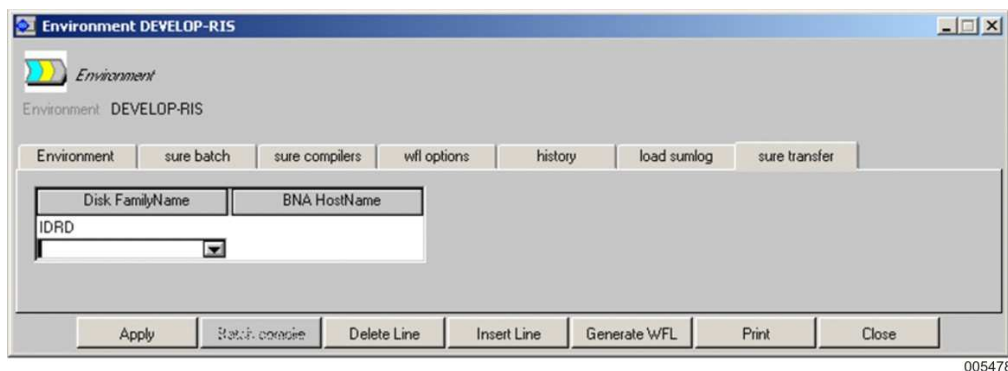
### **ClearPath Enterprise Server Compiler Card Input**

The RESPECT/SURE/COMPILE starts the compiler with a primary and secondary input file (the "card- file" and the "tape-file"). The tape-file is file-equated to the source that has to be compiled and this tape-file is merged with the card-file.

The card-file contains dollar options that are used to control the compiler. The card-file must contain at least the \$ SET MERGE option. The organization is free to declare its own dollar options in the card-file. Refer to Section 23, "Compilation and Object Files," more information.

## 35.7. Transfer Options

The SURE transfer options are defined for each environment allowing the use of different options for development or production.



The object-location of a file gives the destination of an object that was compiled during the evening batch.

By default, the compiled objects are transferred to the object-locations through tapes. Transfer tapes are created for each destination family: all objects with the same object pack (destination family) will be copied on the same tape. The operator can mount the tape and copy the object from the tape to the object-pack.

If a family-name is entered in this table, then the objects are not transferred through the tape to that family, but transferred directly with library maintenance from pack to pack. If the hostname is entered, then the objects are copied through BNA to that host.

If a pack is linked to multiple hosts, then the objects for that pack are copied to all those hosts, unless a specific object-host was entered for a file.

See "RESPECT/SURE/TRANSFER" in Section 47.

## 35.8. Resource Definitions

The content of resource definition tables is maintained for each environment.

A default repository environment is assigned to each project. If a resource definition table is activated, resources, belonging to this project, will by default be retrieved from this environment.

The ADD function creates a new resource definition table. It may be attached to a new identifier or an existing identifier.

The DEL function removes all definitions for the current environment. When all environments are cleared, the resource definition is deleted completely.

Refer to Section 15, "Copy Files," for more information.

## **35.9. Runinfo Usercodes**

Information from ClearPath Enterprise Server Series SUMLOG files can be loaded into SURE. By default, these run statistics are loaded for a file if the usercode and pack of the object that has run are identical to the defined object-usercode and object-pack in SURE.

Through this option, you can limit the statistics sampling to one or more usercodes that started the object. Notice that an object can be placed on disk under usercode AA, but this object can be started from any other usercode through the command `RUN (AA)<object name>`.

Refer to Section 28, "SUMLOG Information," for more information.



# Section 36

## Versioning

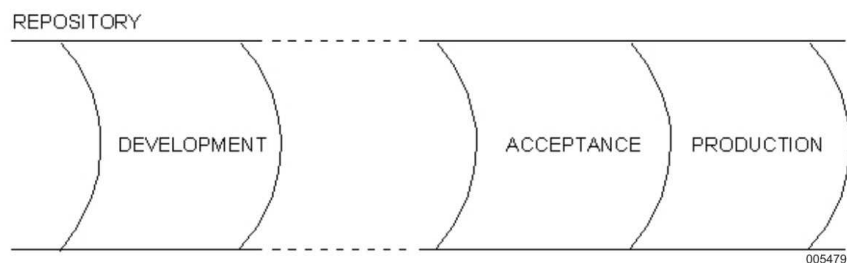
### 36.1. Terminology

#### Repository

The repository is an INFDB database. Therefore, every physical INFDB is a separate repository.

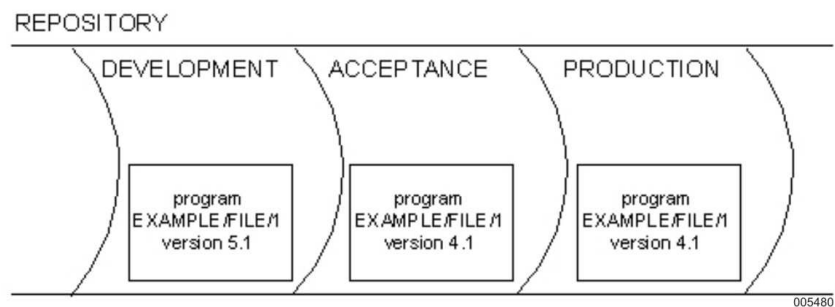
#### Environment

The repository is divided in different environments such as DEVELOPMENT, ACCEPTANCE, and PRODUCTION. The user organization defines the number and names of the environments. This configuration can be adjusted at any time. Every environment contains a complete set of definitions and sources.



#### Version

A Version is an instance of a definition or source in any environment. So, there is a distinct difference between a version, which is an instance of one thing, and an environment, which is an instance of a complete application system.

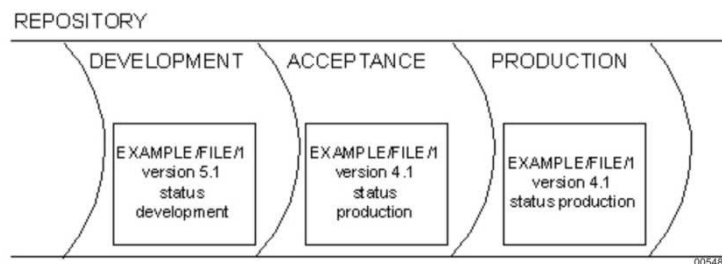


### Versioning

Versioning is the mechanism, which supports different versions and environments in SURE. Therefore, versioning is not a thing but the total of procedures and technical implementations supporting transfer and control of software.

### Status

The status of a definition or source determines which version of that definition or source is available in which environment. A specific version can be available in multiple environments.

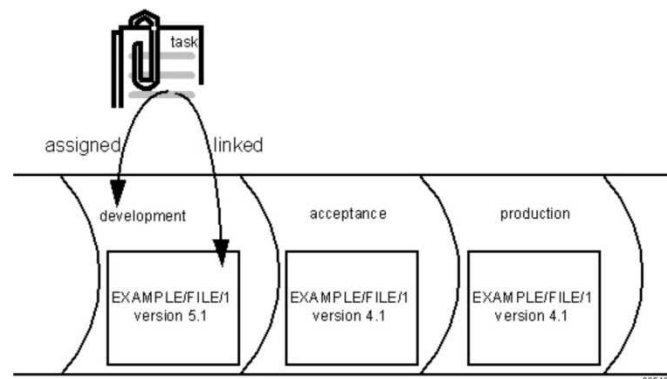


### RIS entity

A RIS entity is a “pre-defined” (by Infra Design) entity, used in the SURE development environment. These entities match logical development entities in RIS and SURE, such as FILE, PROBLEM, OPTION, DIM, FCM, SEL, and DCIS. For each of these RIS entities, the default development environment is defined. This means that by default changes for a RIS entity should normally occur in this defined environment; otherwise, the versioning system will identify the change as a quick fix.

### Task

A task is a defined unit of work in the repository. A task is added in SURE and has a number of attributes, such as project, environment, and handle-by. The environment attribute of the task indicates where the changes for this task will be made. This implies that a user does not directly select the target environment, but using the assigned task, the system will select the target environment. The “Handle-by” attribute identifies a usercode or an employee function that can handle this task, so if the system requires a task assignment, a list of available tasks will be presented depending on the employee function and the usercode. The system will link all adaptations that are made for a task automatically to that task.



**Impact Analysis**

Impact analysis is a procedure executed by the system that will detect overlapping tasks and changed definitions. If a source is modified, then SURE checks that it is not linked to another task (because of changes due to that other task). If this is the case, the user is informed and he must verify this indication. Another form of impact analysis is done at transfer time, where overlapping tasks are identified. Tasks are overlapping if the same definition was changed due to two or more different tasks. In this situation, the previously defined verification is given by the programmer.

**Grouping of Information**

Each relation contains a group identification indicating the "RIS entity" owning this relation. For example, the relations that belong to a file have FILE group. Each group has a corresponding control group. The control group of FILE is FILE-CONTROL. Relations with the control group are never transferred. They control the group in an environment, and may differ in each environment.

**Quick Fix**

A quick fix is identified as a change to definitions or sources in an environment other than the one specified as the default one for the RIS entity. If the same definition or source is changed in an environment with a lower hierarchy (closer to the default maintenance environment), and the definition is transferred to the environment where the quick fix was made, and then the system will guard the quick fix (to prevent that the quick fix is lost).

## 36.2.Overview and Advice

The ability in SURE that allows to choose environments, offers flexibility. However, it also requires decisions taken by the SURE user to organize the environments in such a way that they match the organization of the DP department. The "stability of an environment" and "control of the sum of the environments" are of main factors for this choice. We advise to start with a configuration, which matches the organization of the DP department. In other words, there is no merit in creating an ACCEPTANCE environment, if there is no acceptance procedure and test facility in the DP organization. It is possible to add or to delete environments after installation. The following example shows environment definitions for different situations.

**Examples**

A possible environment definition:

DATABASE	for database design and development of global functions
END-USER	for development of user applications
TEST	for test procedures
PRODUCTION	for production

Another possible environment definition:

MAINTENANCE	for program development
TEST	for test procedures
ACCEPTANCE	for acceptance procedures
PRODUCTION	for production

For program development based on release distribution of software products, such as SURE itself, the environment definition can be:

DEVELOPMENT	SURE program development
TEST	Test environment
NEXT-RELEASE	Next SURE release to be delivered
RIS80	SURE release 8.0 historical release
RIS70	SURE release 7.0 historical release
RIS60	SURE release 6.0 historical release

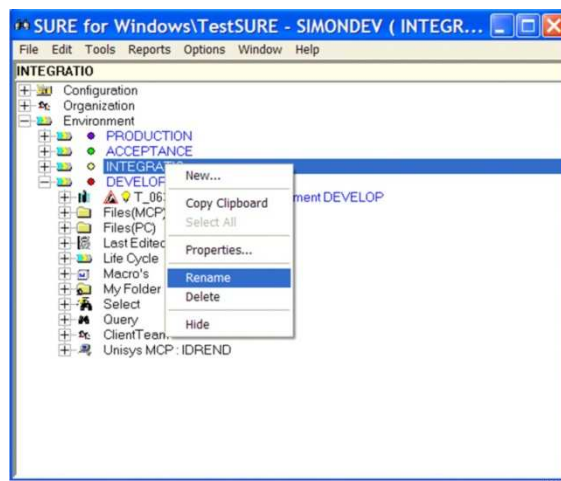
As shown in the examples, choosing the environments offers flexibility and the choice must be one matching the DP organization structure and the software development method.

### 36.2.1. Renaming an Environment

It is possible to rename an environment. This may happen during the initial configuration of SURE, when they might be discussions about the configuration of the environments.

To rename an environment:

1. Right-click the **Environment**.
2. Select **Rename**.



You must log-off and log-on to refresh your SURE browser. Other SURE users also have to log-off and log-on to refresh.

The standard SURE batch jobs are automatically regenerated for the new environment name. Notice that manual written SURE jobs must be checked after the rename. Also notice that jobs that are queued in a Unisys system queue need to be re-queued.

### 36.2.2. Environment

One can define new environments using <environment> folder. By default, the initial repository contains only one environment, called DEVELOP. Extra environments are created through the new function on the popup menu after a right-click an existing environment name. A maximum number of eight is presently pre-defined by Infra Design. This screen is the only physical limitation of the amount of environments that can be defined. The repository structure has no limitations.

The contents of the new environment will be copied from an existing environment. It is possible to copy an existing environment upwards or downwards. If the existing environment is copied upwards, then the hierarchical place of the new environment is directly above the existing environment; otherwise, the hierarchical place of the new environment is directly underneath the existing environment.

### WFL

A set of workflows is generated for each environment. The workflows have the following name structure: WFL/<environment>/<function>. In these workflows, the environment is passed as a parameter to the SURE programs, so that these programs are forced to run in a specific environment. If this environment parameter is not passed, the task assigned to the usercode will determine the target environment for the program. Under normal circumstances, this might be sufficient; however, under special usercodes, where daily procedures are executed; this mechanism guarantees that the batches are executed for the designated environment. A user, who by mistake assigns a task to this batch usercode, will not jeopardize the batch environment.

### **Tasks Arriving in This Environment are Indicated As SOLVED**

Multiple environments can be defined, indicating that a task is solved when it is transferred to this environment. Often this environment will be the one used as "PRODUCTION." However, if using historical releases, this indication may also be set for a historical release.

### **Task Indicated As Solved After Transfer in Two Phases**

If this option is set, and a task is transferred to this environment, then the status of the task will not be set to SOLVED and the linked files are not moved to the history of the task. Only when the task is transferred again to this environment the solved-status will be set and all further actions are done.

### **Delivery Enabled for This Environment**

This identifies that task have to be delivered from the environment. In a delivery environment an overlap analysis will be made at delivery time on solved tasks.

### **Environment is Read-Only (Secured)**

It is possible to secure an environment. No updates can be made on the environment if this option is set.

### **Use Extra Userid for All Usercodes**

This option is only relevant for the terminal emulation interface, when multiple users are working under the same CANDE usercode.

### **Use Extra Accesscode For All Usercodes**

This option is only relevant for the terminal emulation interface, when multiple users are working under the same CANDE usercode.

### **Default Work-File Location**

This directory identifies the CANDE work-environment for this SURE environment.

If the work file usercode is \* then all usercodes on the work file family are part of the CANDE work-environment.

Files that are checked out from SURE will be placed in the CANDE work-environment. If the usercode is \*, then the file will be placed under my-usercode in the work-environment.

**Phase 1, 2, 3 development**

This option makes it possible to split up the development process into (maximum) three different steps.

The first step can be "database design" in a SURE environment called DESIGN.

The second step can be "global functions development" in a SURE environment called GLOBAL.

The third step can be "application development" in a SURE environment called DEVELOP.

The following environments can be called TEST, ACCEPT, PRODUCTION, HISTORY.

**Creating A New Environment?**

A new environment is created in three steps:

1. Add the new environment using the online function. This zips the RESPECT/REPOSITORY batch program, which does the actual creating.
2. Wait until the RESPECT/REPOSITORY batch program is completed.
3. Update options that are environment specific, such as:
  - Object-usercode/pack (For each system)
  - Work-usercode/pack (For each system)
  - Bind object usercode pack (For each system)
  - Save compile options
  - Authorizations
  - Integrity options (For each system)
  - "put include file in work directory" option (For each system)
  - Debug pack
  - Data file pack
  - Coms-usercode
  - SURE compilation card files
  - SURE translate tables
  - File RISAPPLICATION/TITLES
  - File RESPECT/TITLES
  - "Solved" option
  - WFL options

### Example

Consider a repository with three environments: DEVELOP, ACCEPT and PRODUCTION. A new environment "TEST" needs to be created between DEVELOP and ACCEPT. This new environment must be defined using folder "Environment" (speed-menu "New"), and the order of the environments appears on the screen: PRODUCTION, ACCEPT, TEST and DEVELOP.

A specify on the TEST <create environment> button starts the RESPECT/REPOSITORY program for TEST. When this program is completed, the new TEST environment is available as an exact copy of ACCEPT environment.

### Delete An Environment

An environment can be deleted in two steps:

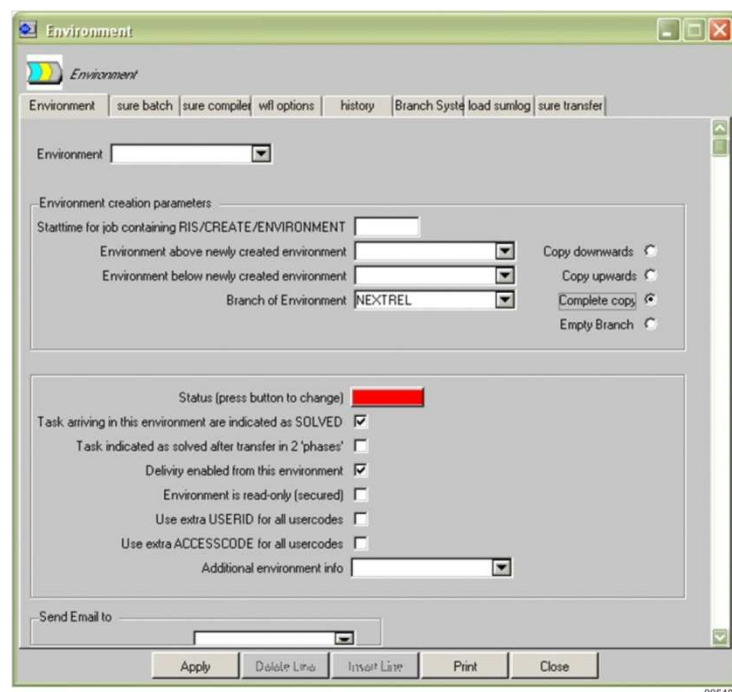
1. Delete an environment by performing the delete function using the popup menu.  
This zips the RESPECT/REPOSITORY program that does the actual deleting.
2. Wait until the RESPECT/REPOSITORY batch program is completed.

## 36.2.3. Branching of an Environment

It is possible to create a new environment as branch of an existing environment.

### Creating a Branch

1. Right-click the environment that has to be branched.
2. Click **New**.





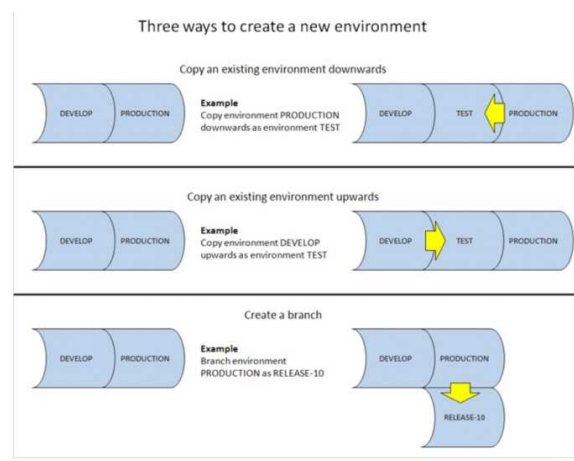
3. Enter the name of the new branch in "Environment" field.

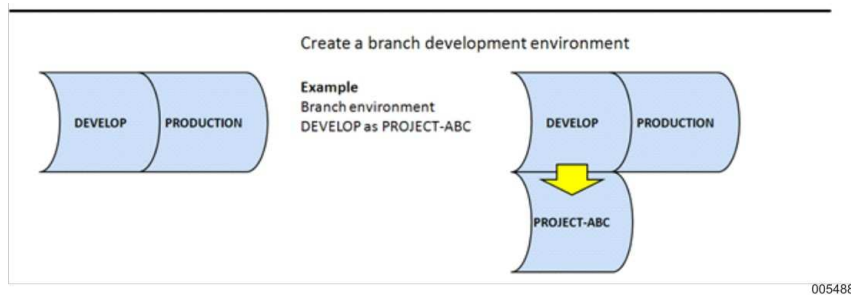
The screen (including the tab sheets) is prefilled with the attributes of the source environment. They must be changed to values that support the new branch. This is especially important for the work-location (usercode/pack) and the SURE-batch-location (usercode/pack).

The screen offers three methods to create a new environment:

- Copy an existing environment downwards.
  - Fields "Environment above new environment" + "Copy downwards."
  - In this case, the main development line is extended with a new environment.
  - This new environment becomes a stage in the full development life cycle.
- Copy an existing environment upwards.
  - Fields "Environment below new environment" + "Copy upwards."
  - In this case, the main development line is extended with a new environment. This new environment becomes a stage in the full development life cycle.
  - Methods "copy downwards" and "copy upwards" both create a new environment in the main life cycle, but with copy downwards, the new environment is filled with more stable file versions.
- Create a branch.
  - Field "Branch of Environment."
  - In this case, an existing environment is copied as a branch. The branch is not part of the main development line.
  - A branch is for example a release. The status of the application software in a specific environment at a certain point in time.
  - A branch can also be a separate development environment for a special project. In this case, the modifications in the branch have to be applied to the main development line when the project is ready.

The following picture shows the result of the three methods:





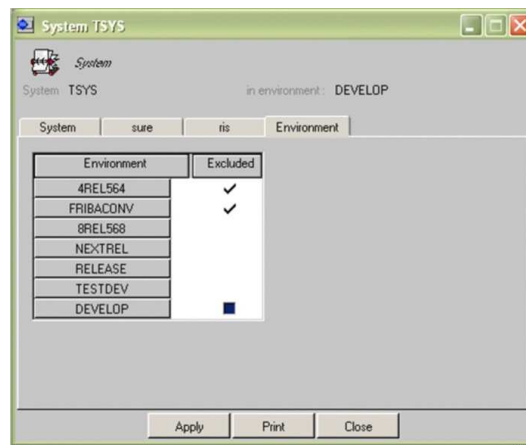
Branching an environment can be done in two ways:

- Create the branch as a complete copy of the source environment.  
In this case, individual systems can be excluded from the branch. This must be done at using the system properties.
- Create the branch as an empty environment.  
In this case, individual systems can be included from the branch. This must be done at using the system properties.

If an environment is excluded for a system, then the files that belong to that system are not stored in that environment.

To include or exclude environments or branches in a system using the System properties screen:

1. From Configuration menu, select **System**.
2. Click **<environment>**.
3. Click **<system>**.
4. Click **Properties**.
5. Click **Environment** tab.



The environment or branches that are marked are excluded.

After you included or excluded individual application systems, you must run batch function

```
RESPECT/REPOSITORY( "CHECK-SYSTEM-ENVIRONMENTS <system>" );
```

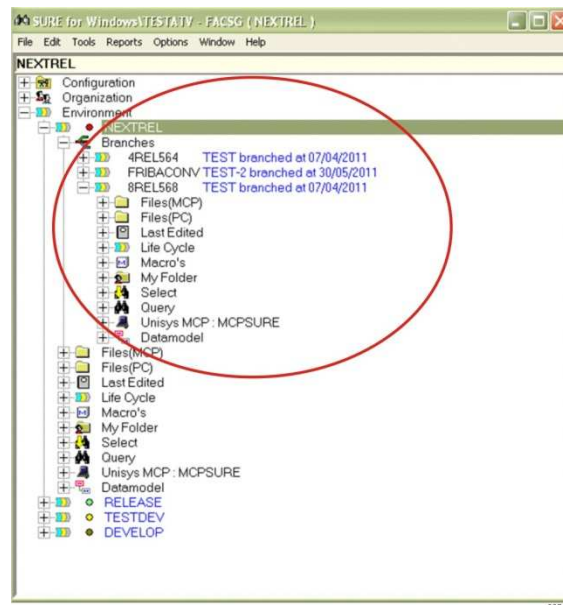
This environment exclusion for each system makes it possible to define separate branches for each system.

### Branches in the SURE Browser

Branches of an environment are visible in the SURE browser in a new folder of that environment, folder "Branches."

- If the environment has no branches then there is no "Branches" folder.
- Each branch of an environment is mentioned in the "Branches" folder and each of those branches contains the regular folders of an environment.
- The date when the branch was created is placed behind the name of the branch.

Environment NEXTREL has three branches: 4REL564, FRIBACONV, and 8REL568.

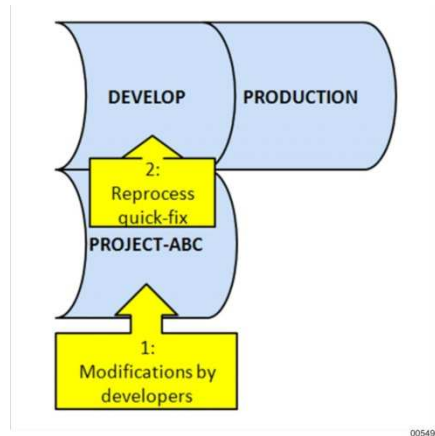


### Working with Branches

A branch is basically a regular environment, and almost all functions that work on a regular environment work also for a branch environment: Check-out, check-in, delta-files, tasks, the SURE batch, and so on. It is all supported in a branch environment.

The most typical aspect of a branch environment is that it is not part of the main development life cycle. If a task is transferred in the main line (from development to production, and all the environments in between), then the branch is not affected. The sources that are modified because of the task are not copied to the branches.

Changing a source in a branch also requires a task, but in this case the work-environment of that task is the branch-environment. It is not possible to transfer a task from a branch environment to the main life cycle, so source-modifications in a branch cannot accidentally overwrite sources in the main line.



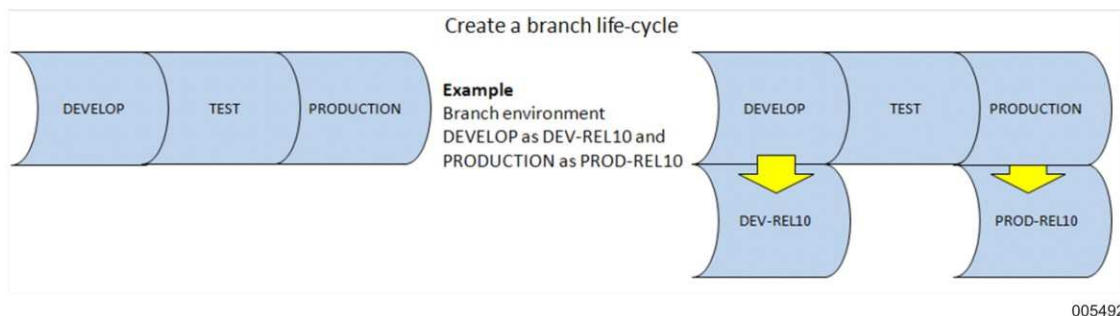
A source-modification in a branch is basically a patch on an earlier release of the system. Using the function "Reprocess Quick-fix" these patches can be applied to the main development line (similar as when a quick fix in production is reprocessed to the develop environment).

It is always possible to compare the source-content with a version in another environment, also if these are branches, and also if the source is checked out.

Source files and copy books of an excluded system do not appear in a branch. It may happen that a source of system SYSA references a copy-book that belongs to system SYSB, and that SYSB is excluded in the branch. If RESPECT/SURE/COMPILE has to compile the sources of SYSA in the branch then it use the missing copy-books from the environment from which the branch was created. In the example screen-shot on the previous page: RESPECT/SURE/COMPILE that runs for branch 8REL568 uses the missing copy-books from environment NEXTREL.

### Creating a Branch Life Cycle

It is also possible to define a life cycle of environments for a branch.



The rule is now:

- Tasks of the main life-cycle (DEVELOP → TEST → PRODUCTION) are not transferred to the branches
- Tasks of the branch life-cycle (DEV-REL10 → PROD-REL10) are not transferred to the main environments.
- After a branch-task is transferred to PROD-REL10 it can be reprocessed to the main life cycle with function reprocess quick fix.

### 36.2.4. Work-Environment

The CANDE work-environment can be defined for each application system. It is also possible to define a default CANDE work-environment. Each SURE environment must be linked with a CANDE work-environment.

The default CANDE work-environment must be defined using the Environment folder. To define:

1. Click **Environment** folder.
2. Click **<environment name>**.
3. Click **Properties**.

A work-environment for a system must be defined using the System folder.

1. Click **System** folder.
2. Click **<system name>**.
3. Click **Properties**.

The options that directly interact with the versioning mechanism are the WORK-USERCODE and WORK-PACK and the corresponding option <put the copy files in the work directory after save and load>. This option maintains the versions of the copy files; the rule files; and the RIS include files in this working environment.

It is important that the correct options are defined for each system/environment combination. Note, that options like <name-standards are active> and <use resource versions> will dramatically affect the behavior of the system. Therefore, it is advisable to make a formal procedure within your environment to add new projects or systems.

The work-environment definition identifies the directory to which a user has to be logged on when he wants to work on a task (updating definition or sources in the repository).

### Example

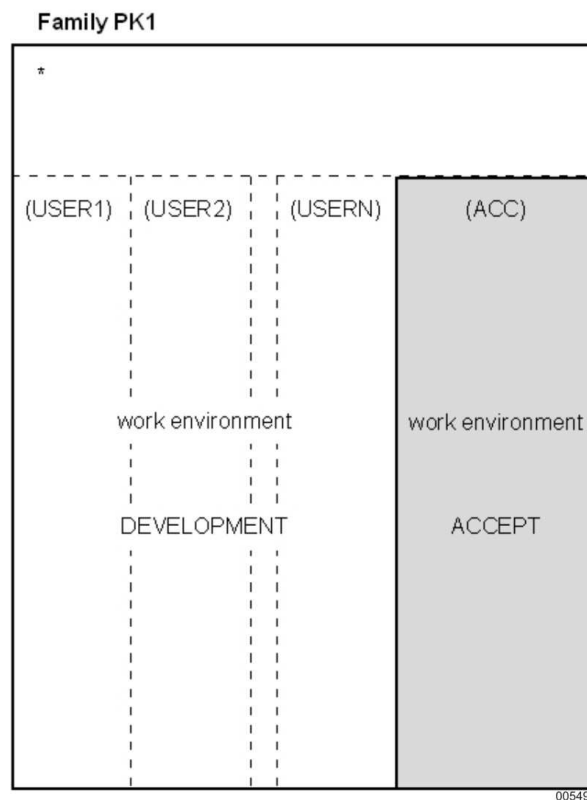
Consider DEMOSYS system with the following work-environments for each SURE environment:

SURE environment	Work environment
DEVELOPMENT	* on PK1
ACCEPT	(ACC) on PK1
PRODUCTION	(PROD) on PK2

The work-environment of DEVELOPMENT is \* on PK1 that means if a programmer is logged on with any usercode on family PK1, then the programmer is logged on in the work-environment of DEVELOPMENT, and is allowed to update sources or definitions in DEVELOPMENT.

The work-environment of ACCEPT is usercode (ACC) on family PK1. If a programmer is logged on with this usercode/family combination, then he is only allowed to update the repository on ACCEPT.

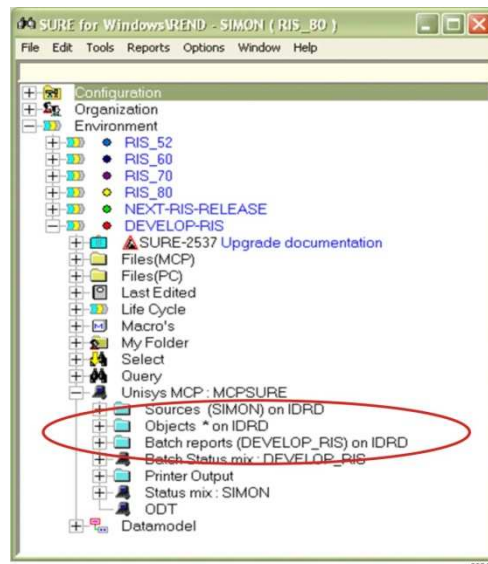
If a user is logged on with usercode ACC on PK1, then he is logged on in the work-environment of ACCEPT. A user is in the work-environment of DEVELOP if he is logged on with any usercode on PK1, except usercode (ACC).



The <put copy file in the work directory after the save on load> option enforces that copy files, rules and RIS-include files are copied from a SURE-environment to the work directory of that environment immediately after such a file is checked in, loaded, or transferred to that environment. The work directory is visible using CANDE for all programmers that are working for the corresponding SURE environment. This ensures that the programmers can normally do their job. (Writing, compiling, and testing programs)

To set this option using <SYSTEM properties>, click **Sure** tab.

The CANDE Work directory is visible in the SURE browser, under Unisys MCP folder.



### 36.3. Security

Security within the SURE development environment is based on the use of employee functions. Privileges are linked to employee functions, and a user inherits those privileges from its employee function. In specific circumstances, it might be decided to define privileges for an individual user. The rules and the behavior of the system are the same for both methods, but we advise to use the employee function method, because it appears to be far better manageable in most organizations.

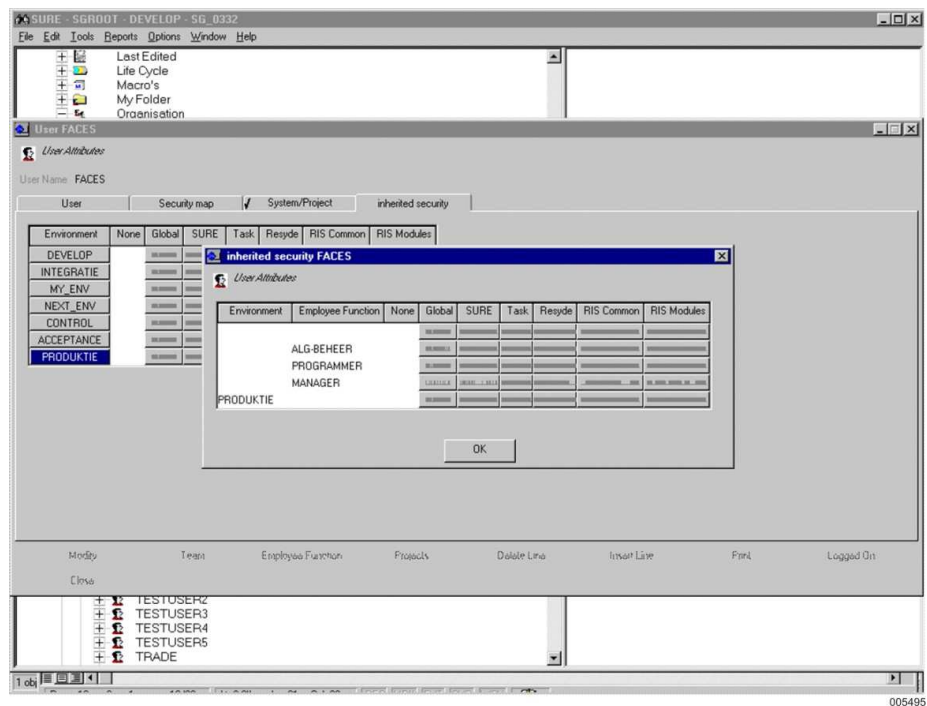
To define new employee functions using the Configuration folder:

1. Click **Employee** function.
2. Click **New**.

To define new users using the Configuration folder:

1. Click **User**.
2. Click **New**.

Both functions allow setting of the security maps for each user or for each employee function. The “inherited security” folder on the properties of a user shows the security map for the user for each environment. Clicking the environment will show how this security map was constructed (see following screen example).



Example

Consider the following two employee functions:

<b>Programmer</b>	Default authorization map:	RIS: SURE, RESYDE, Define task
		SURE: GET, SAVE, ASSIGN, LIST
	Production authorization map:	NONE
<b>Acceptant</b>	No default authorization map	
	Authorization for environment DEVELOP:	RIS: transfer-from
	Authorization for environment ACCEPT:	RIS: transfer-to

A usercode that has both employee functions is allowed to:

- Work as a programmer in DEVELOP and ACCEPT.
- Transfer tasks from DEVELOP to ACCEPT.

However, he is not allowed to do anything in PRODUCTION.



Environment	System	Project	Default	None	Global	SURE	Task	Resyde	RIS Common	RIS Modules
PRODUKTIE				✓						

005496

Environment	System	Project	Default	None	Global	SURE	Task	Resyde	RIS Common	RIS Modules
DEVELOP										
ACCEPTANCE										

005497

We advise to define authorizations for each employee function and to link these employee functions to usercodes. This is an easy manageable method to maintain the authorizations: by adapting the authorization map of one employee function, a whole group of usercodes can be authorized to do a specific function.

There are a number of variables within the security system, which have a direct relation with the versioning mechanism:

### User Default Environment (User)

The default environment defines the environment visible by default for a user, if the user is not assigned to a task. This variable allows users to view information in the repository in the default environment, but also in other environments by specifying the correlation letter in the environment summary. Note that changing information in the repository requires the assignment to a task and implicitly the assignment to the environment designated by this task.

### **Project (User)**

The project list assigns a user to one or more projects or systems. User's update capabilities are then limited to files and tasks that belong to one of those projects or systems. Therefore, a result of this definition is that a user can only see files and tasks that belong to a project in his project-list, and for each update, it is verified that the project of the file is part of the users' project-list. A user without a project-list does not have these restrictions, and he is allowed to make adaptations for all projects and systems.

### **Transfer From and Transfer To (Global Security)**

The combinations of these two authorization bits identify from which source environments to which destination environments the user is allowed to transfer tasks.

# Section 37

## Task

### 37.1. Task Function

A task is the main carrier in SURE to promote changes to environments. Because of its dominant function in SURE, a number of attributes are connected to a task that results in a number of major mechanisms.

#### **Distribution of Tasks over Various Programmers**

A task contains an employee function and a project. Using these two attributes, SURE builds a To Do list for a programmer. The programmer works on tasks from this list.

#### **Dependency of Actions**

Tasks can be made dependent on each other. This means that a task cannot be promoted before another task is promoted. For example, a program can only go to a next environment after database reorganization.

#### **Master Task**

SURE allows to group tasks together as parts of a master task. This implies that if the master task is promoted, it takes all dependent tasks together to the next environment.

#### **Impact Overlap**

Using the task mechanism, SURE will detect when sources or definitions are changed due to different tasks. This mechanism warns the programmer that after his change, the program contains changes not only for his task, but also for another task. Therefore, promoting a task can involve a risk in relation to the integrity of the application system.

#### **Quick Fix Mechanism**

SURE supports a quick fix mechanism that allows you to change sources in a production or acceptance environment. The task mechanism provides a feedback for these occurrences and prevents not to override these changes.

#### **Authorization Flexibility**

Often situations occur that a programmer needs additional authorization to perform a task. In his normal work, he does not need these additional authorizations (programmers in stand-by shift). The task mechanism allows you to temporarily assign an additional employee-function to a programmer while working on a task.

## **37.2.(Task) Organizations Not Using SURE**

Many Electronic Data Processing (EDP) departments use a task registration system for various planning purposes. Problems and new feature suggestions are registered and a project leader assigns these "units of work" to his staff. The programmer starts working on one of these tasks and after a while, when he has finished his adaptations, the new software is released for test and later on for production. The programmer knows the sources that are adapted for the same reason and thus have to go simultaneously to production, and he has to pass this information to the change control staff. Using this information, the change control staff releases the correct software at the right moment for the run-time environment.

The above described procedure is clear enough, but it is a manual procedure and therefore not waterproof. It often happens that an incorrect set of new software is released for the run-time environment. Sometimes new adaptations are released too early, but most of the times a new version of a program that should have gone live together with some other adapted programs, is forgotten to be released.

Some of the weak points are as follows:

- The programmer forgot that he adapted a particular source for this task.
- The communication between the programmer and change control failed, with the result that a particular source was not released.
- The programmer informed change control about the files that needs to be released simultaneously, but change control forgot to release a particular source.

SURE contains an integrated task registration system that gives a solution for the above-described problems.

Following functions are available in the task registration system:

- Register new tasks (error reports, new features suggestions).
- Approve or deny a task.
- Assign a task to one or more programmers.
- Transfer a task together with the adaptations that are made due to this task.
- Link "solution information" or other documentation to the task.
- Re-activate a task if necessary.
- Report the user about the solved tasks.

The task registration system is fully integrated with the daily working procedures of the EDP department. This means

- A source is automatically linked to the correct task when this source is adapted due to that task.
- Adaptations are released for a test or production level by releasing the linked task. If a task is transferred to a test or production environment, then all adaptations that were made for that task are transferred simultaneously.

- The task is automatically added to the historical information of a source when the task gets status solved. This history of a source is maintained by the system and is always up-to-date, replacing the need for additional comment in programs.
- The task registration system is not developed for planning purposes. It is possible to define the estimated work time per task, but there is no reporting mechanism to visualize this.

Since various commands of the task registration system can be authorized separately, it is possible to split up the responsibilities in the EDP department over several persons or employee functions.

#### **Example**

- A secretary who enters the newly reported tasks
- A project leader or analyst who approves or denies the tasks
- A project leader who assigns the tasks to the programmers
- A programmer who solves the task
- A change controller who releases the task

### **37.3. (Task) Tasks and Environments**

#### **Functional**

A task, or problem, is the main carrier for the versioning implementation in the SURE repository. Using the task as carrier for all technical changes and allowing transfer of tasks instead of technical definitions from one environment to the other, a system is offered whereby software management does not have to involve technical staff.

Tasks are added in SURE through the online function <define a task>, or through the task object in the SURE Explorer interface. The description of the task is entered together with a pre-defined PROJECT. Using this project, SURE gives an identifier to this task-description. A newly added task has a status of ENTERED. This status will change to <environment name> as soon as a programmer starts working on the task. When the task is solved, the status will change to SOLVED. Various overviews of tasks including tasks in progress, tasks that are solved during a period, and tasks per project are available.

Adaptations to sources or SURE definitions can only be made if the programmer is assigned to a "current task." A source that is adapted by the programmer is automatically linked to the programmer's current task.

The moment the first source is linked to the task, the task's status changes to <current environment>.

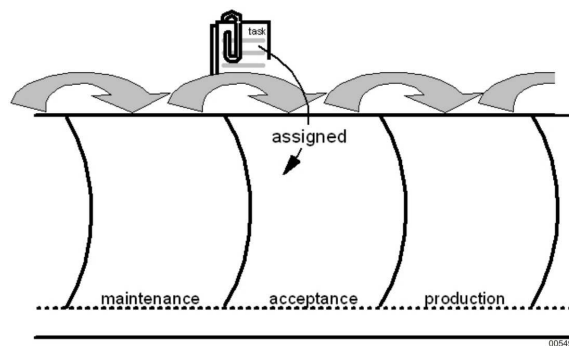
The moment the task is transferred to the next environment, the task's status changes to <next environment>. If the task is transferred to a "solved" environment, then the task is added to the historical information held against each of the linked programs or definitions. This method makes it easy to see why it was necessary to maintain a particular source or definition.

### Technical

A task definition is defined globally in the repository. For sources, it is possible that multiple versions can exist in the repository (each environment can contain a different version of a source), but for tasks this is not possible. A task is defined in a global level of the repository and therefore visible from all environments.

The reason for this is obvious. A task is the main carrier for the versioning implementation in the SURE software. A task controls an amount of adaptations that were made because of that task, and together with the task, these adaptations are transferred to other environments (test, accept, production). The task itself is environment independent, the task is equal for all environments, and the task must be visible from all environments, because the task is the bridge between the environments.

The following picture represents a repository with three different environments called MAINTENANCE, ACCEPTANCE, and PRODUCTION. Each one of these environments contains the definitions and sources valid for that environment. A task is connected to the definitions or sources changed due to this task. By transferring this task from one environment to the other, all of these changes will be transferred.



### Task Environment

The task environment identifies the environment where the programmer has to work on the task. All adaptations for a task will be made in the <task environment>.

## 37.4. (Task) Type

The task definition is supported since the first release of the SURE package. Ever since that time, we recognized that formal procedures within EDP organizations were not always appreciated. This resistance against formalized tasks often disappeared after a proper implementation of such a mechanism. This implementation requires a correct definition of the unit of work associated with a task. In practice, this results in some different types of tasks, dependent on the DP organization and the software development structure.

Examples of the task types include the following.

### New Feature Implementation

New features often involve a long implementation time and will often involve multiple developers and multiple sources.

### Reported Errors That May be Solved in the Next Release

Error reports normally involve a limited implementation time and will often involve one developer and a limited number of sources.

### Reported Errors That Need to be Solved Immediately in Production

This task involves direct change of the production software, with or without the acceptance through a specific environment. Changes are made in the development environment if no other changes are made in the same definition or source in the development environment. If the definition or source is already in maintenance due to another task (in the development environment), the task must be fixed using a quick fix mechanism.

You can also define task types for specific maintenance actions; for example, a database reorganization task.

Task types divide the tasks in global categories. Each category has a specific work-mode for the programmer: Working on a new feature is not the same as working on a high priority production error.

Since the task type identifies a work-mode for the programmer, a number of attributes for tasks with the same work-mode (task type) will also be the same. For example, tasks with "ERROR" task type always have a high priority. These default attributes can be defined for each task type.

To define these default attributes, perform the following steps:

1. Click the Configuration folder.
2. Click **Task type**.

The following default attributes can be defined for each task type.

### Priority

The priority identifies the task that needs to be solved first and the tasks that can be solved in a later stage.

To define a site priority value through dialog:

1. Click the Configuration folder.
2. Select **DropDdownBoxValue**.
3. Click **Priorities**.

Common values for priority are HIGH, LOW, MEDIUM, MUST, SHOULD, COULD, WONT, and FIRST. Notice that the priority can be used in user-defined macros, such as "Show all tasks with priority SHOULD, that are not yet assigned."

If a new task is entered and the "priority" field is not filled in, then the task type "priority" is used as a default value.

### Effort

This is an indication in how much time a task of this type should be solved. This attribute is for documentation purposes only. If a new task is entered and the "effort" field is not filled in, then the task type "approx. time" will be used as a default value.

### Environment to work on tasks of this type

The task type environment determines the environment where tasks of this type have to be solved. If a new task is entered and the "task environment" field is not filled in, then the task type environment is used as a default value.

### Security function

An employee function offers extra dynamic authorization to the user, if the user is working on a task of this type. See "(Task) Authorizations" for more information.

For example, if a new task is entered and the "Security function" field is blank, then the task type's security func is used as a default value.

### Environment where quick fixes must be made

This identifies the environment where the task can be quick fixed. If this field is not entered, then the task cannot be used for quick fix purposes.

### Tasks of this type must be created as a dependent task of task-type

With this option, you can separate the tasks with a functional task description from the tasks with a technical description. For example, a "technical" task must be dependent on a "functional" task.



### Tasks of this type can only be used to modify files of the system

The new task type option identifies a system-name. If entered, it is only possible to modify files of that system when you are working on a task with this task type.

Notice that with this option, you can enforce a new patchfile that is linked to the same project and system as the Project/System of the task.

### Name standard for tasks of this type

With this option, you can define a name standard formula for each task type. The name of a new task is calculated through a formula.

Refer to Section 39, "Name Standards," for information on name standard formulas.

When a new task is created, the name standard routine is called. If a new task is created, the project and task type must be entered to the screen. If a name standard formula is defined for the task type, then the "project name" and the "taskname" field values are passed as parameters to the name standard routine. The new task name is then constructed according to the name standard formula and the parameters.

Some examples of name standard formulas.

Project	Input Field	Formula	Result
	TaskName		
DEMO	T-001	<PROJECT>{<PROJECT,4}	DEMO0001
DEMO	T-001	<PROJECT>'-'{<PROJECT,4}	DEMO-0001
DEMO	T-001	<PROJECT>'-'{<PROJECT,2}	DEMO-02
DEMO	T-001	<PARAMETER>	T-001
DEMO	T-001	<PROJECT>'_'<PARAMETER>	DEMO_T-001

### Use this task type as error in reporting statistics

Tasks with a task type with this option are cumulated in the statistics at the end of a task overview.

### Tasks are place into the final deliver queue

Tasks with a task type with this option are placed into the final deliver queue at the moment that they get status solved.

### Solved tasks cannot be reactivated

Tasks with a task type with this option cannot be reactivated once they got status solved.

### Example: Task Type

#### Transfer Task: Transfer Task Role

- All users with employee-func TRANSFER-DEVELOP receives an email, if a task with this task-type is ready to be transferred from DEVELOP.
- All users with employee-func TRANSFER-ACCEPT receive an email, if a task with this task-type is ready to be transferred from ACCEPTANCE.

Task type ERROR

Task type

Type ERROR

Task type TransferTask Approval Assignment

Environment	Transfer task Role
DEVELOP	TRANSFER-DEVELOP
ACCEPTANCE	TRANSFER-ACCEPT

Apply Insert Line Delete Line Print

#### Transfer By

Limit the transfer of a task with a specific task-type to an employee-function.

On this tab, one can define the roles (employee-function) that are allowed to transfer a task with this task-type FROM an environment. If nothing is entered, then there is no limitation on role. If a role is entered without a corresponding environment, then the task-transfer is limited to that role for all environments.

- Tester users are allowed to transfer tasks with type "DailyChanges" from environment INTEGRATION to the next environment.
- Release Coordinator users are allowed to transfer tasks with type "DailyChanges" from environment ACCEPTANCE to the next environment.
- Team Leaders are allowed to transfer "DailyChanges" tasks from any environment.
- No other users are allowed to transfer "DailyChanges" tasks.

Task type DailyChanges

Task type

Type DailyChanges

Environment	Task can only by transferred (from this environment) by Role
INTEGRATION	Tester
ACCEPTANCE	ReleaseCoordinator
	TeamLeader

Apply Insert Line Delete Line Print

005501

### Approval: Approve Task Role

This task must be approved on DEVELOP by any senior developer, and on environment ACCEPTANCE by any project-leader and by any task approver.

All users with one of these employee functions receive an email when the task is ready to be approved.

Task type ERROR

Task type

Type ERROR

Environment	Approve task Role
DEVELOP	SENIOR-DEVELOPER
ACCEPTANCE	PROJECT-LEADER
ACCEPTANCE	TASK-APPROVER

Apply Insert Line Delete Line Print

005502

### Assignment: Assign Task Role

All TEAMLEADERS get an email when a task with this task type must be assigned on DEVELOP, ACCEPTANCE or PRODUCTION.

Task type ERROR

Task type

Type ERROR

Task type TransferTask Approval Assignment

Environment Assign task Role

DEVELOP	TEAM-LEADER
ACCEPTANCE	TEAM-LEADER
PRODUCTION	TEAM-LEADER

Apply Insert Line Delete Line Print

005503

### Add: Add Task role

If the user has role PROJECT-LEADER then he can add a task for any environment

If the user has role RELEASE-MANAGER then he can add a task for environment NEXT-RIS-RELEASE.

Task type FEATURE

Task type

Type FEATURE

Task type TransferTask Approval Assignment Add Miscellaneous

Environment Add task role

NEXT-RIS-RELEASE	PROJECT-LEADER
	RELEASE-MANAGER

Apply Insert Line Delete Line Print Close

005504

**Miscellaneous: AutoReady**

Given a task of this type automatically status READY when that task is transferred to a specific environment.

In this example, the option is added for task-type FEATURE on environment PRODUKTIE.

When a new task is added, it must be linked to an existing task type.

The task type attributes "effort" (approx. time), "priority", "environment," and "security func" are linked to the newly added task if these attributes are not manually defined for the task itself.

Notice that it is possible to define deviating values of these attributes when a task is entered or updated.

## 37.5. (Task) Definition

A task is defined on the task definition screen. This screen can be accessed through several ways.

**Terminal Emulation Interface:**

- Through a specify on RIS/<define task>.
- Through command TASK in SURE.
- Through the <define task> button on the "verify my task" screen.

Refer to "37.8.5 (Task) Verification" for more information.

**Sure Explorer Interface:**

1. Click the **Toolbar** function.
2. Click **Edit**.
3. Click **New**.
4. Click **Task**.

Defining a task is a global function. Therefore, the functionality on the task definition screen is not dependent on the way the task definition screen was entered.

### 37.5.1. (Task) Definition

The task definition is done through the new task dialog that can be opened as follows:

1. Click **Edit**.
2. Click **New**.
3. Click **Task**.

## Task

The default entries in this dialog of the task definition window include the variables most used for a task definition.

The new task dialog has the following layout.

The 'Task: New' dialog box is shown. It has a title bar 'Task: New'. The main area is divided into sections. The 'Task Definition' section includes fields for Group, Project, Type, Reported by (set to 'ISGROOT'), and Reference. There are checkboxes for 'Inform status change via eMail' and 'Inform assigned to handle by via eMail'. The 'Description' section has a large text area for Description and a smaller one for Short description. The 'Attachment' section has a text field and a button. The 'To Handle By' section has radio buttons for 'Not Assigned', 'I start working for this task (current)', 'User Code', 'Employee Function', and 'Team'. There are 'OK' and 'Cancel' buttons at the bottom.

Organizations uses SURE in a different way, and therefore, some fields may be omitted in the default task definition window. Other fields on this form may be given a default value. These settings are controlled using “Customize task” in the popup menu under the Environment folder.

1. Click the Environment folder.
2. Click **Customize Task**.
3. Click the **Task Fixed Fields** tab.

The 'Task Fixed Fields' dialog box is shown. It has a title bar 'Task Fixed Fields'. The main area is a table with columns: Field Name, Hidden, Required, and Default Value. The table lists various task fields. The 'Hidden' column has checkboxes, and the 'Required' column has checkboxes. The 'Default Value' column has text entries like 'GVB'. There are 'Apply', 'Default Value', 'Reset Line', 'Print', and 'Close' buttons at the bottom.

Field Name	Hidden	Required	Default Value
TaskName	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	GVB
TaskGroup	<input type="checkbox"/>	<input type="checkbox"/>	
ProjectName	<input type="checkbox"/>	<input type="checkbox"/>	
TaskType	<input type="checkbox"/>	<input type="checkbox"/>	
TaskReportedBy	<input type="checkbox"/>	<input type="checkbox"/>	
StatusByEMail	<input type="checkbox"/>	<input type="checkbox"/>	
TaskReference	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
TaskShort	<input type="checkbox"/>	<input type="checkbox"/>	
TaskAtt	<input type="checkbox"/>	<input type="checkbox"/>	
_req_ResolutionBy	<input type="checkbox"/>	<input type="checkbox"/>	
Name	<input type="checkbox"/>	<input type="checkbox"/>	
MetaName	<input type="checkbox"/>	<input type="checkbox"/>	
IsUser	<input type="checkbox"/>	<input type="checkbox"/>	
ByUser	<input type="checkbox"/>	<input type="checkbox"/>	
IsEmpFunc	<input type="checkbox"/>	<input type="checkbox"/>	
ByEmpFunc	<input type="checkbox"/>	<input type="checkbox"/>	
IsTeam	<input type="checkbox"/>	<input type="checkbox"/>	
ByTeam	<input type="checkbox"/>	<input type="checkbox"/>	
Space3	<input type="checkbox"/>	<input type="checkbox"/>	
InformByEMail	<input type="checkbox"/>	<input type="checkbox"/>	

This window allows you to modify standard SURE task fields, hide fields, and set required and default values.

Field Name	Field Type	Size	Required	Description
CONTACTPERSON	string	18		Contact person
TELEPHONE	string	18		Telephone
CUSTRELEASE	drop down value	6		Release
CUSTENV	drop down value	18		Environment
INFOATT	number	2		Number of attachments
DEBUGFILE	boolean	1		Debug File
EFFORT	number	3		Planning # Days
DELIVERY	date	8		Scheduled Date

This window allows definitions of site-specific task fields. These fields are added to the standard task entry and inquiry forms and can be used in reports, queries, and macros.

### 37.5.2. (Task) Attributes

#### Task Number

A unique task identification. By default, the system determines these unique task numbers, but authorized users define their own task-id.

### Task Type

A global identification for each task. Typical task types are ERROR, FEATURE, and QUICK-FIX. Each site can define its own task types.

To define task types for a site, perform the following steps:

1. Click the Configuration folder.
2. Click **Task Type**.

### Project

The project defines the relevant project for this task. If a user has a default project defined through its security attributes, he will only be able to assign a task where the project is equal to this default project in the security definition. Based on this project name and the defined name-standard, a task number is created that is the actual task name. The SURE software will connect all changes due to this task to this task number, and afterwards, this task number will be transferred to other environments.

### Task Group

The name of a task group. A task group is defined in the Task Group folder by clicking Configuration folder and then clicking Task Group. Normally, a task group includes a project, a type, and a short description. These fields are set for the newly entered task. The name of the given task will be set according to the name-standard set in the Global Options. It is not mandatory to use task groups. In fact, it is unhandy when you have many application systems loaded in the repository.

### Description

The description of the task. The first line (Text80\_1) is used as recognition text in the user interface.

### Short description

A short description.

### Reported by

The usercode or customer that reported the task. Various overviews are available to list the announced, solved, and active tasks per usercode or customer.

### Task environment

The environment where the developers works on the task. If this field is empty when a new task is added, then the environment is inherited from the task type.

### To handle by

This field identifies the usercode or employee function that has to solve this task. The task appears in the To Do list of all users with that employee function. If this field is blank then the task is not assigned to any user, and it will not appear in any user's task list.



The following assignments are possible:

Not assigned	Task assignment will be done later using Assign function.
I start working for this task	The task is assigned to the current user and made his current task.
<usercode>	The task is assigned to an individual user.
<employee function>	The task is assigned to an employee function.
<team>	The task is assigned to a team.

### Status

This field is maintained by the SURE software and represents the status and sub-status of the task. This field has the following values (the sub-status is mentioned behind the slash.)

ENTERED	After entering a new task
ENTERED/APPROVE	After entering a new task, when option <approve newly entered tasks> is set.
<environment>/ASSIGNED	When a newly entered task is assigned to a user in <environment>.
<environment>	When a task is active in <environment> or transferred to <environment>. Adaptations due to this task are available in the objects for this environment.
<environment>/TO-COMPILE	When a task is transferred to <environment> but the adaptations due to this task are not yet available in the objects for this environment.
<environment>/TO-ASSIGN	When a task is transferred to <environment> where it must be assigned (for example to a test-team).
QUICK-FIX	If a quick fix has been made due to this task.
SOLVED	After transferring a task to an environment that has the "solved" indication set.
SOLVED/DENIED	After a newly entered task is not approved but denied.

Other values for sub-status are:

<environment>/OK:	The task is transferred from this environment. See "Overview of Transferred Tasks."
<environment>/QUICK-FIX:	The task had status quick fix, and is now re-processed on <environment>.

### Inform assigned “to handle by” using Email

The assigned UserCode, EmployeeFunction, or Team receives an email that the task is assigned.

### Inform Status change using Email

The reporting UserCode or Customer will receive an email that the task is transferred to another environment.

Security func

Filling this field allows the temporary usage of the security map from this employee function when making changes for this task.



The standard security definitions in combination with the security function attribute for the task definition can resolve many security aspects in the DP organization. The example shows a configuration where a user with the employee function “module-programmer” can change things for the project “Insurance” at the environment maintenance. These securities are defined through the following options:

- Set default project for the user to “Insurance.”
- Define a security map for “module-programmer” in the environment “Maintenance.”
- Set the employee function for the user to “module-programmer.”

The ability to change things in the environment “acceptance” for the project “Insurance” comes with the task definition. For such a task definition, it is required to set the security attribute for the task to an employee function that is different from “module-programmer,” for example “quick-fixer.” Define a separate security map for the employee-function “quick-fixer” that allows you to change things in the “Acceptance” environment.

- Define a security map for “quick-fixer” in the environment “Acceptance.”
- Define a task with the security func to “quick-fixer.”

production					
Acceptance					
Test					
Maintenance					
	Customer	Account	Insurance	Base	etc.

 Module programmer
  Quick Fix Task

The diagram illustrates that by default, a “module-programmer” for the “Insurance” project can only change things in the environment “maintenance.” If this programmer works for a “quick-fix” task, he is allowed to change some definitions in acceptance.

### Attachment

The name of a Windows file that is an attachment for this task. This attachment is stored with the task and can be updated afterwards.

It is possible to link multiple attachments to one task through a ZIP file: Add multiple files in one zip-file and attach that zip-file to the task. A special function is available to change an existing attachment to a zipped attachment, as follows:

1. Right-click the **Task**.
2. Select **Attachment**.
3. Select **Zip Attachment**.

This function requires that WINZIP is declared in the AW\_OBJ.INI file.

```
[EXE]  
WINZIP = <winzip.executable>
```

### Priority

The priority identifies the tasks that needs to be solved first and the tasks that can be solved in a later stage. An organization can define its own priority values through the Configuration folder.

1. Click **DropDownBoxValue**.
2. Click **Priorities**.

The common values for priority are HIGH, LOW, MEDIUM, MUST, SHOULD, COULD, WONT, and FIRST. Notice that the priority can be used in user-defined macros, such as "Show all tasks with priority SHOULD, that are not yet assigned."

### Compile queue

If a task is linked to a user-defined compile queue, all files that are adapted because of this task will be compiled through that queue.

If a task is not linked to a user-defined compile queue, the files will be compiled using the normal queue.

### Department

The department of the user who reported the task. If this field is empty when a new task is added, the user's default departments will be inherited. The user's default department can be defined through the "User" dialog. When the task is reported by a customer, the department field should be left empty.

Various reports cumulate the number of open tasks per customer, or per department. Refer to Section 1, "Introduction," for more information.

### Effort (hrs)

Expected hours required to solve the task. If this field is empty when a new task is added, the default "approx. time" will be inherited from the task type.

### **Release**

The software release for which the task has to be solved.

This field is used for planning and documentation purposes. If you set the global options "use task-field 'next-release' to update the release number of its system" and entered a value in the "next-release" task-field, then the release number of the system changes.

### **Resource Definition**

The name of a resource definition table that defines the default location for retrieval of resources.

Refer to Section 15, "Copy Files," for more information.

### **Date Received**

The introduction date of the task. If this field is empty when a new task is added, today's date is used. This date is used for the overview of tasks that were reported during a defined period.

### **Performed by**

All users that worked on a task are assigned to the task. This field is used for documentation purposes only. Notice that multiple users can work on the same task, so multiple users can be enumerated in this field.

### **Date delivery**

The date by which the task has to be solved. This field is mandatory if the corresponding option is set. An overview of the tasks that are "overtime" is available. This overview lists all tasks that should have been solved but are still in progress on a specific date.

### **Date opened**

The date when the first adaptation was made for the task.

### **Date ready**

The date when the task got status solved. This date is used for the overview of tasks that were solved during a defined period.

### **Solved in release**

This gives the software release in which the task was solved. This release is inherited from the system to which the task is linked, at the moment that the task gets status SOLVED.

This release id is used on various overviews and on the delivery screen where it is possible to deliver all tasks of the same release.

## Reference

A task reference of the user or customer who reported the task. It often happens that a customer has his own task registration system where he keeps a log of reported tasks. Newly added tasks receive automatically a task number using a naming standard formula, and therefore, this task identifier will differ from the task number that is used by the organization task registration system. It is possible to search for a task using the reference field to obtain the correct task number.

## Master task

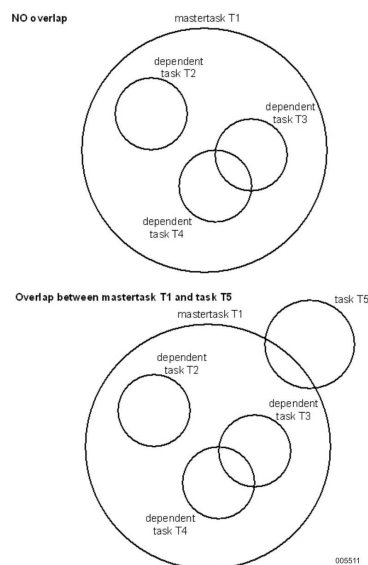
Define the current task as Master Task. A master task plus all its dependent tasks will be handled by the system as one logical work unit.

This means

- If the master task is transferred, all its dependent tasks will be transferred too.
- An impact analysis will be done for the master task and all its dependent tasks as one group.
- No overlap warnings are given between a master task and its dependent tasks.
- A master task can be dependent on another task.

If a master task is dependent on another task, then the master task cannot be transferred to the next environment before the other task is transferred.

If a master task is dependent on another master task, and this other master task is transferred, then the dependent master task is also transferred (with all its dependent tasks or master tasks).



Notice that the master task definition is stronger than the definition of task dependency. A task (T2) is made dependent on another task (T1) if this other task (T1) has to be promoted to a higher environment BEFORE the task itself (T2) can be promoted.

If the controlling task (T1) is defined as master task, then the two tasks will be promoted SIMULTANEOUSLY to the higher environment.

### **Power**

Define the current task as POWER task.

A power task overrides locks on files that are delivered for test.

Tasks can be delivered to another repository through the deliver function (SURE/<delivery>). It is possible to do a "test" delivery and a "final" delivery.

A test delivery can be used to check that the load of the task in the target repository is without errors, and that all files that are linked to the task are also correctly loaded. Generations and compilations can be started and the newly compiled objects can be tested to check that they work fine.

Tasks that are delivered for test are locked in the source repository (where the delivery was made). Files that are linked to the task cannot be adapted anymore, until the task lock is removed. The task lock is removed using a final delivery. If all tests in the target repository were correct, then the task can be delivered finally. The lock on the task during the test-phase ensures that the same versions of files are delivered during the final delivery as during the test-delivery.

When a task is locked, then none of its linked files can be adapted. An error is given if one tries to adapt a file that is connected to a locked task. It might happen that the locked file has to be adapted urgently because of run-time problems in the production environment. In that case, the task that reports the run-time error must have the power to override the locks on the files that have to be adapted. This can be achieved through a specify on the POWER button that defines the current task as power task.

### **Short**

A short description of the task. (This is only for documentation purposes.)

### **Task description**

A concise task description.

### **Task solution**

A description on how the task is, or can be, solved. This field can also be used to give an answer to the user or customer who reported the task.

### **Documentation impact**

A description of the possible impact of this task on the documentation. A batch function is available to list all the tasks that have documentation impact that can be used as a guide to update the documentation.

### **Reminder**

Various reminder info used as a yellow marker for informal communication between developers. This type of info does not appear on task overviews.

### Attachment

Windows file attached to the task. Any type of Windows file may be used as an attachment for a task.

### 37.5.3. Alternative Task Definition Screens

You can implement a site specific task screen that overrules the default task screen. We have several examples of programs that support site specific task screens. These programs are written in Visual Basic.

The default task screen can be customized by adding site-specific fields or by deleting standard fields, but on this screen, it is not possible to define cross checks between fields.

The main reason to implement a site-specific task screen is to enforce standard workflows. The generic task screen that is included with SURE is only limited customizable. Writing a custom task entry and update form may integrate better with the defined workflow and the specific customer requirements. For example, the workflow for task-type CHANGE: in the case that a CHANGE-task is entered, it must be linked to a predefined RELEASE-task.

See "Customizing the SURE Explorer" in Section 43 for examples of alternative Visual Basic task definition screens.

### 37.5.4. Hyperlinks to an URL or the SharePoint Page

You can place hyperlinks in the variable task information blocks: task description, solution, documentation impact, and reminder. This can, for example, be an URL of a SharePoint page, where more information about this task (for example a detailed analysis) can be found. This functionality is only available on alternative Visual Basic Task definition screens.

### Example

- The full URL name must be placed in the text.
- Double-click the URL name to open the web page.

## **37.6. (Task) Commands and Click Actions**

### **New/Task**

A new task is entered in the repository.

If field "task number" is left blank, then the new task number is automatically determined by the system, according to a pre-defined naming standard formula.

Refer to Section 39, "Name Standards," for detailed information on name standard formulas.

The name of the new task is created according to following rules:

1. If the TaskName field is not entered:
  - If an existing task name is entered in the "Task Group" field, then the master task name standard is used. The task that was entered in the group field is then automatically promoted to "master task" and the newly created task becomes a subtask of that master task. The name of the new task is <name of mastertask>/<4 digit number>. See "37.10 (Task) Master Tasks and Dependent Tasks" for details.
  - Otherwise, if a name standard formula is defined for the task type then that name standard is used.
  - Otherwise, if a task group is entered, then the name standard formula for newly entered tasks per task group (see Global Options screen) is used.
  - Otherwise, the name standard for newly entered tasks (see Global Options screen) is used.
  - If no task name is created (for example because of invalid name standard formulas) an error is returned.
2. If the TaskName field is entered:
  - If a name standard formula is defined for the task type then that name-standard is used. The content of the "TaskName" field is passed as a parameter to the name-standard routine, and can be used in the formula.
  - Otherwise, the name of the new task is set equal to the "TaskName" field value (only available for authorized users).
3. In all cases, the new task name must be unique.

The following fields are mandatory when entering a task: Task type, Project, Reported by, and Task definition. The "date delivery" field is mandatory if the corresponding option is set.

The following attributes are inherited from the task type when they are not entered: Priority, Effort, Security func, and Task environment.

If no department is entered then this attribute is inherited from the usercode who reported the task. If the "date received" field is left blank, then today's date is used.



The “to handle by” field is enabled only if the user is allowed to approve a task.

### **Assign**

This function assigns a task to one or more individual users, or to a team, or to an employee function.

### **Modify**

This function corrects a mistake or adds more information to the task.

The “to handle by” field can only be entered if the user is allowed to approve a task.

### **Properties**

Show all information attributes of a task or task group.

The following inquiries are possible:

- If the “task id” field is entered, that task is presented on the screen.
- If the “task id” field is empty and the “reference” field value is entered, a search is done for a task with that reference. If a task is found, then it will be presented on the screen.
- In the case that tasks are defined per task group (see options):

If the “task id” field is empty and “reference” field is empty, but “taskgroup” field is entered, then that task group will be presented on the screen.

- If “task id” field is empty and “reference” field is empty and (if available) “task group” field is empty, then the used current task will be inquired and presented on the screen.

### **Delete**

Delete a task from the repository. A warning is given if the task is linked to any file. This warning can be overridden using a second transmit.

A user is always allowed to delete a task if that task has status ENTERED and if it was added by him (in the case that a task was added by mistake.)

### **Close**

Set the status of a task to SOLVED. This command cannot be used if the task is linked to a file. Field “to handle by” is made empty, to ensure that the task will not appear anymore in any programmer's task list.

Most tasks that are reported will result in software adaptations. There are also tasks that do not result in a software adaptation. The reason for this are:

- The task is in fact a question. This task can then be solved by answering the question through the “task solution” field.
- The task reports an error, but the error is already solved in an earlier stage, or the error was reported unjustly (the software works as intended.)

### Deny

Set the status of the task to SOLVED/DENIED. This command cannot be used if the task is already linked to a file. The "to handle by" field is made empty to ensure that the task will not appear anymore on any programmer's task list.

### Current

Define the task as "my current task." If the "to handle by" field was not yet entered, and the user is allowed to approve a task, then the user ID of this person is entered automatically.

Notice that a task can only be made current if the task environment corresponds with the current environment of the user. It is possible to change from current environment on the task screen through a specify on the environment summary (upper-right corner). The newly chosen environment will then be valid for the total SURE session.

### Activate

Reactivate the task for another environment. A continuation screen is given where the valid environments on which the task can be re-activated are shown. A specify on one of these environments will change the task environment and the status of the task.

If the user is allowed to approve a task, then "to handle by" field is entered with this command too.

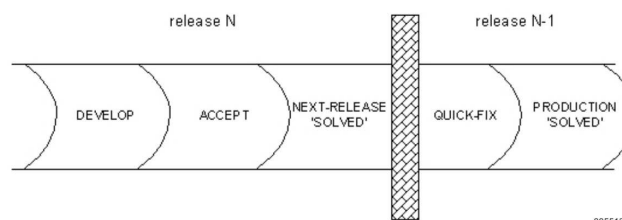
A task can be re-activated to one of the following environments:

- Each "first" environment of a release.
- All "earlier" (= closer to develop) environments that are in the same release of the current task status.
- The current task environment.

### Example

Consider a repository with the following environment:

DEVELOP	
ACCEPT	
NEXT-RELEASE	"SOLVED"
QUICK FIX	
PRODUCTION	"SOLVED"



005513

Notice that for the NEXT-RELEASE and PRODUCTION environments, the “solved” indication is set. The “solved” indication for NEXT-RELEASE enforces that the tasks that are transferred to that environment will get status SOLVED. Since solved tasks cannot be transferred, it is impossible to transfer from NEXT-RELEASE to QUICK-FIX.

In this example, the five defined environments have the following functions:

DEVELOP	=	The environment where the programmers do their normal development of new features.
ACCEPT	=	An acceptance environment where system tests are done.
NEXT-RELEASE	=	The future production environment.
QUICK-FIX	=	An environment where quick fixes can be made and tested. When these quick fixes are TESTED-OK, they can be transferred to production.
PRODUCTION	=	The current production environment.

In this case, the repository is divided into five environments, and in two releases. The first release exists of environments DEVELOP, ACCEPT, and NEXT-RELEASE. The second release exists of environments QUICK-FIX and PRODUCTION.

- The task can be re-activated to each “first” environment of a release: these are DEVELOP and QUICK-FIX.
- The task can be re-activated to all “earlier” environments that are in the same release as the current task status. If the current task status is NEXT-RELEASE, then the task can be re-activated to environments DEVELOP, ACCEPT, and NEXT-RELEASE.
- The task can be re-activated to its current task environment. If the current task environment is DEVELOP and the status of the task is ACCEPT, then it can be re-activated to DEVELOP.

Notice that the ACT function updates the status of the task, but it can also update the task environment.

### Ready

This function can be used to indicate that the task is ready to transfer to a next environment.

### Transfer

This function can be used to transfer the current task to a higher environment. The transfer function (screen) will be entered where the target environment can be specified. Refer to “37.8.10 (Task) Transfer” for more information.

### <toolbar>→Reports

Various overviews can be created of active, solved and announced tasks. Refer to “37.13 Task Overviews” for more information.

### Configuration→TaskGroup

A task group can be defined and updated with this command. This command is only valid if “use task groups” option is set. Refer to the “Use Task Group” option for more information.

### Drag a Task and Drop it on Other Task

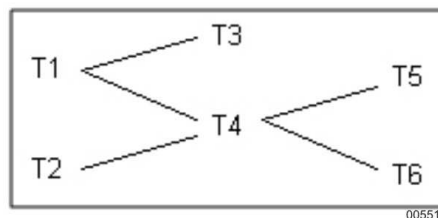
Make this task dependent on another task.

If a task (T2) is dependent on another task (T1) then this dependent task (T2) cannot be promoted to a higher environment BEFORE the controlling task (T1) is promoted to that environment. A task can be dependent on multiple other tasks, and a task can be dependent and controlling at the same time.

### Example

Consider task T3 that is dependent on task T1, task T4 that is dependent on tasks T1 and T2, and tasks T5 and T6 that are dependent on task T4.

The following picture represents this situation.



These tasks can be promoted to the next environment in one of the following orders:

- T1, T2, T3, T4, T5, T6
- T1, T3, T2, T4, T6, T5
- T2, T1, T4, T5, T6, T3

And so on.

An impossible order is:

T1, T4, T3, T2, T5, T6

(T4 is promoted before T2 and this is impossible.)

In the example, tasks T1, T2 and T4 are controlling tasks. Notice that a controlling task is not as 'strong' as a master task. When a master task is promoted, then all dependent tasks are promoted automatically too.

### Query

The select function is entered where it is possible to make a selection of tasks depending on the various attributes of the tasks. Attributes of tasks are stored in the repository through the following relations.

<b>Attribute of Task</b>	<b>Relation CLASS</b>
Task type	PROBLEM-TYPE
Project	PROJECT
Task group	PROBLEM-GROUP
Reported by	ANNOUNCED
Task environment	RIS-VERSION
To handle by	SETTLEMENT
Status	STATUS
Security func	SECURITY
Priority	PRIORITY
Department	DEPARTMENT
Effort	EFFORT
Release	NEXT-RELEASE
Solved in release	SOLVED-IN-RELEASE
Assigned to	EXECUTION
Date received	RECEIVED
Date delivered	DELIVERY
Date opened	STARTED
Date ready	READY

See “Select” in Section 31 for more information about selection expressions.

The task number and the first time of the task description of the selected tasks are given on a continuation screen.

### **Sub-folder LINKED**

All files that are linked to the task in the current environment are presented. If the task is transferred to a “solved” environment, then these files are added to the task's history. This history is presented by this function too.

### **Log**

The log of the task is presented. The following actions are logged:

- The time that the task was entered.
- Each status change (as a result of approval, assignment, transfer and re-activate).
- Each time the assigned user or employee function is changed. (Function ADD, UPD, ACT or CURR.)
- Each time the task environment is changed (function ADD, UPD and ACT.)

## **37.7.(Task) Authorization Mechanism**

The task registration system is developed as an integrated part of the SURE software. Programmers can only make adaptations to the application system if they are connected to a "current task." All software adaptations that the programmer makes are automatically linked to his current task. The software adaptations are promoted to a higher environment by transferring the task to that environment.

The task registration system is used to:

- Register new tasks (error reports and new feature suggestions.)
- Approve a deny task.
- Decide which tasks have to be solved in what order.
- Assign a task to one or more programmers.
- Link various kinds of information to the task.
- Re-activate a task if necessary.

Through the task authorization mechanism, you can split up the responsibilities in the EDP department over several persons or employee functions.

### **Example**

- A secretary who enters the newly reported task
- A project leader or analyst who approves or denies the task
- A project leader who assigns the tasks
- A programmer who solves the task
- A change controller who releases the task

The following task authorizations are available:

### **Authorization to add a new task with a self-chosen task number**

By default, the system determines the task numbers for newly entered tasks. If a user has this authorization, he is allowed to choose his own task-ID.

### **Authorization to add a task**

Allows you to add a new task and to modify all fields on the task screen, except "to handle by" and "task solution" fields.

The "to handle by" field can only be updated if the user is allowed to approve a task when option "approve tasks" is set.

The "task solution" field can only be updated if the user is allowed to update a task.

**Authorization to update some fields of a task**

This makes it possible to modify the task solution, technical info and 'documentation impact information.

The "to handle by" field can be entered if the user is allowed to approve a task if the "approve tasks" option is set, or if "approve tasks" option is not set.

**Authorization to delete a task.****Authorization to change the task status to solved immediately.**

Tasks that do not result in software adaptations can be changed to status SOLVED immediately through "Close" command. This authorization makes it possible to use this command.

**Authorization to Reactivate a Task.****Authorization to Make a Task Dependent on Other Tasks.****Authorization to Approve or Deny a Task.**

A task is approved by entering "to handle by" field. If this field is entered, then the task will appear in the To Do list of the programmers who have handled the task. A task can be denied using DENY command.

**Authorization to define and update task groups.****Authorization to define variable authorizations per task.**

This makes it possible to update "Security function" field.

**Authorization to define a task as power task**

Thus makes it possible to set the power-attribute of a task.

## **37.8.(Task) Flow of a Task in the EDP Department**

The process of a task is newly registered in the system and reported as solved; several other task stages can be recognized.

1. A task is newly registered in the system. The status of the task is ENTERED.
2. A task is approved or denied by a project leader.

If the task is denied (DENY function) then the status of the task will become SOLVED/DENIED.

If the task is approved by entering the "to handle by" field, then the status of the task will be changed to task environment. This enforces that the task appears in the To Do list of the indicated user, when that user is working in the task environment.

3. The programmer is ready with his adaptations for the task and the task is promoted to the next higher environment. This can be a test or acceptance environment. A task is promoted through the transfer function. The task will not appear anymore in the programmers To Do list.

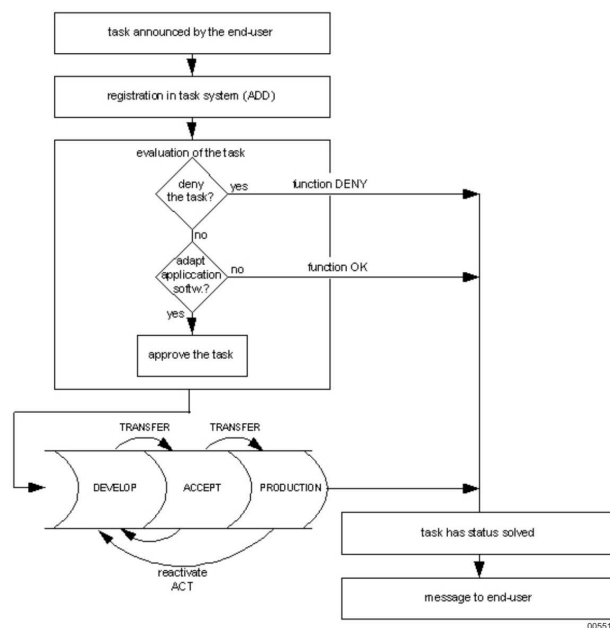
4. The test or acceptance team approves or denies the task solution.

If the adaptations are not correct, then the task has to be re-activated and the programmer can continue his work for that task.

If the adaptations for the task are accepted, then the task can be promoted to the next higher environment.

5. If the task is promoted to a “solved” environment (production), then the status of the task will change to SOLVED.

The following figure illustrates a task through a repository with three environments: DEVELOP, ACCEPT and PRODUCTION.



### 37.8.1. (Task) Validation

A task is approved if “to handle by” field is entered or updated.

If “approve tasks” option is set, then the user must have authorization “approve or deny a task”; otherwise, the user must have authorization “add a task” or “update a task.”

This can be done through the following functions:

- |          |   |  |
|----------|---|--|
| New      | : | A new task is entered and approved immediately.                        |
| Modify   | : | An existing task is now approved by entering the “to handle by” field. |
| Activate | : | A task is re-activated and routed to a user.                           |



Current : A task that is entered but not approved is defined as <my current task>. The "to handle by" field is automatically updated with <my-usercode>.

Assign : An existing task is approved by assigning it to a user.

A task can be denied through DENY function. Its status will then be SOLVED/DENIED.

A separate task authorization is available for the task approval and DENY function. Only users who have this authorization are allowed to approve or deny a task. Obviously this authorization is only meaningful if the above-mentioned option <approve newly entered tasks> is set.

### 37.8.2. (Task) Routing

A task is the main carrier for the versioning implementation in the SURE software. Each adapted source is linked to a task. By transferring the task to a higher environment (that is closer to production), the adapted programs are transferred.

All sources that a programmer adapts are automatically linked to the programmer's current task. If a programmer does not have a current task, then he cannot make any updates in the repository.

Before the programmer can start working, he has to select his current task from his task list. This current task is normally already defined in SURE by a task administrator.

Tasks are routed to a programmer's To Do list using the following three task-attributes:

- to handle by
- task environment
- project

The "to handle by" field can be entered or updated only if the user has the appropriate authorization.

A task will appear in the programmer task list if:

task to handle by	=	The programmer's usercode or one of the programmer's employee functions.
AND "task environment"	=	The current environment of the programmer.
AND "task project"	=	In the programmer project list, the check on the task project will not be done, If the programmer does not have a default project or system.)

### 37.8.3. (Task) Assignment

A task can be assigned by a user in various ways. In the following situations, a task is assigned to the user, or is made the current task:

- Function "Assign Task."
- Drag a task and drop it on a user.
- Function "re-activate task."
- Modify task attributes. Change field "to handle by."
- Make a task "my current task". This assigns it to myself.

A task can be assigned to one or more individual users through function Assign Task.

A task can be assigned to a team: all the team members will see the task in their To Do lists and can work on that task.

A task can be assigned to an employee-function. All users with that function will see the task in their To Do lists and can work on that task.

Any user that worked on a task remains assigned to it until that task gets status solved.

The "Undo Assign" function does an undo assign of the last assigned user, team or employee-function.

All user-task-assignments are removed when that task gets status solved.

#### 37.8.3.1. (Task) Assign a task to multiple individual users

If "Allow to assign a task to multiple individual users" option is set (To set the option, click **Global Options** and then click the **Task** tab), then it is possible to link a task to multiple individual users. Repeat "Assign Task" function for each required user to assign the task to each of those users. Each assigned user will see the task in his To Do list and can work on that task.

Any user that worked on a task remains assigned to it until that task gets status solved.

A task can be assigned through the following functions:

- New task : The option is not relevant, because the task was not yet assigned.
- Modify task : If the option is set then always assign to multiple individual users, else assign to single user.
- Activate task : If the option is set then always assign to multiple individual users, else assign to single user.
- Make current : Assign the task always to multiple individual users. If user U1 is working on a task, and user U2 defines that task as "my current task", then that task must stay in the To Do list of user U1.
- Assign task : The current setting of the option is pre-filled on the assign-screen, and it is possible to overrule the option for this single task.

Assign : BA-PROBLEM-0030

Assign task to

☒ Usercode  
☐ Employee function  
☐ Team

SGROOT Search

☒ Assign task to this user only (reassign)  
☐ Inform assigned user by eMail

OK Cancel

005516

### 37.8.4.(Task) Adapting the Application System

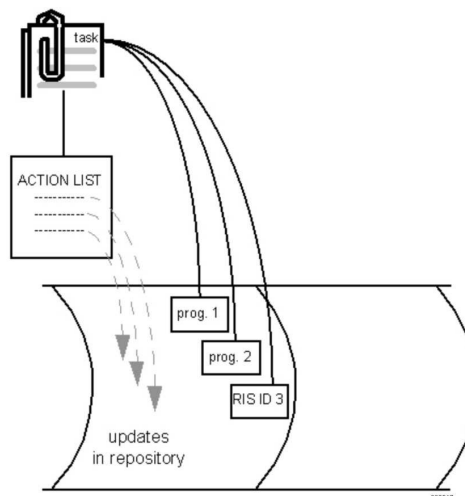
All adaptations to the application system will be linked to the current task. This means:

- When a source is adapted, this modified source is linked to the current task.
- When a RIS-ID is adapted, this RIS-identification is linked to the current task.

In some cases, the update action itself will be linked to the current task. This happens for the following actions:

- Add or update a message.
- SURE replace action.
- Generate an ALGOL include file.
- Define a verb, subject or abbreviation.
- Define an entity relation.
- Link or delink a selection function to a selection ID.
- Link or delink an FCT file to an FCM-ID.
- Link or delink a transaction to an LFI.
- Add or delete a username.

If the task is transferred to a higher environment, then the linked actions will be executed again on the destination environment.



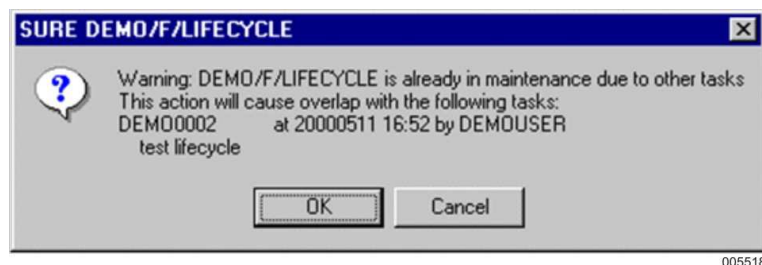
### 37.8.5. (Task) Verification

#### Indication of Overlapping Tasks

Together with the verification screen, a list of tasks that are already linked to the source can be given. This list is only given if other tasks than the current task are already linked and if the current task is not yet linked. If this list is given and an <OK> is given on the task verification screen, then an overlap of tasks is created.

Tasks that are overlapping are dependent on each other. When a task is transferred to the next environment, the check for overlap with other tasks will also be done, and a warning will be given if overlaps are encountered.

Notice that an overlap warning at transfer time does not mean that the overlapping tasks are also transferred. Only the source in the overlapping task that causes the overlap will be transferred, but this is obvious, since that source is also linked to the task that is transferred. Parts of the overlapping task are transferred to the next environment and this may cause integrity problems in the target's run-time environment.

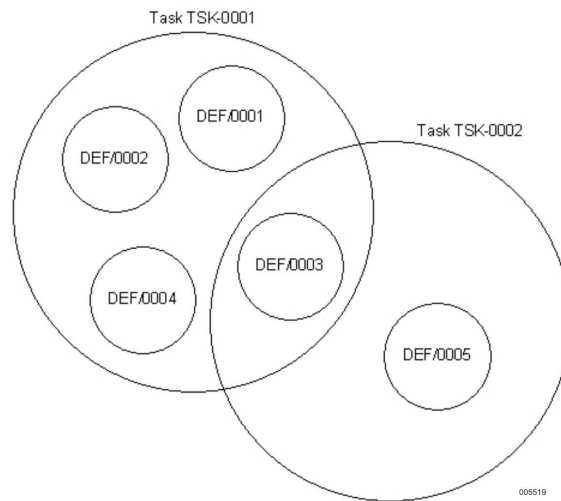


005518

#### Example

Task : TSK\_0001  
 Def/001  
 Def/002  
 Def/003  
 Def/004

Task : TSK\_0002  
 Def/003  
 Def/005



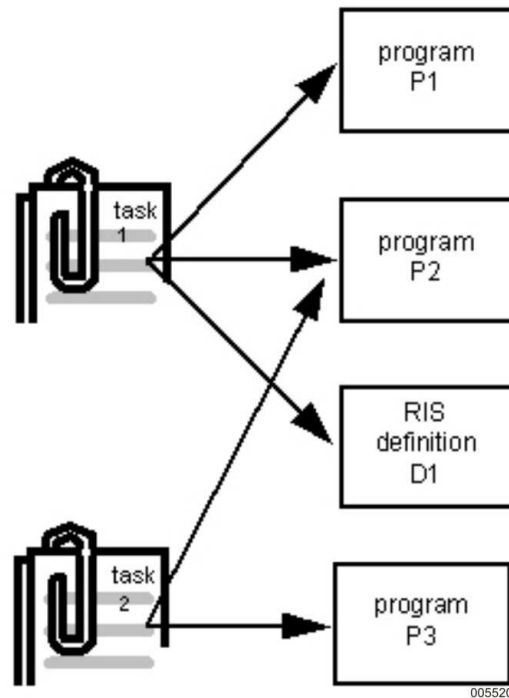
Changing Def/003 for task TSK\_0002 resulted in an overlap warning caused by Def/003 definition.

When task TSK\_0002 is transferred to the next environment, this overlap warning is given again. When the warning is approved, TSK\_0002 is actually transferred to the next environment, and all linked sources are also transferred. In this case, definition DEF/003 contains changes for task TSK\_0002 but also for task TSK\_0001, and this last task is NOT transferred to the next environment. However, this means that parts of the changes of TSK\_0001 are transferred while other parts are not transferred and this may cause inconsistency errors in the next environment.

Currently the SURE system moves the responsibility for these decisions to the person who is responsible for the development and the person who is responsible for the transfer. The quick fix solution offers adequate functionality to avoid these decisions.

### 37.8.6.(Task) Overlap Analysis

The overlap analysis is a procedure that initiates automatically. Alternatively, it may be initiated at user request. If a task is transferred from one environment to the other, the impact analysis is initiated as a part of the transfer procedure.



If task T1 or T2 is transferred, then the overlap analysis will give an overlap of these two tasks, due to reason "program P2."

The overlap analysis is manually initiated in the SURE Explorer interface in the transfer function by selecting an environment and thereafter pressing detail.

Warnings that are given when the task is transferred to a higher environment are also visible on the online impact overview. One of these warnings is the check on copy files:

- When a task is transferred to a higher environment, then the copy files of all sources that are linked to the task are checked. A warning is given for the copy files that do not exist in the target environment, because that will cause problems at compile-time.
- In the case of a "supplier" include-file (for example a Unisys-copybook, or a standard C++ include-file, or MicroFocus CBLTYPES.CPY) that is not stored in any environment in the repository, the warning is only given when the task is transferred from the lowest (DEVELOP) environment to the next level.

Example: Transfer Task Screen

Transfer task BB-PROBLEM-0002 from environment DEVELOP

Task

NameBB-PROBLEM-0002Reported BySIMONDEVMastertask

EnvironmentDEVELOPSolved BySIMONDEV

StatusDEVELOP

more patches

Target Environment	Blocked	Overlap	Fatal	Warning
DEVELOP		true	true	Can not transfer to the same environment
PRODUKTIE		true		

Errors in Linked...

Queue

Block

Detail

Compile Impact

Linked

Environment	Changed By	Entity	Item	Link	Error
DEVELOP	SGROOT	File	DATA/P/FACDB		
DEVELOP	SIMONDEV	File	DEMO/PROG/3		
DEVELOP	SIMONDEV	File	EA/DBP/EADB0015.CBL		
DEVELOP	SIMONDEV	File	EA/DBP/EADB0016.CBL		
DEVELOP	SIMONDEV	File	EA/DBP/EADB0017.CBL		
DEVELOP	SIMONDEV	File	EA/DBP/EADB0019.CBL		
DEVELOP	SIMONDEV	File	EA/DBP/EADB0045.CBL		

Link

Move

Task Detail

Undo Request

Undo Assign

Check In

Close

Select an environment and click "Detail" to see the impact overview of that environment.

Impact Overlap for Task : BB-PROBLEM-0002

Task

Overlap with Task	Overlap	
	warning	S/BA/WFL/BAWFL0001 is not yet compiled on DEVELOP
	warning	S/BA/DBP/OVERRIDE is not yet compiled on DEVELOP
	warning	DATA/P/FACDB is not yet compiled on DEVELOP
	warning	S/BA/DLP/BAALP0019 is not yet compiled on DEVELOP
	warning	S/BA/DLP/BAALP0300 is not yet compiled on DEVELOP
BA-P-00073		Release 52.0
	SEL	XREKENING/01
	SEL	XKLANT/01
	true	FILE S/BA/DBP/OVERRIDE
		FILE S/BA/DLP/BAALP1234-12345678
	true	FILE S/BA/DLP/BAALP0300
		FILE TEST/P1M-TRANSFER
		DATABASE SNUNX
		SEL-FUNC UXS001
		SEL-FUNC UXS002
SIEMT001		A
	true	FILE S/BA/DBP/OVERRIDE

Print

Close

The files where the actual overlap exists are indicated with "Overlap = true."

37–38

8207 4 121–000



### 37.8.7.(Task) Delink

You can delink a file from a task. There are several methods to delink a file:

- The delink button on the task transfer dialog.
- "Undo Request" function.

The only valid reason why a source should be delinked from a task is that a mistake has been made, and that it was not necessary to adapt the source after all.

Delinking a file through "Undo CheckOut" or "Undo Request" functions is only possible if the source is not changed since the last time that it was transferred to the next environment.

Notice that the DELINK function is dangerous. Files that are adapted are not linked anymore to a task and the adaptations might never go to production.

The overview of active tasks shows all changed files that are not linked to any task. This overview is made for each environment.

### 37.8.8.(Task) Ready

Function make-task-ready is used to indicate that the modifications for that task are ready, and the task is ready to be approved or transferred.

An email is sent to the task-approve-team or to the task-transfer-team (if the appropriate email function is enabled)

It is possible to define for each task-type and for each environment that a task of that task-type is automatically set to READY when the task is transferred to that environment.

### 37.8.9.(Task) Approve

SURE offers the task-approve function. A task cannot be transferred to the next environment before it is approved on the current environment.

You can define one or more task approvers for a task type. Each environment has a different set of task-approvers. If no task approvers are defined for the task-type on an environment then no approval is required for that environment.

The task-approver is notified by email when a task must be approved

A task cannot be transferred to the next environment before it is approved by each defined task-approver. A task-approver is an employee function. Any user with that employee function can approve the task.

**Example: Define One or More Task Approvers for a Task Type**

Environment	Approve task by
PRODUKTIE	ACCEPTANT
DEVELOP	TASK-APPROVE-2
DEVELOP	TASK-APPROVE-3

005523

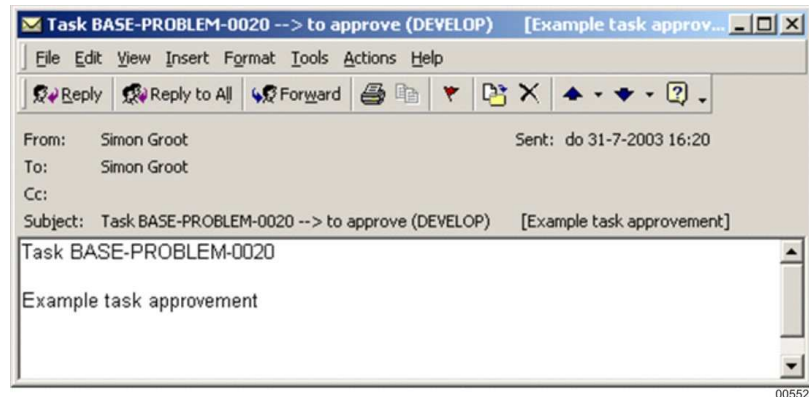
This example shows task-type FEATURE-17. Tasks with this task-type must be approved on environment DEVELOP by a user with employee-function TASK-APPROVE-2 and by a user with employee-function TASK-APPROVE-3 (or by a user with both employee-functions). Tasks with this task-type must be approved on environment PRODUKTIE by a user with employee-function ACCEPTANT.

The next example shows a task with task type FEATURE-17. The task cannot be transferred from DEVELOP to the next environment, because it must first be approved by roles TASK-APPROVE-2 and TASK-APPROVE-3.

Target Environment	Blocked	Overlap	Fatal	Warning
DEVELOP			true	Can not transfer to the same environment
PRODUKTIE			true	Task not yet approved on DEVELOP.

005524

- A task can only be approved when it has READY status.
- An email is sent to the task approvers when the task is READY.
- The email subject contains the task-name, the required action, and the first line of the task description.

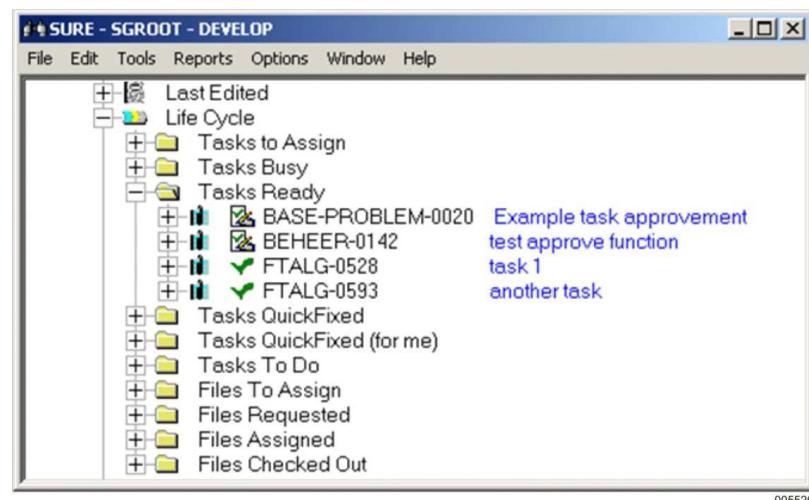


The email is sent to each user with the task-approve employee function. Any of those users can approve the task. Therefore, this task must be approved by a user with employee-function TASK-APPROVE-2 and by a user with employee function TASK-APPROVE-3 (or by a user with both employee-functions).

It is possible to select the task that I have to approve in SURE: The following macro or query shows all tasks that must be approved by myself:

```
TO-APPROVE (MY-EMPLOYEE-FUNCTION) AND SUB-STATUS(READY) AND PROJECT(MY-PROJECT)
```

Tasks with status TO-APPROVE are still visible in the READY list with another icon.



To approve the task, right-click the task name and select the **Approve** function. The icon is changed to the ready-status when no further approvals are required.

The task is ready to be transferred when all the required approvals are done.

### Considerations

- It is not necessary to approve a task, if no changes were made for that task.
- Changing the task type of a task recreates it's to approve queue according to the new task type.
- The to-approve queue is also recreated when the developer makes an extra source modification under that task while the task has status READY, or when the task is re-assigned while it has status READY.
- The email is only sent to a user if a valid email address is defined for that user.
- If the user who makes a task ready is also a task-approver, then that approve is done automatically.

### 37.8.10. (Task) Transfer

The Transfer option for a task shows a screen with the environments listed and all the linked entities to this task. It is possible to transfer the task if no errors are present. The transfer button is disabled until all errors are solved and a correct environment is selected.

**Transfer task SG\_0332 from environment DEVELOP**

Task

Name SG\_0332      Reported By SGR00T  
 Environment DEVELOP      Solved By SGR00T  
 Status DEVELOP

```
ISPLAY("E:ëëëëE");
ISPLAY("A:äääãA");
A :äääã
E :ëëëë
```

Target Environment	Blocked	Overlap	Fatal	Warning
DEVELOP	true	true		Can not transfer to the same environment
INTEGRATIE				
MY_ENV		true		This environment is secured (busy with creating or deleting)
NEXT_ENV		true		This environment is secured (busy with creating or deleting)
CONTROL		true		Errors on environment MY_ENV
ACCEPTANCE		true		Errors on environment MY_ENV
PRODUKTIE		true		Errors on environment MY_ENV

Errors in Linked...   Queue   Block   Detail   Compile Impact

Linked

Environment	Changed By	Entity	Item	Link	Error
DEVELOP	SGR00T	File	S:/S/DBP/ISDBP0010		File is still in use by user SGR00T
DEVELOP	SGR00T	File	S:/S/DBP/ISDBP0011		File requested and assinged, but not yet ...
DEVELOP	SGR00T	File	S:/S/DBP/ISDBP0012		File requested and assinged, but not yet ...
DEVELOP	SGR00T	File	S:/S/DBP/ISDBP0013		
DEVELOP	SGR00T	File	SIMON/P/20		File is still in use by user SGR00T

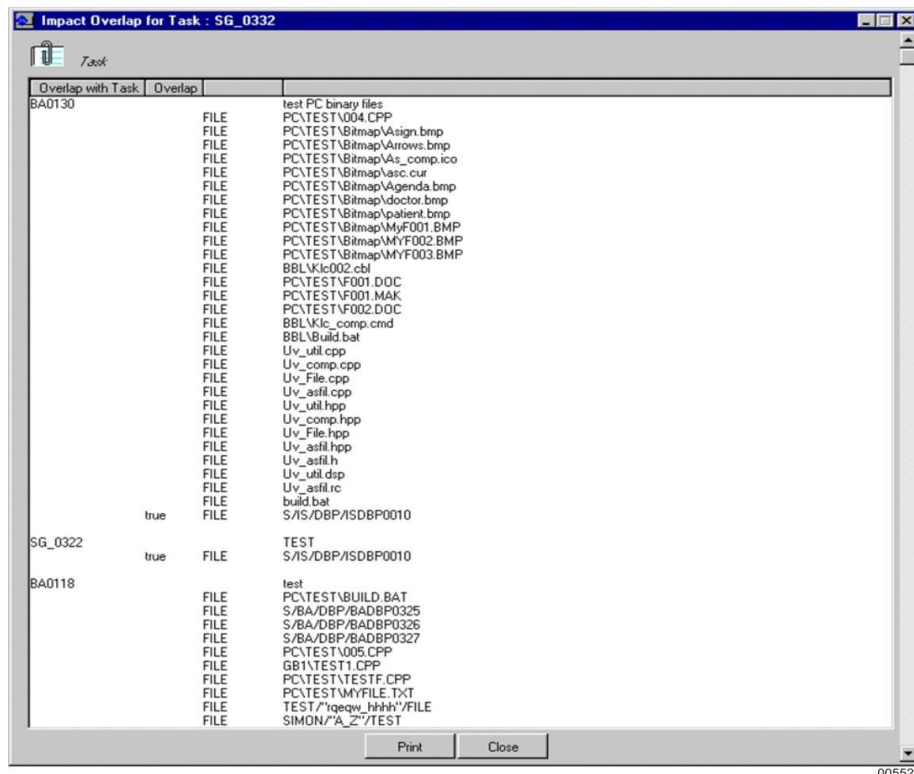
Link   Move

Close

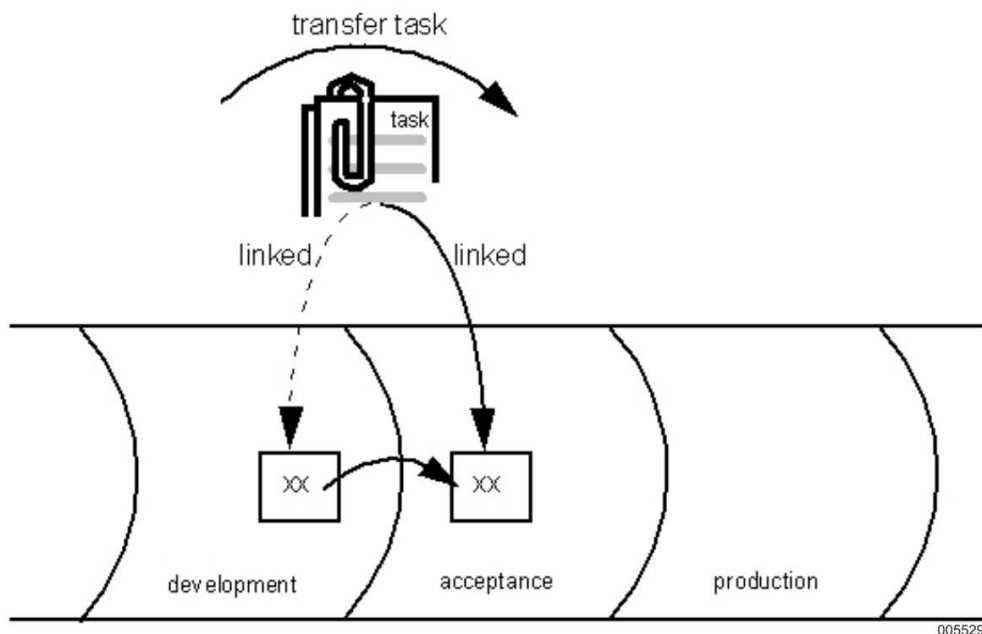
005527

Button	Action
Transfer/ Refresh/ Errors in Linked	Starts the task-transfer from one environment to the other. First an environment must be selected and the linked entries must not contain any error.
Queue	Queue a transfer for the evening batch run.
Block	Blocks a task until a certain date.
Detail	Shows the impact analysis for a task on an environment.
Compile Impact	Shows the compile impact of the task on a target environment after the task is transferred to that environment. So it answers the question: what will be compiled <b>extra</b> when I transfer this task?  The following files are returned in the list:  Files that must be compiled on the target-environment because of the task but which are not yet compiled.
Link/Delink	Link or Delink an object from a task.
Move	Move an object to another task.

The transfer will detect all irregularities in the definitions and sources targeted by the transfer. This means the sources that are still in maintenance or still assigned to a user will result in error indications. These errors are presented in an overview and they must be resolved before the transfer is actually executed.



The transfer copies the sources immediately to the target environment and to all environments between the originating environment and the target environment. As stated before, every environment contains a complete application system, containing all definitions and sources. The task-transfer only copies the new FileVersions to the destination environments. The actual compilations of sources are handled by the integrity mechanism. The transfer triggers the integrity mechanism because every transferred source gets a changed indication in the transferred environment with a timestamp equal to the current time of transfer. (RIS users only: Because of the integrity actions, all sources using a changed RIS-definition are placed in the generation queue. In the daily evening batch, the modules that are resident in the generation queue are regenerated and recompiled.) If a copy file is changed (or regenerated) then all programs that use this copy file are recompiled. Therefore, the transfer of sources is done during the transfer command, but the effect of these changes will be applied in the daily batch procedures.

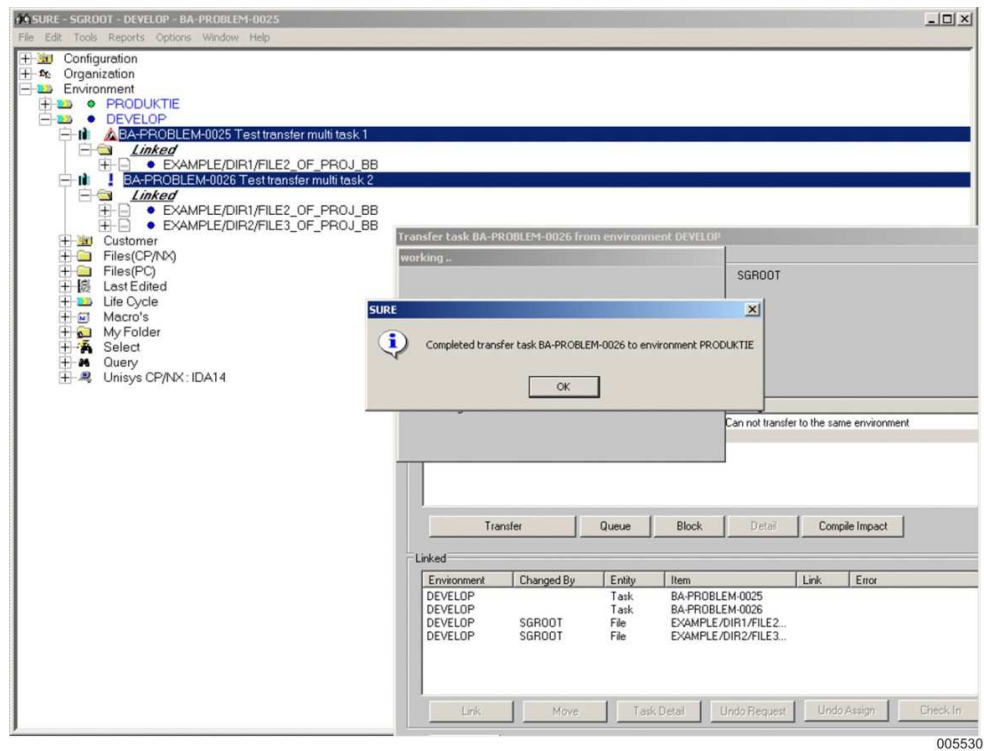


When a master task is transferred, all the tasks that are dependent on this master task are also transferred. In that case, the check for overlapping tasks will be done for the master task and all its depending tasks as one group.

### 37.8.10.1.(Task) Transfer Multiple Selected Tasks as One Unit

#### Example

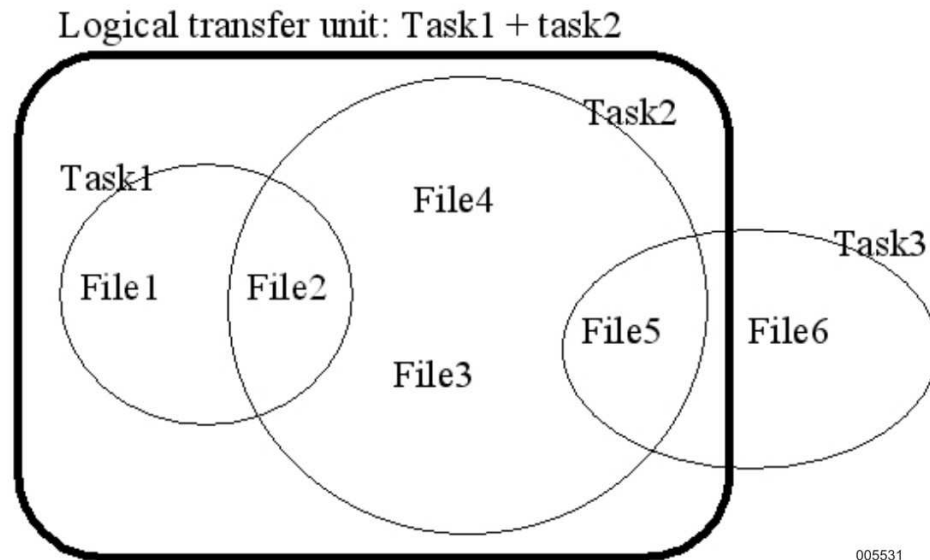
The following example shows that the two tasks are selected in the browser and transferred to the next environment.



All selected tasks are transferred as one logical unit. Overlaps between the selected tasks will not be reported. Overlaps with other tasks are still reported.

The logical transfer unit consists of two tasks: Task1 and Task2. These two tasks overlap with each other at File2, but that overlap is not reported.

The logical transfer unit has another overlap with Task3 at File5. This overlap is reported because Task3 is not part of the logical transfer unit.



The logical transfer unit exists only during the transfer action. After the transfer action, all tasks of the logical transfer unit are again treated as independent entities.

### 37.8.11. (Task) Reactivation

When a task is transferred, the status of the task is changed from <old-environment> to <new-environment> or SOLVED. It is only possible to make changes for a task if the task environment is equal to the task status. In that case, the task can be made current in the task environment. After a transfer, the status of the task is not equal anymore to the task environment, and therefore it is not possible to work on the tasks. If extra changes need to be made for the task, then the task has to be reactivated. The environment to which the task must be reactivated can be specified on an intermediate screen. The status of the task will set equal to the chosen task-environment.

If a task is resident in the quick fix queue (status quick fix or sub-status quick fix), then the task can only be reactivated using the "Reprocess QuickFix" function. The quick fix adaptations can then be re-processed on another environment. This means that command re-activate cannot be used for a task that is resident in the quick fix queue.

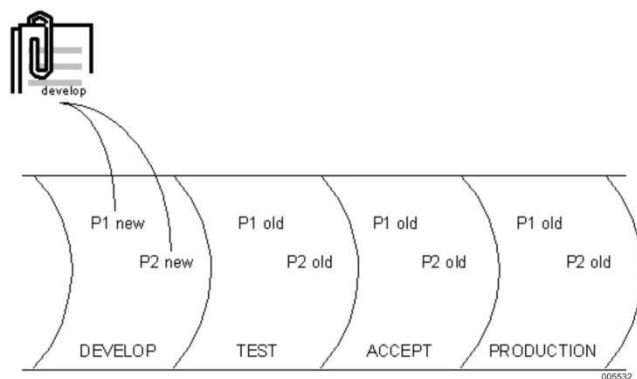
#### Example

Consider a repository with four environments: DEVELOP, TEST, ACCEPT, and PRODUCTION.

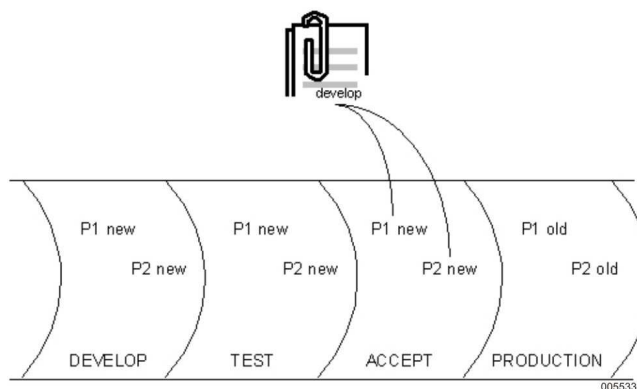


The following table shows the status and environment of a task after various commands.

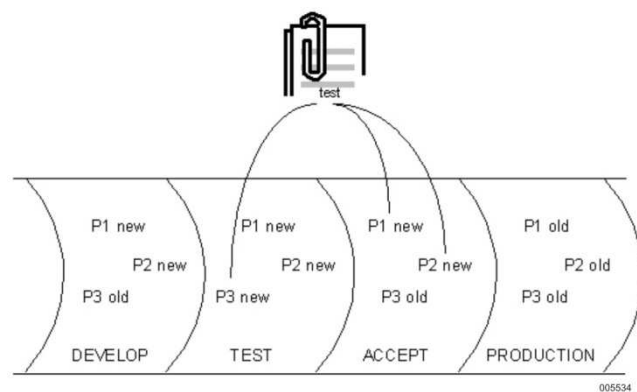
Command	Environment	Status
Add a task	DEVELOP	ENTERED
Approve the task: UPD "to handle by" Make adaptations in DEVELOP	DEVELOP	DEVELOP



Transfer the task to TEST and ACCEPT	DEVELOP	ACCEPT
--------------------------------------	---------	--------

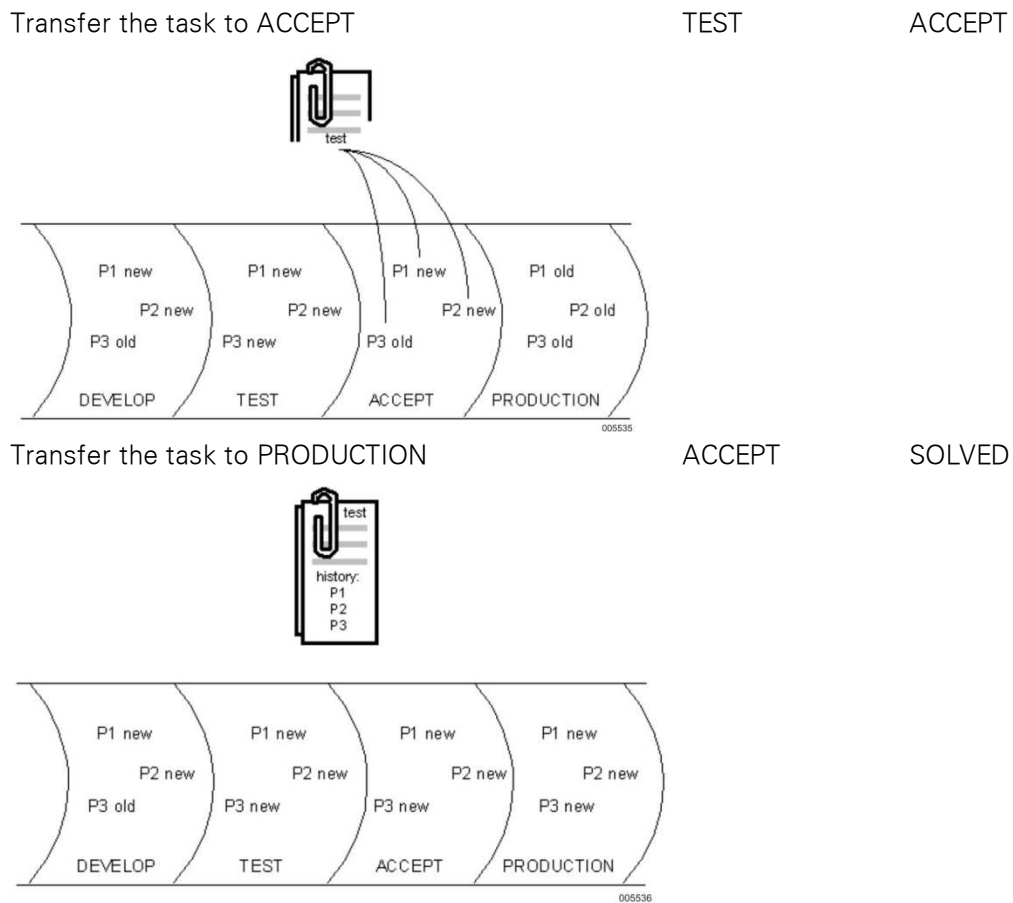


Reactivate the task to TEST and make adaptations	TEST	TEST
--	------	------



## Task

---



Notice the reactivation of the task. In this case, the task was reactivated to environment TEST. If adaptations to the application system are made in environment TEST, then these adaptations will not be available in the DEVELOP environment. For example, prog. P3.

### Example

A task must be reactivated through function. To activate task, perform the following steps:

1. Right-click the **Taskname**.
2. Select **Reactivate**.

The following screen is given.

Reactivate Task: FTALG-0423

Activate task in

Activate in:

☐ I start working for this task (current)

☐ Request connected files for user which modified the file

Assign task to

☒ Keep same assignment

☐ Usercode

☐ Employee function

☐ Team

☐ Inform assigned user by eMail

005537

### Keep same assignment

If the "Keep Same Assignment" field is checked, then all the selected tasks are reactivated, but each task will keep the same assignment. If the field is not checked, then all selected tasks are reactivated and assigned to the user, team or employee-function that is entered on the screen.

By default, the "Keep same assignment" field is set. So, the field must be made empty before it is possible to enter another assignment.

A task with status solved is not assigned to anybody. Therefore, if a solved task is reactivated, then field "keep same assignment" is switched off.

### I start working for this task

If "I start working for this task" field is checked, then the assignment is set to myself.

### 37.8.12. Send Email for Assign, Approve, or Transfer

The following table lists an overview of all situations when an email is created.

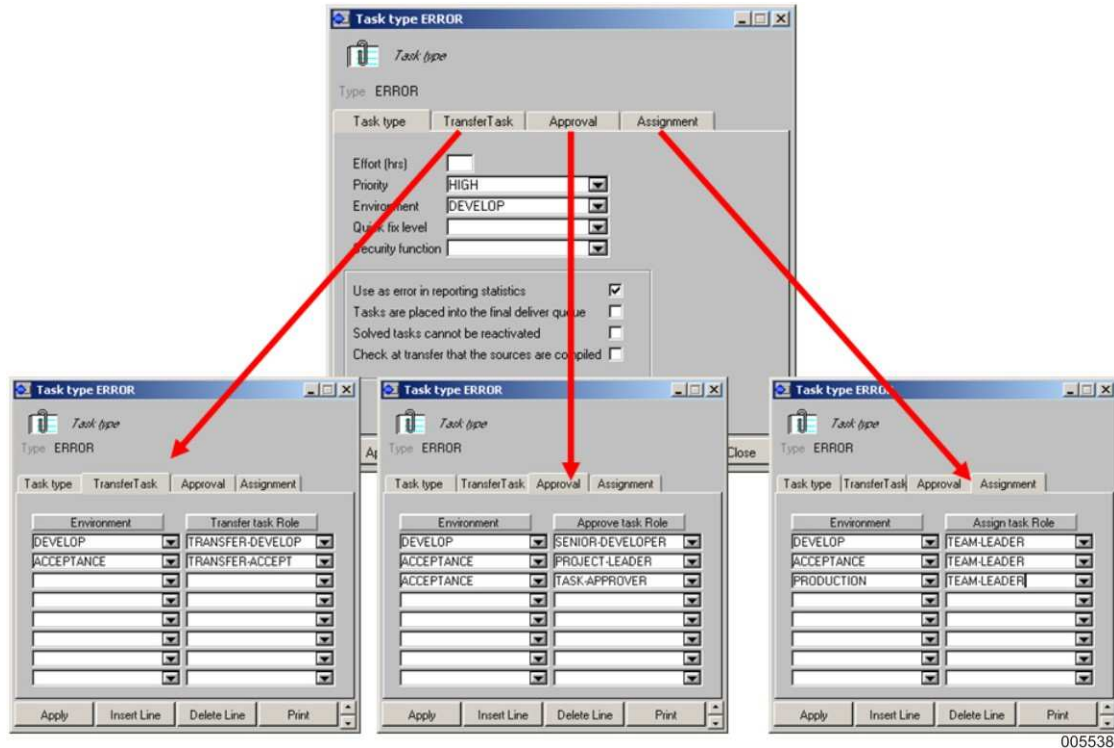
Function	Email to
New task	<ul style="list-style-type: none"> <li>– Send “Task entered” email to the person who reported the task, only if the task has option “Inform status change by email.”</li> <li>– Send “Task to assign” email to the task-assign-role, only if the task is not assigned during the add.</li> </ul>
Change task assignment (modify, assign)	<ul style="list-style-type: none"> <li>– Send “Task assigned” email to the users who must work on the task, only if the task has option “Inform &lt;to handle by&gt; using email.”</li> </ul>
Undo assign	<ul style="list-style-type: none"> <li>– Send “Task undo-assigned” email to the previous user (only if the task has option “Inform &lt;to handle by&gt; using email.”</li> <li>– Send “Task to assign” email to the task-assign-role.</li> </ul>
Task ready	<ul style="list-style-type: none"> <li>• When the task must be approved: <ul style="list-style-type: none"> <li>– Send “Task to approve” email to the task-approve-role.</li> <li>– Send “Task to transfer” email to the task-transfer-role if all required approvals are done.</li> </ul> </li> <li>• When the task does not have to be approved: <ul style="list-style-type: none"> <li>– Send “Task to transfer” email to the task-transfer-role.</li> </ul> </li> </ul>
Transfer to next env.	<ul style="list-style-type: none"> <li>– Send “Task transferred to &lt;env&gt;” email to the project-coordinator, to the system-coordinator and to the environment coordinator of the next environment.</li> </ul>
Task reactivate	<ul style="list-style-type: none"> <li>– Send “Task assigned” email to the users who must work on the task (only if the task has “Inform &lt;to handle by&gt; option using email.”</li> </ul>
Task gets status solved	<ul style="list-style-type: none"> <li>– Send “Task is solved, closed or denied” email to the person who reported the task (only if the task has option “Inform status change by email.”</li> </ul>

Emails are never sent to the user who triggers the email. For example, If I make a task ready, then the emails “task must be approved” and “task must be transferred” will not be sent to myself.

The same email is never sent twice to the same user. For example, if a user is defined as environment-coordinator and as system-coordinator, then he will only receive one email when the task is transferred.

Sending an email depends on the tasktype. The tasktype has three new tabs where you can define the employee-functions that need to be informed by email that a task must be assigned, approved, or transferred.

It is possible to define more than one employee function for each action. All users with one of those employee functions will receive the email, except the users who do not have the task-project in their project list.



### Transfer Task Role

All users with employee func TRANSFER-DEVELOP gets an email if a task with this task-type is ready to be transferred from DEVELOP.

All users with employee func TRANSFER-ACCEPT gets an email if a task with this task-type is ready to be transferred from ACCEPTANCE.

### Approver Task Role

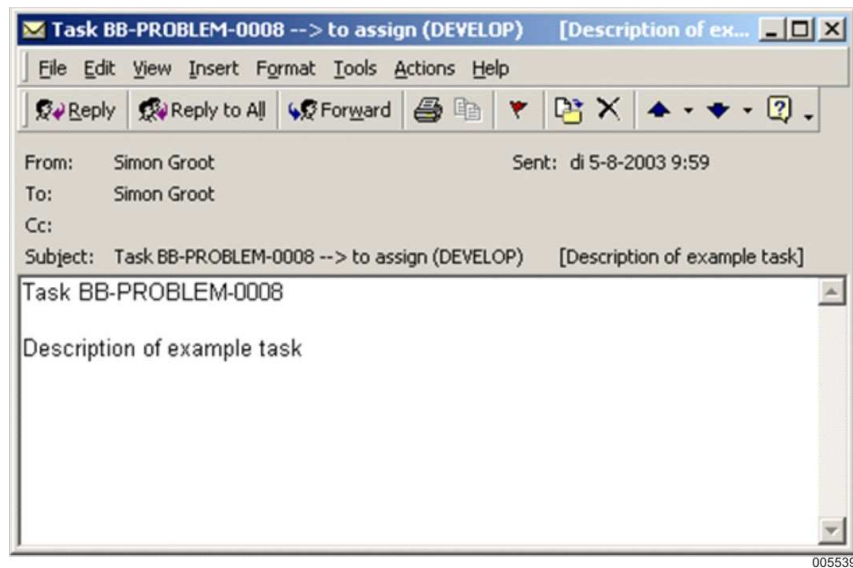
This task must be approved on DEVELOP by any senior-developer, and on environment ACCEPTANCE by any project leader and any task-approver.

All users with one of these employee-functions get an email when the task is ready to be approved.

### Assignment Task Role

All team leaders get an email when a task with this task type must be assigned.

Notice that an email is not sent to users who do not have the task-project in their project list.



The subject of the email shows the task name, the required action, and the first line of the task description.

### 37.9. (Task) Changing to Status Solved

A task can be solved in the following ways:

- Transfer the task to a "solved" environment.
- Function "Close" to close the task immediately, to indicate that no further actions are necessary for this task.
- Function "Deny" to indicate that the task is invalid.

The following actions are made when a task is solved:

- Change the status of the task to solved.
- Change "date ready" to today's date.
- Clear the "to handle by" field to ensure that the task will not appear anymore on user To Do list.
- Add the task to the history of all sources that were adapted because of this task.
- Add all sources that were adapted due to this task to the history of the task.
- Link the current system release to this task to identify in which release the task was solved. This release is linked to the task through the "SOLVED-IN-RELEASE" relation.

## 37.10. (Task) Master Tasks and Dependent Tasks

A dependent task is a task that depends upon another (controlling) task. The controlling task must be transferred to an environment before the depending task can be transferred. A task must be made dependent on another task using drag and drop function.

To make a task dependent on another task, click a task and drag and drop on another task.

A master task is a special controlling task. If a master task is transferred, all its dependent tasks are transferred too.

It is possible to add a new task as a sub-task of an existing task. The existing task must be entered in field "task-group" on the new-task-screen. The existing task becomes automatically master task and it is used to create the name of the new task

### Example

In the next example there is an existing task called FTALG-0360. This existing task is entered in "task group" field to create a new sub-task for FTALG-0360.

**Task :: New**

**Task Definition**

Task Name:

Group:

Project:

Type:

Environment:

Reported by:

Reference:

Inform status change via eMail: ☐

Description:

Short description:

Attachment:

**To Handle By**

Not Assigned: ☒

I start working for this task (current): ☐

UserCode: ☐

Employee Function: ☐

Team: ☐

Inform assigned to handle by via eMail: ☐

Task variable 1:

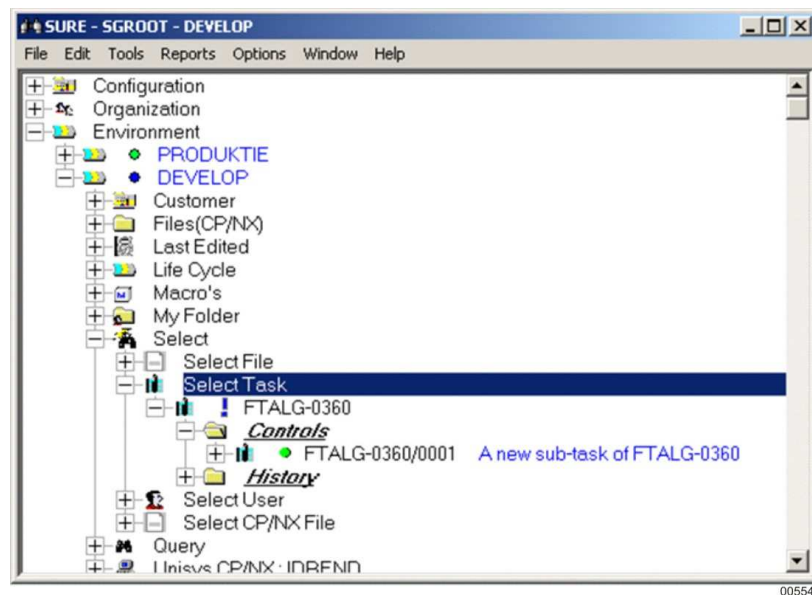
Task variable 2:

Task variable 3:

Apply Cancel

005540

The new task is called FTALG-0360/0001 and it is a sub-task of FTALG-0360.



## 37.11. (Task) Quick Fix

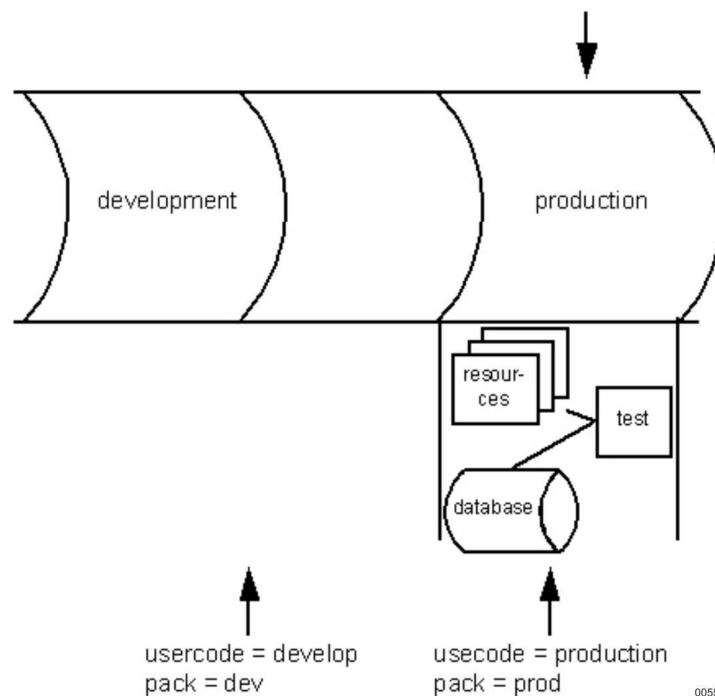
### 37.11.1.(Task) Quick Fix Preface

A quick fix is the procedure to change software or definitions in an environment other than the default. An organization could implement this quick fix procedure as change a program in the production environment due to a serious error and do not take any new features into production. The SURE software provides a generic quick fix mechanism that supports multiple environments and different behaviors for different entities. For this reason, the definition of a quick fix results in the abstract definition defined in the first sentence.

Impact overlap detection and verification is a used procedure in the SURE software. This impact overlap does not prevent changes in software due to multiple tasks. Impact overlap just warns you against unexpected functionality taken into production. A quick fix, however, is the procedure to modify software for a single error using the software running in production.

To explain in some detail the complexity incorporated in the quick fix procedure, the programmer's job is taken as an example. A programmer modifies a program using the resources applicable for the production environment. After modification, the program is tested in an environment containing database software and other environmental software. Suppose that the quick fix is done at the development environment. In that case, the environment contains all resources matching the new developed software. This often is quite different from the production environment.





Changing and testing software in an environment other than development requires the ability to work in that environment. Changing the software is just one part, testing is just as important. ClearPath server usercode or family specification needs to be changed if one changes from regular development to quick fix mode. SURE supports quick fix procedures as a seamless operation within the SURE life cycle support. Changing usercodes required for quick fix procedures is done by SURE.

Creating a quick fix results in a feedback mechanism for development. SURE supports a queue of tasks that were used as quick fix. This queue may be processed and that applies the modifications to another environment. SURE supports this queuing mechanism and automates the process of introducing the modifications.

Another aspect of the quick fix procedure is the authorization. Handling of a serious production error may introduce temporary privileges required to do the job. However, in normal development, these privileges are not present. SURE supports a temporary employee-function if working for a task. Using this mechanism, a person may obtain more privileges if working for that task.

As described in this preface, a quick fix is a set of definitions that results in the ability to change software for production errors or historical releases. SURE provides a seamless procedure and reports the presence of quick fixes.

### **37.11.2.(Task) Quick Fix Definition**

A quick fix is a change of the application software at an environment other than the default maintenance environment.

Only in special circumstances a quick fix should be made:

- An urgent error occurs in the live production environment, and this error has to be solved immediately.
- An error is reported that has to be solved on a specific date, but the software where the error has to be solved is already in maintenance for other reasons.
- Maintenance is done on a historical release.

All three situations are described in detail in the following paragraphs.

#### **37.11.2.1.(Task) Quick Fix: Urgent Errors**

An urgent error is reported in the live production environment, and this error has to be solved immediately.

Many sites have a special support group that is responsible for the live production environment. If an urgent production error occurs then these people must have the flexibility to make a quick fix modification directly in the production software. In cases of urgent production errors, time is always a critical factor. In these cases, the quick fixer does not want to be hindered by all kinds of transfer procedures from development to production. He adapts the source directly in production, and takes care that the quick fix adaptation is reported to the development department. The development department can implement the quick fix adaptations in a later stage.

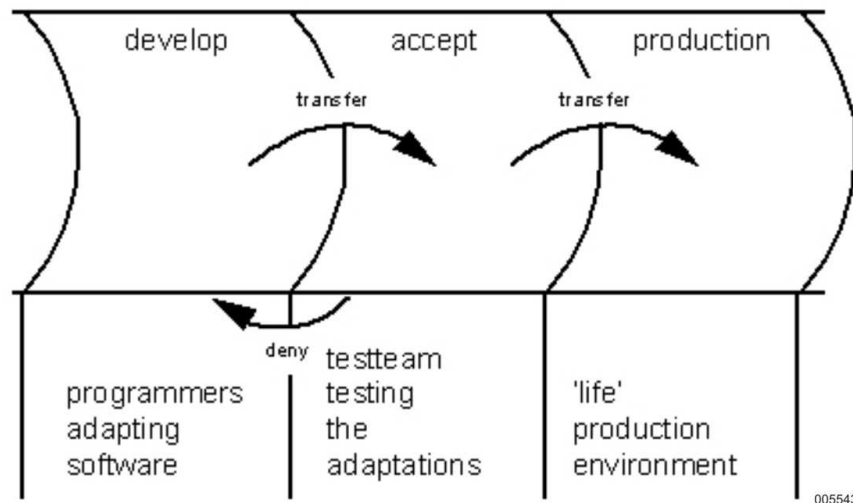
#### **37.11.2.2.(Task) Quick Fix: Overlap with Other Tasks**

An error is reported in the live production environment. This error has to be solved before a specific date, and can be fixed by a developer, but the software where the error has to be solved is already in maintenance for other reasons.

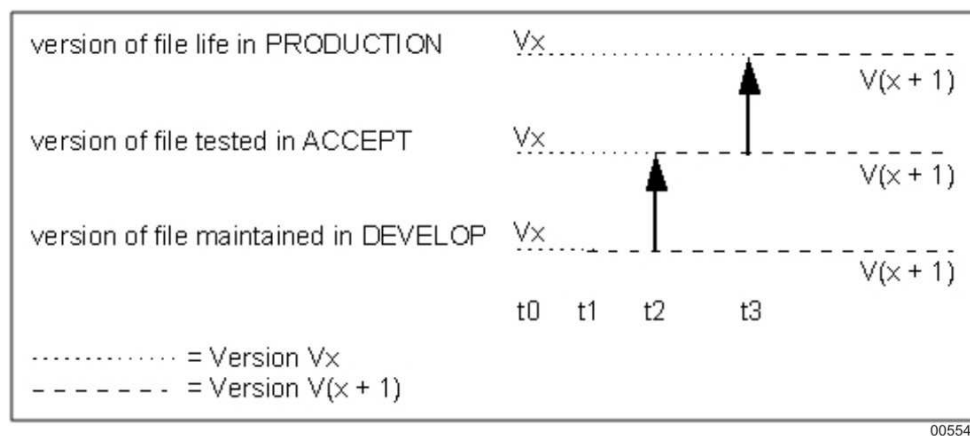
The usual life cycle of software goes from maintenance through a testing phase to production. When an error occurs in production, or when extra features have to be built, the program is taken into maintenance again.

The above-described situation is well supported by the SURE software. It is not difficult to install a repository with three environments: DEVELOP, ACCEPT and, PRODUCTION.

Programmers work in the DEVELOP environment. When they finish their job, the modified sources are promoted to ACCEPT, where a test team can test the new adaptations (and verify that the old functionality of the program still works). When all test results are OK, the adapted source is transferred to production, where it will be used in a life-situation.

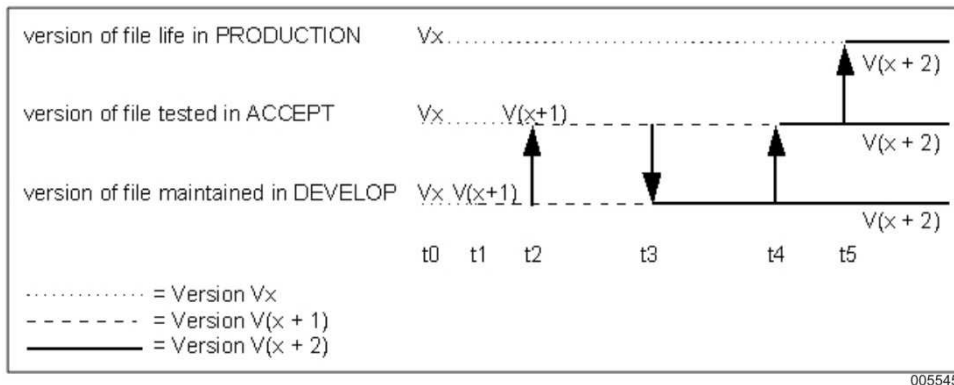


The following picture shows the status of a file and the various actions on that file.



- Timestamp **t0**: version **Vx** of the file is active in all environments.
- Timestamp **t1**: the file is adapted and version **V(x + 1)** becomes active in **DEVELOP**.
- Timestamp **t2**: the adapted source is promoted to **ACCEPT**. FileVersion **Vx** is still active in **PRODUCTION**, but **V(x + 1)** is already active in **DEVELOP** and **ACCEPT**.
- Timestamp **t3**: the new version of the file is transferred to **PRODUCTION**. The new version **V(x + 1)** is now active in all environments, and the old version **Vx** is not used anymore.

It is obvious that the path from maintenance to production is not always as smoothly as described above. Adaptations that a programmer made can be rejected by the test team (perhaps several times rejected) before they are accepted and promoted to the production environment.

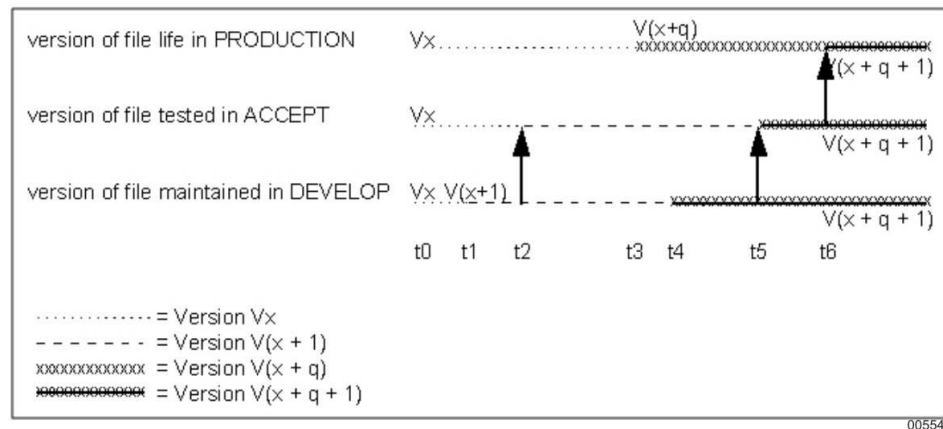


- Timestamp  $t_0$ : version  $V_x$  of the file is active in all environments.
- Timestamp  $t_1$ : the file is adapted and version  $V(x+1)$  becomes active in DEVELOP.
- Timestamp  $t_2$ : the adapted source is promoted to ACCEPT.
- Timestamp  $t_3$ : the modifications are rejected by the test team and the task is reactivated for DEVELOP. FileVersion  $V_x+1$  remains active in ACCEPT, but the programmer starts working on enhanced adaptations in FileVersion  $V_x+2$ .
- Timestamp  $t_4$ : the enhanced adaptations are promoted to ACCEPT.
- FileVersion  $V_x+2$  is now active in DEVELOP and ACCEPT. FileVersion  $V_x+1$  is gone.
- Timestamp  $t_5$ : the adaptations are accepted by the test team and the file is transferred to PRODUCTION. Version  $V_x+2$  is now used in all environments. Version  $V_x$  is not used anymore.

Run-time errors of programs in the live production environment occur. It is obvious that these errors must be fixed with a high priority.

This can give conflicting situations if the source that has to be adapted to solve the run-time error, is already in maintenance for other reasons (for example, modified for a new feature request). The error has to be solved immediately, but the adaptation for the other task cannot go live yet.

The solution for the above-described conflicting situations is to fix the error directly in the file version that is available in the production environment. The fix can be made in the other environments in a later stage.

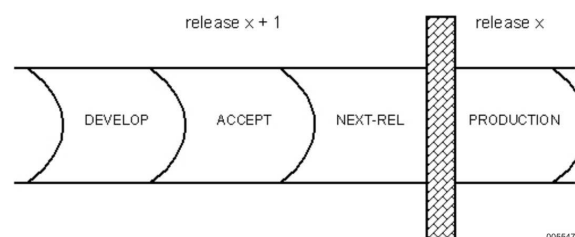


- Timestamp  $t_0$ : version  $V_x$  of the file is active in all environments.
- Timestamp  $t_1$ : the file is taken into maintenance to implement a new feature.
- Timestamp  $t_2$ : the new feature is transferred to ACCEPT.
- Timestamp  $t_3$ : a run-time error is reported and the file is directly adapted in PRODUCTION. The new FileVersion is  $V_x+q$ .
- Timestamp  $t_4$ : the error is also fixed in DEVELOP (version  $V_x+1+q$ ).
- Timestamp  $t_5$ : the fix in DEVELOP is transferred to ACCEPT. The error is now solved in all environments but the new feature is now only available in DEVELOP and ACCEPT.
- Timestamp  $t_6$ : the file is transferred to PRODUCTION and in a stable situation. The error solution and new feature are active in all environments.

Notice that the software errors, reported in the production environment, can only be solved in the default maintenance environment if the error situation is reproducible in that environment. If it is not possible to reproduce the error situation in the maintenance environment (by creating the correct set of test data), the error should be fixed directly in the production environment through a quick fix.

### 37.11.2.3.(Task) Quick Fix: Adaptations in a Historical Release

It is possible to create historical environments in the repository. Normal environments are part of the normal life cycle of the application software: production; in maintenance for an adaptation; through test/accept back to production. A historical environment identifies a software release that is not equal to the release on which the development staff is currently working. The following picture shows this situation.



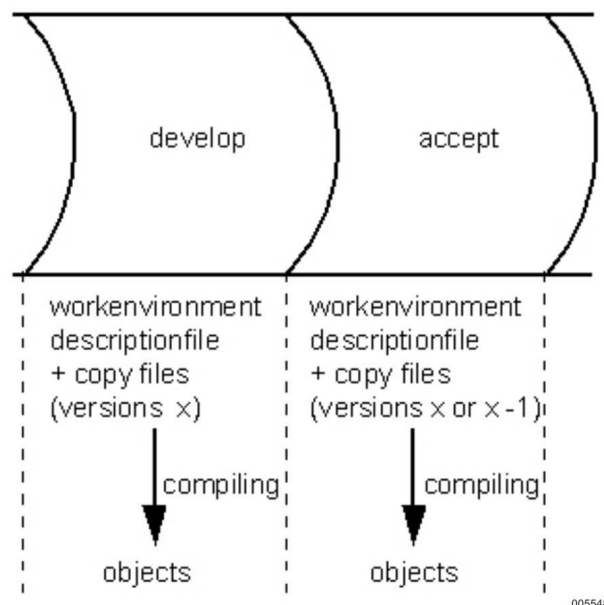
Software release REL-x is active in the live production environment PRODUCTION. The development staff is working on the next release REL-(x+1). The life cycle of the software goes from DEVELOP, through ACCEPT to NEXT-RELEASE, and later on again to DEVELOP for the next modification.

If an adaptation has to be made for the live production (an error solution or a new feature implementation), then this modification must be made directly in the production environment (since it is not possible anymore to transfer the adaptations, because environment PRODUCTION is blocked for transfer). This means that each adaptation in a historical environment must be made using a quick fix.

### 37.11.3.(Task) Quick Fix Work-Environment

You can define a different work-environment for each system and for each repository environment.

The work-environment identifies the directory where global files are placed that is necessary for the programmer activities. These files are: database description file, copy files, and jobs. The work directory of a system must be available for the programmer when he is working on a task of that system; otherwise, he cannot perform his daily job. When the programmer is logged on using CANDE then the work directory must be visible according to the standard visibility rules on ClearPath Enterprise Servers.



005548

When a quick fix is made, the software adaptations are done in an environment that is not equal to the default maintenance environment. The programmer has to be logged on through CANDE under the correct work-environment too; otherwise, he will compile his source using wrong versions of description files and copy files; otherwise, he will test his compiled program using wrong data-files and databases.

### 37.11.4. (Task) Quick Fix Procedure

**Reprocess Quick Fix : SG\_0322 from environment PRODUKTIE to DEVELOP**

Task Name: SG\_0322      Reported By: SGROOT  
 Status: DEVELOP      Solved By: SGROOT

This is a long message for a programmer developing large systems.  
 The integration required with the internet browser should be made available through transactions processed in various different ways by simple routines.

Reprocess:

Quick Fixed in Environment:       PRODUKTIE / DEVELOP      View Changes  
 Reprocess in Environment:       PRODUKTIE      View Changes      List  
    DEVELOP      View Changes      List  
    DEVELOP      Preview Apply

Entity	Name	PRODUKTIE	DEVELOP	Summary	Status	Index
Task	SG_0322				ToDo	0000
File	SIMON/C/2	1.5 (SGROOT)	2.1 (SGROOT)	*..Q	Done	0001
File	SIMON/C/3	1.3 (SGROOT)	1.2 (SGROOT)	*..Q	ToDo	0002

Reprocess      Check Out      Apply      Override      Close

005549

The "Reprocess Quick Fix" dialog shows the name of the task that has to be reprocessed, the status and description of that task, and the files that are linked to the task.

#### 37.11.4.1.(Task) Quick Fix: Different Work-Environment

A quick fix is made by defining a task with an environment not equal to the default maintenance environment of that file.

The SURE software (if local edit is set to true) copies the source to the work-environment of ClearPath Enterprise Servers. On the PC (if local edit is set to true) the source is copied to the work directory specified for that environment. Therefore, editing and compilation can be done without disturbing other users of this file in other environments.

Testing of the executables must be done from the quick fix work-environment on ClearPath Enterprise Servers.

### 37.11.4.2.(Task) Quick Fix: Changes Status

After a source is quick-fixed, the environment where the quick fix is made is marked with a "Q" in the status summary. A quick fix relation between the source and the task is added in the repository.

#### Example

Consider a repository with three environments: DEVELOP, ACCEPT and PRODUCTION.

Program P1 has the following status summary

```
DAP
..*
```

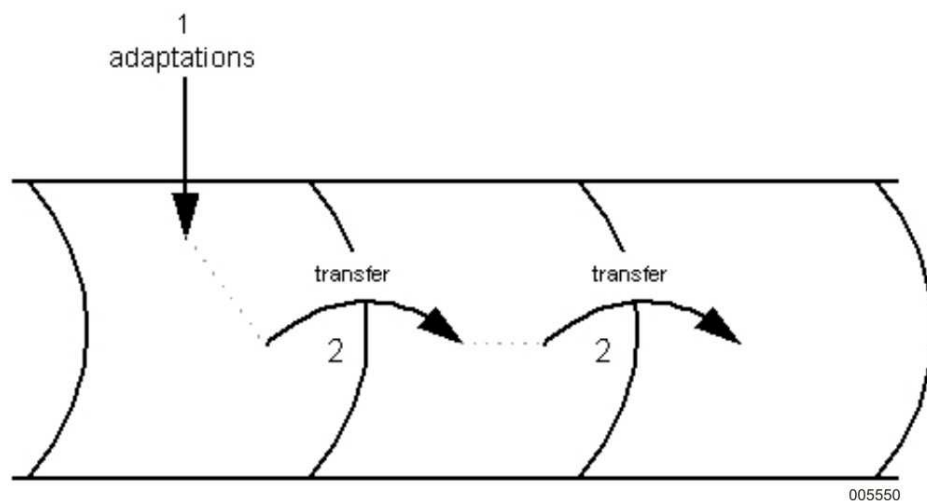
After a quick fix of the program P1 on environment PRODUCTION, the status summary changes to:

```
DAP
..Q
```

### 37.11.4.3.(Task) Quick Fix: Reprocess Modifications

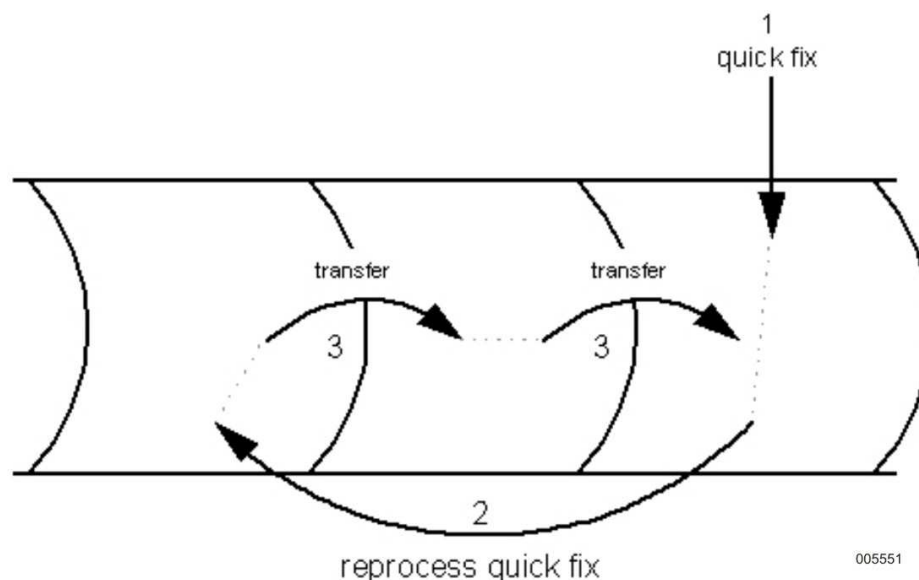
Adaptations that were made as a quick fix have to be reprocessed on the default maintenance environment. This procedure is supported through the task menu selection "Reprocess QuickFix." A queue with the quick fix tasks is available in the "Team Task QuickFixed" folder or "My Task Quick Fixed" folder. Each user has his own queue of quick fix tasks: the quick fixes that meet the programmer's project and employee function.

In a normal situation, the adaptations on a source are made in the default maintenance environment, and are later transferred to next environments.



In a quick fix situation, the adaptations are made in a non default maintenance environment, then reprocessed in the default maintenance environment, and after all transferred to next environments.





The procedure to reprocess a quick fix works as follows:

1. Select the name of the quick fix task to reprocessed and select **Reprocess QuickFix** using the popup menu.
2. The environments where the quick fix can be reprocessed are given on the reprocess quick fix screen. One of these environments has to be specified.
3. All sources that were adapted to solve the quick fix are mentioned. The status summary of each source is also given. In case of a source the file version in the target environment and the file version in the quick fix environment is shown.
4. Every source must be selected individually and SURE will propose the actions to be taken.

Button	Action
View Changes	This option shows the differences between the files. It can show the difference between the quickFix environment and the target environment, the changes made in the QuickFix environment or the changes made in the target environment. This function is used as a supporting function for the developer.
List	This option shows the delta files. It can show the delta file in the QuickFix environment or the delta file in the target environment. This function is used as a supporting function for the developer.
Reprocess	This option is enabled if the entity is not adapted in the target environment. This function will copy the contents of the QuickFixed environment to the target environment.
Check Out	Do a checkout of the file in the target environment so that the changes can be made manually.

Apply	This option is enabled if the entity is adapted in the target environment. This function will merge the changes made in the QuickFix environment into the source in the target environment.
Override	This option is enabled if the entity is adapted in the target environment. This function will copy the contents of the QuickFixed environment to the target environment. Note that this function will discard the changes that were made in the target environment.

Notice that for the above described actions (Reprocess, Check Out, Apply and Override), the task is reactivated in the target environment.

### Example

Consider a repository with three environments: DEVELOP, ACCEPT and PRODUCTION.

Task QF1 is solved as a quick fix in environment PRODUCTION and two sources are adapted because of this quick fix: P1 and P2.

Program P1 is already in maintenance in environment DEVELOP for another reason.

The continuation screen with the sources that were quick fixed because of task QF1 gives the following information.

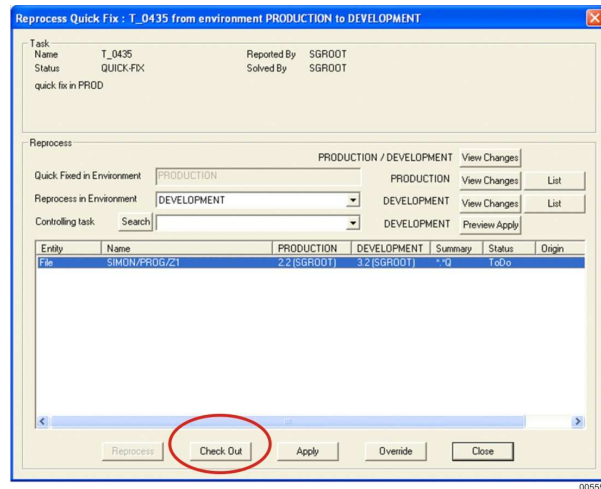
Name	RIS-entity	Version	Status
P1	(FILE)	5.1 4.2 *.*	DEVELOP
P2	(FILE)	3.1 3.2 ..*	PRODUCTION

File P2 can be reprocessed directly from quick fix environment PRODUCTION to target environment DEVELOP, since there are no intermediate adaptations on this file between version 3.1 in DEVELOP and 3.2 in PRODUCTION.

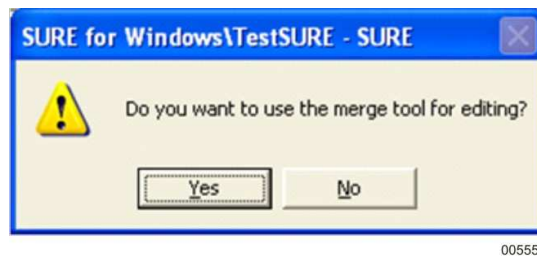
File P1 can NOT be copied directly from PRODUCTION to MAINTENANCE, since the previous file version in quick fix environment PRODUCTION was 4.1 and that is not equal to the current file-version in DEVELOP (5.1). The Apply function will create a patch file in the programmer's work-environment called: PATCH/P1/4000002. This patch file is merged into the current available source in DEVELOP.

### 37.11.5.Example: Reprocess Quick Fix Using Check Out

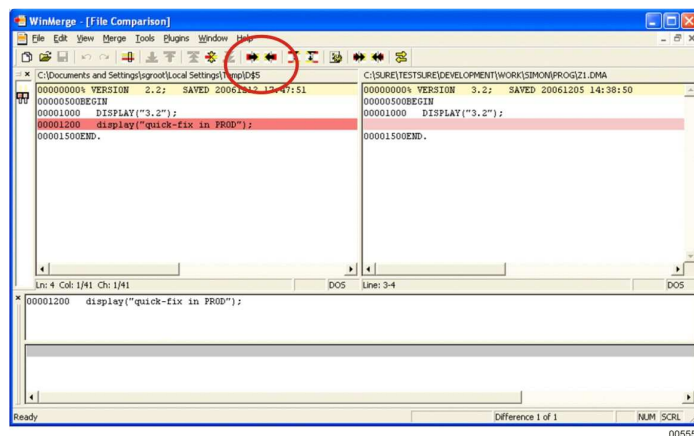
The Check Out button on the Reprocess Quick Fix screen offers the possibility to apply the quick fixes manually.



With the Check Out button it is possible to load the current source and the quick fixed source in the compare utility. Some compare utilities allow moving the modified lines to the other side.



In this example, the two file versions are loaded in the compare utility WinMerge. The modified lines are highlighted. It is very easy to move the modified lines to the checked out workflow.



## Task

Save the file in the Compare Utility when the modified lines are moved to the checked out workfile, and close the Compare Utility to return to the Quick Fix screen.

Reprocess Quick Fix : T\_0435 from environment PRODUCTION to DEVELOPMENT

Task Name: T\_0435, Status: QUICK-FIX, Reported By: SGR00T, Solved By: SGR00T, quick fix in PROD

Reprocess: PRODUCTION / DEVELOPMENT

Quick Fixed in Environment: PRODUCTION, View Changes, List

Reprocess in Environment: DEVELOPMENT, View Changes, List

Controlling task: Search, DEVELOPMENT, Preview Apply

Entity	Name	PRODUCTION	DEVELOPMENT	Summary	Status	Origin
File	SIMON/PROG/Z1	2.2 (SGR00T)	3.2 (SGR00T)	*-Q	Done	

Buttons: Reprocess, Check In, Apply, Override, Close

The "Checked Out" button is changed to "Checked In" so that you do not have to leave the quick fix screen.

The Check In screen has a "View MY changes" button. This allows you to verify the changes for the last time before you do the check in.

Check In: T\_0435

Environment: Usercode: SGR00T, Environment: DEVELOPMENT, Current task: T\_0435

Local editing: ☒ Enabled, ☐ Send complete workfile (do not send difference file), ☐ Remove CR/LF

Local work file: C:\SURE\TESTSURE\DEVELOPMENT\WORK\SIMON\PROG\Z1.DMA

Local source file: C:\SURE\TESTSURE\DEVELOPMENT\SOURCE\SIMON\PROG\Z1.DMA

Concurrent file maintenance: Current file version: 3.2, Changes relative to version: 3.2

Buttons: View OTHER change, View MY changes, Preview merge, Apply merge and Edit, Replace with my file

Other users assigned

Buttons: OK, Cancel

## 37.12. (Task) Delivery

This information is only for RIS users.

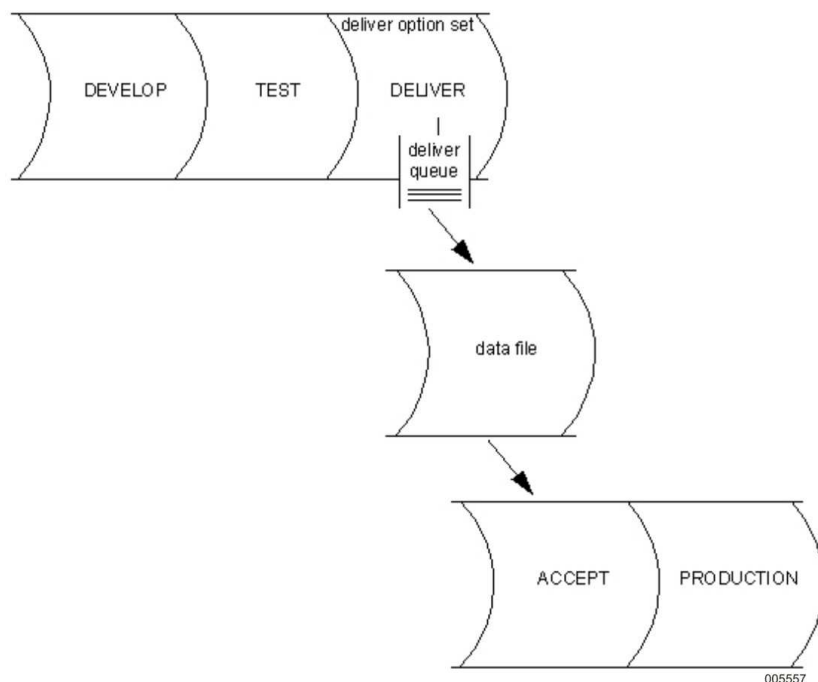
It is possible to dump a task from the repository into a file, and to load this file into another repository. This mechanism is called delivery.

The deliver function presents a screen where it is possible to add tasks into the deliver queue, to show the current contents of the deliver queue, and to delete tasks from the deliver queue.

A batch program reads the deliver queue and dumps all tasks that are resident in this queue into a file. A second batch program can read this file to copy the information from this file into a target repository.

You can add tasks into the deliver queue of an environment if the deliver option is set for that environment. To set this option using dialog, perform the following steps:

1. Click **<environment>**.
2. Click **Properties**.



A task can only be delivered if it is in a stable situation in the delivery environment. If a user is working on the task in the delivery environment, then it cannot be delivered.

The task is stable in an environment if it has `SUB-STATUS(OK)` on that environment, and the task status is not equal to the environment.

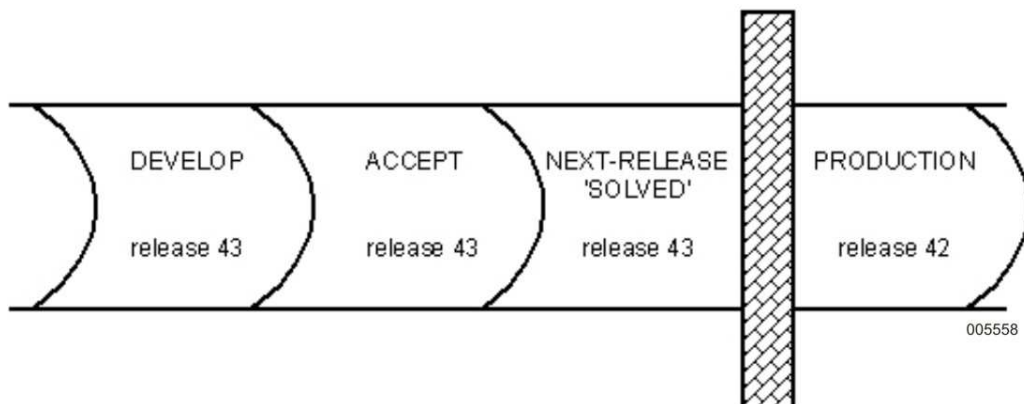
### Example

Consider task T1 and a repository with environments DEVELOP, TEST, ACCEPT and PRODUCTION. The task is transferred from DEVELOP to ACCEPT, and therefore it has `SUB-STATUS(OK)` in environments DEVELOP and TEST. The current status of the task is `accept` (it was transferred to that level). This means that the task is stable in DEVELOP and TEST, and unstable in ACCEPT and PRODUCTION (no sub-status OK).

It is possible to add a single task to the deliver queue, but is also possible to add a total release. In that case, all tasks that were solved in the indicated release are added to the queue. A task gets the “solved in release” indication as soon as it gets status solved. The current release of the system/project to which the task belongs is then linked to the task.

### Example

Consider task T1 that belongs to project PP. Project PP is a sub project of system BASE.



The repository is divided in four environments and two releases. Environment PRODUCTION refers to the live production environment that is working with the previous release. The development staff is working on the next release 43.

If task T1 is transferred to environment NEXT-RELEASE, then it gets `STATUS(SOLVED)`. Task T1 belongs to system BASE and therefore the task gets also `SOLVED-IN-RELEASE(43)`.

### 37.12.1.(Task) Deliver Commands

The following commands are possible on the deliver screen.

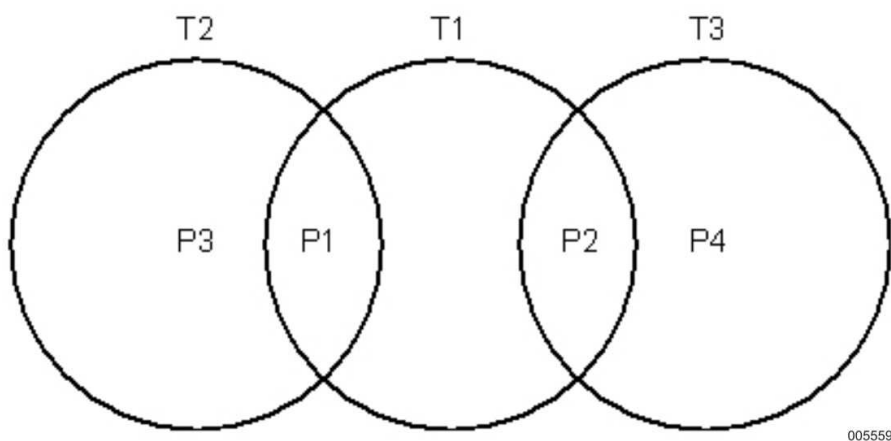
ADD	<task>	This function places a task into the deliver queue.
ALL	<release>	This function places all tasks that were solved in <release> into the deliver queue.
ADD	<release>	This function places all tasks that were solved in <release> but not yet delivered earlier into the deliver queue.
DEL	<task>	This function deletes one task from the deliver queue.
INQ	<task>	This function checks for overlaps with other tasks.
Specify on QUEUE		This function shows the contents of the deliver queue.
Specify on PURGE		This function empties the deliver queue.

### 37.12.2. (Task) Delivery Overlap

Each time a single task is added into the deliver queue, a check is made for overlaps with other tasks. Tasks that are already delivered earlier are ignored in this overlap check. If a task has overlap with another task, and this other task is not delivered, then this might cause problems in the target repository where the delivery file is loaded.

#### Example

Consider the programs P1 and P2 that were adapted because of task T1. Programs P1 and P3, both adapted because of task T2. Programs P2 and P4, both adapted because of task T3.



Task T1 has overlap with tasks T2 and T3. Task T3 is already delivered earlier, so it is ignored during the overlap check.

If task T1 is added into the delivery queue, then the overlap analysis will report that there is an overlap with task T2. If this warning is ignored, then the adaptations in program P1 that were made because of task T2 are delivered, but program P3 will not be delivered. This may cause problems in the target repository.

The overlap analysis at delivery time is triggered by HISTORY relations between tasks and files. These relations are added if the “delivery” option is set for the environment.

The procedure works as follows:

- The “delivery” option can only be set for environments with the “solved” option enabled. The delivery option has two functions:
  - Allow that the delivery queue can be filled for an environment.
  - Add HISTORY relations for the tasks that get status solved. All PROBLEM relations are changed to HISTORY relations as soon as the task gets status solved. If the delivery option is not set, then the PROBLEM relations are removed and not changed.
- If a task is delivered finally, then the history relations are removed. This prevents an overlap warning on tasks that are already delivered.
- If the release number of the system is changed, then the history relations are removed of all tasks that belong to that system. This prevents from overlap warnings on tasks that were solved in another release.
- Tasks that have HISTORY relations are not yet delivered and will be mentioned in the overlap analysis if they overlap with the task that is placed in the queue.

### 37.12.3. (Task) Delivery Final or Test

When the delivery function is left and the user is authorized to do a final delivery, the system presents a verification screen where it can be specified that this delivery is a final delivery or a test delivery. In all other cases, the delivery will be a test-delivery. The purpose of a test delivery is to check if the task arrives correctly into the target repository, and that the generations/compilations and test runs are all correct. During the time that the test delivery is busy, the task and its linked sources cannot be changed. This ensures that the same versions of sources are dumped at the final delivery as during the test-delivery.

A task that is delivered for test gets DUMP-STATUS(ACTIVE). All files that are linked to the task get DUMP-STATUS(<task>) and this relation blocks them for further adaptations.

A file that is delivered finally gets DUMP-STATUS(DUMPED). The batch program (that does the actual dump from the repository into a file) removes DUMP-STATUS ACTIVE for all tasks that have also DUMP-STATUS(DUMPED), and removes then the DUMP-STATUS from the file. This unlocks the files against other adaptations.



### Example

Consider program P1 that was adapted because of task T1.

Task T1 is delivered for test.

Entityowner:class(asset)	Reason
PROBLEM   T1 : DUMP-STATUS ( ACTIVE ) FILE-CONTROL   P1 : DUMP-STATUS ( T1 )	These relations block file P1 against further adaptations.

When task T1 is delivered finally (when the test phase is finished).

Entityowner:class(asset)	Reason
PROBLEM   T1 : DUMP-STATUS ( ACTIVE ) PROBLEM   T1 : DUMP-STATUS ( DUMPED ) FILE-CONTROL   P1 : DUMP-STATUS ( T1 )	An indication is added that T1 is now dumped finally. The batch uses this indication.

The batch program checks the tasks that are dumped finally and removes the locks.

Entityowner:class(asset)	Reason
PROBLEM   T1 : DUMP-STATUS ( DUMPED )	This relation gives the timestamp that the task was dumped finally.

It may be necessary that a program that is blocked for adaptations, has to be modified urgently because of a run-time error in production. There are two ways to override a lock on a file.

- Reactivate the task that was delivered for test and that blocks the file. A blocked file can always be modified for the same task.
- Define another task as power task. A power task has the power to override locks on files.

### 37.12.4. (Task) Delivery Batch Procedure

The RESPECT/REPOSITORY("DUMP-REPOSITORY") batch program reads the deliver queue and dumps all the tasks into a file called RESPECT/DUMP/TRANSFER.

The following actions can be performed for each dumped task:

- Dump the task, task description, and other attributes into file RESPECT/DUMP/TRANSFER.
- Read the history of the task and dump all files, and RIS-id's that are mentioned in this history also into file RESPECT/DUMP/TRANSFER.

- Read the history of the task and dump all files that are mentioned in this history to disk under RESPECT/DUMP/SOURCE/<filename> directory. The repository attributes of the file are copied into RESPECT/DUMP/TRANSFER file.
- Copy the special commands that are linked to the task also in RESPECT/DUMP/TRANSFER file.
- Read the history of the task and place all formats that are mentioned in this history into the deliver-format queue. The repository attributes of these formats are copied into RESPECT/SURE/TRANSFER file.
- Check the tasks that are delivered finally, and remove the locks and HISTORY relations of these tasks.

The formats that have to be delivered are placed into the deliver-format queue. The RIS/EDITOR;VALUE=9996 program reads this queue and dumps the queued formats into RESPECT/DUMP/SCREENS/TRANSFER file.

After the dump phase, the following files are resident on disk:

RESPECT/DUMP/TRANSFER	Contains delivered tasks and RIS-id's and information about delivered files and formats.
RESPECT/DUMP/SCREENS/TRANSFER	Contains delivered formats.
RESPECT/DUMP/SOURCE	A directory with the delivered files.

The delivered files are loaded into a target repository using the RESPECT/REPOSITORY(LOAD-REPOSITORY) batch program . This program reads the RESPECT/DUMP/TRANSFER delivery file and loads all tasks, sources, RIS-id's and format attributes that are mentioned in this file.

The status of each loaded task is set equal to the environment name on the task that was loaded.

Sources will be loaded on the target repository with the same file version and cycle as they had in the original repository. However, if the source is already available in the target repository with the same file version-cycle and that source is not identical to the one that is going to be loaded, then the new source will be loaded with a cycle that is one higher.

### Example

Consider file F1 with version/cycle = 1.1

and file F2 with version/cycle = 2.1

and file F3 with version/cycle = 3.1

and file F4 with version/cycle = 4.1

File F1 is not yet available in the target repository so it will be loaded under version 1.1.

File F2 is already available in the target repository under version 1.1. This differs from the new version, so this new version 2.1 will be loaded and version 1.1 is removed at a later stage.

File F3 is already available in the target repository under version 3.1 and this file is identical to the file that has to be loaded. No further actions will be made for this file because the correct file is available.

File F4 is already available in the target repository under version 4.1, but this file is not identical (match) to the file that has to be loaded. The new file will be loaded under version 4.2, and a match between 4.1 and 4.2 will be created.

The delivered formats have to be loaded using a second batch program RESPECT/REPOSITORY(LOAD-FORMATS TRANSFER). This program reads the RESPECT/DUMP/SCREENS/TRANSFER file and loads all mentioned formats from this file into the target repository.

### **Example Job**

```
BEGIN JOB WFL/DELIVER/DUMP;
  REMOVE RESPECT/DUMP/=;
  RUN OBJECT/RESPECT/REPOSITORY("DUMP-REPOSITORY");
  RUN OBJECT/RIS/EDITOR;TASKVALUE = 9996;
END JOB.

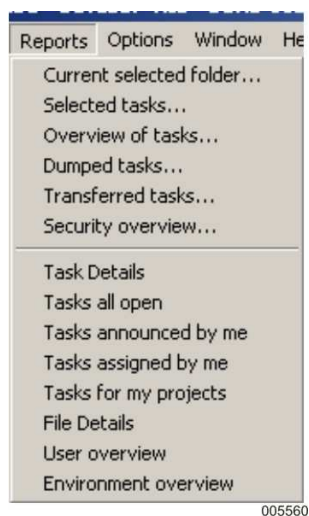
BEGIN JOB WFL/DELIVER/LOAD;
  IF FILE RESPECT/DUMP/TRANSFER IS RESIDENT THEN
    RUN OBJECT/RESPECT/REPOSITORY("LOAD-REPOSITORY");
  IF FILE RESPECT/DUMP/SCREENS/TRANSFER IS RESIDENT THEN
    RUN OBJECT/RESPECT/REPOSITORY("LOAD-FORMATS TRANSFER");
  END JOB.
```

## 37.13. (Task) Overviews

The available task overviews can be separated in two groups:

1. Overviews that are created by a batch program that runs on the mainframe. The layout of these overviews cannot be customized. The contents of these overviews are customizable using input parameters.
2. Overviews that are created through a Windows RTF input file and integration with Word or Excel. These overviews are fairly easy to customize, and it is also possible to create site specific overviews.

To create a report using the **Toolbar** function, click **Reports**. The following menu is shown.



The menu choices above the line separator are mainframe overview. The ones underneath the separator are Windows RTF overviews.

### 37.13.1.(Task) Customer Overview

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Overview of tasks**.
4. Click **List option = 'Customer Overview.'**

This is an overview of tasks that are reported by the customer. It shows the tasks that were reported or solved during a period plus a status overview at the end of the period. This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.

The customer overview shows all tasks that are reported by a customer or by a department.

### Example

Consider task T1 that is reported by XX, and task T2 that is reported by user U1 of department XX.

If a customer overview is created for XX, then both tasks will be selected.

The customer overview consists of three parts:

1. Tasks that are reported by the customer or department during a period.

Tasks are selected if:

<period from> LEQ <task date received> LEQ <period to>

2. Tasks that are solved for a customer or department during that same period.

Tasks are selected if:

<period from> LEQ <task date ready> LEQ <period to>

3. Tasks of that customer or department that are still active at the end of the period.

Tasks are selected if:

<task date received> LEQ <period to>

AND <task date ready> GEQ <period to>

The customer or department can be defined on the continuation screen, where it is also possible to enter other selection criteria (such as the period).

All three sub-overviews are sorted by task number. The output example shows an overview of features of customer BBL concerning January 1996.

### Output Example

Overview of announced tasks

BBL 960101 - 960131 --- Page 1

Selection:

Problem-type = FEATURE  
Announced-by = BBL  
Period from = 960101  
Period to = 960131

Problem	Reported by	Entered	Type	Priority	To handle by	Status
SURE0581	BBL	960110	FEATURE	HIGH		ENTERED
BBL-1616 SURE status overview						
Description: General overview of a source for all environments defined. See enclosure for more information.						
SURE0584	BBL	960112	FEATURE	HIGH		SOLVED
BBL-2087 SURE/COMPILE						
Description: Please show, at the end of the run of SURE/COMPILE, in the taskvalue, the number of the compiled sources. Plus an indication on the compilation overview.						

-----  
Overview of tasks that were solved during period

BBL 960101 - 960131 --- Page 2

Problem	Resolved by	Ready	Type	Release	Reported by	Status
RIS0736	RAYMOND	960103	FEATURE	43	BBL	
255 NFS TRANSFER TASK						
Description: When INQ-ing on a dependent task in the transfer screen, no message is given						

```

Only at the moment one transfers the tasks, a message is given.
Also for Master-task.

SURE0584      BBL      960112 FEATURE      HIGH      SOLVED
BBL-2087 SURE/COMPILE
Description: Please show, at the end of the run of SURE/COMPILE, in the taskvalue, the
number of the compiled sources. Plus an indication on the compilation overview.

-----
Overview of tasks that were active on date      BBL  960101 - 960131 --- Page 3

Problem      Reported by      Delivery Type      Priority      To handle by

SURE0541      BBL      FEATURE      HIGH
BBL-2034
Description: Make it possible to choose the printer at "WRITE"-TIME:
Present additional screen with default printer and possibility to change
this printer.
The "WRITE"-function should take the chargecode into account.

SURE0581      BBL      FEATURE      HIGH
BBL-1616 SURE status overview
Description: General overview of a source for all environments defined. See enclosure
for more information.

COUNTS FOR PERIOD 960101 - 960131
-----
CUSTOMER:      <-ERRORS->
REPORTED:      SOLVED:      ACTIVE:      REPORTED:      SOLVED      ACTIVE
BBL      2      2      2
ALL      2      2      2
-----

```

### 37.13.2. (Task) Overview "Tasks to handle by ..."

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Overview of tasks**.
4. Click **List option = "Tasks to handle by."**

This overview gives for each user ID or employee function the tasks that have to be handled by that usercode or employee function. Tasks that are solved or transferred do not appear on the list. This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.

The listing is sorted by "to handle by" and task number, where each new settler starts on a new page.

Extra selection criteria can be entered on a continuation screen. If a period is entered on this screen then only tasks that were active during that period will be selected. (<task date received> LEQ <period to> AND <task date ready> GEQ <period to>)

The example output shows all tasks of project FDM that are still in progress since 1994. These tasks are assigned to SIMON and BORGER.

**Example output**

Overview 'Tasks to handle by ...'

940101 - 950101 --- Page 1

Selection:

Project = FDM  
 Period from = 940101  
 Period to = 950101

Problem	To handle by	Delivery Type	Priority	Reported by	Status
FDM0116	SIMON	ERROR	HIGH	BBL	ENTERED
Description: BBL ref.: BBL-001438 Referential integrity mechanism aborts if a unique key is used in a set of another database.					

Overview 'Tasks to handle by ...'

940101 - 950101 --- Page 2

Problem	To handle by	Delivery Type	Priority	Reported by	Status
FDM0057	BORGER	930405	HIGH	BBL	ENTERED
Description: By ADD or UPDATE of a SLAVE-RECORD the MASTER-RECORD could be deleted between the moment of the 'FIND' of the MASTER-RECORD and the STORE of the SLAVE-RECORD.					
FDM0060	BORGER	930421	MEDIUM	WUH	ENTERED
Description: WUH Ref.: 188 Coding "cascaded delete" is false. By deleting an entity, all ERS-relations to that entity are deleted, even the entities that have a referential link.					

**Forms of 'Tasks to Handle by ...'**

A task form will be created for each selected task. These forms are sorted in the order of task number.

**37.13.3. (Task) Overview 'Tasks solved during a period'**

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Overview of tasks**.
4. Click **List option = 'Tasks solved during period.'**

This overview shows tasks that have now status solved, and that were solved during the given period. This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.

Status solved is indicated by relation `<task>:STATUS(SOLVED)`. The solution date is indicated by relation `<task>:READY(<date>)`. Tasks are selected if `<period from> LEQ <task date ready> LEQ <period to>`.

Extra selections (for example a project, announced by) can be made on a continuation screen, where the period can also be entered.

Each task that meets all the selection criteria, but does not have a "SOLVED-IN-RELEASE" indication, inherits the current release number from its project.

The overview is sorted by task number.

The example output gives all tasks of project DCIS that were solved in 1994.

### Output Example

Overview of tasks that were solved during period 940101 - 950101 --- Page 1

Selection:

Project = DCIS  
Period from = 940101  
Period to = 950101

Problem	Resolved by	Ready Type	Release Reported by	Status
DCIS0018	SIMON	940204 ERROR	40 SOFTNET	
Description: When we attempted to use the file transfer software, the following files were missing: OBJECT/S/NP/SER/NPSE0001 OBJECT/SER/PSYDC01				
DCIS0021	GERARD	940813 FEATURE	40 R&D	
Description: Fix priority handler process in FTP generator. At this moment the priority of the TPP is used (default 70)				

COUNTS FOR PERIOD 940101 - 950101

CUSTOMER:	<-ERRORS->			<-FEATURES->		
	REPORTED:	SOLVED:	ACTIVE:	REPORTED:	SOLVED:	ACTIVE:
SOFTNET		1				
R&D					1	
---	---	---	---	---	---	---
		1			1	+
ALL		2				

### Forms of tasks that were solved during a period

This overview creates a form of each task that has now status solved, and that was solved during a given period.

### Overview of solved tasks with documentation impact

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Overview of tasks**.
4. Click **List option = "Solved tasks with doc.impact."**

This overview selects the same tasks as the solved overview, but there is one extra selection: only solved tasks with documentation impact are selected. This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.



Documentation impact can be entered for each task on the task define screen ("Documentation impact <update>" button).

### Forms of tasks with documentation impact

The same selection criteria are used as the overview of tasks with documentation impact. The only difference is that each task is printed on a separate form.

## 37.13.4. (Task) Overview "Tasks active on a date"

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Overview of tasks**.
4. Click **List option = "Tasks active on a date."**

This overview gives all tasks that were active on a given date. If no date is entered, then today's date is used. This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.

A task is active on a date if the RECEIVED timestamp is before that date, and the READY timestamp after that date(<task date received> LEQ <period to> AND <task date ready> GEQ <period to>).

The overview is sorted by task number.

The output example gives an overview of all features of project FDM that were active on 1 January 1995.

### Output Example

Overview of tasks that were active on a date

950101 --- Page 1

Selection:

Project = FDM  
 Problem-type = FEATURE  
 Period to = 950101

Task	Reported by	Delivery Type	Priority	To handle by	Status
FDM0087	ID-EAS	931210 FEATURE	HIGH	RIS34	ENTERED

Description: SNS-EAS Ref.: EA0045

After store rules. A

after store rule is only executed after an update transaction. However, there are situations that I want to do some things after an inquire or browse action. So there should be also a "after inquire rule".

FDM0113	BBL	FEATURE	MEDIUM	RIS34	ENTERED
---------	-----	---------	--------	-------	---------

Description: BBL ref.: BBL-001300

In the current situation the process items are not filled after "total format add".

FDM0117	BAC	941031 FEATURE	HIGH		SOLVED
---------	-----	----------------	------	--	--------

Description: Various suggestions to improve the performance of the generated DIM's

1-5: Don't generate empty or unnecessary sections.

6 :Create a possibility to define the standard transactions that are not valid in a DIM.

COUNTS FOR DATE 950101

CUSTOMER:	<-ERRORS->			<-FEATURES->		
	REPORTED:	SOLVED:	ACTIVE:	REPORTED:	SOLVED:	ACTIVE
BAC						1
BBL						1
ID-EAS						1
+	---	---	---	---	---	3
ALL			3			
-----						

### Forms of tasks that were active on a date

The same selections are possible as at the overview of active tasks, but in this case, each selected task is printed on a separate form.

### 37.13.5. (Task) Overview “Available tasks”

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Overview of tasks**.
4. Click **List option = “Tasks available on a date.”**

This overview selects by default all tasks that are available in the repository. This means that active and solved tasks are reported. Extra selection criteria can be entered on a continuation screen to limit the output. If a period is entered then tasks are selected if <period from> LEQ <task date received> LEQ <period to>. This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.

The overview is sorted by task number.

### 37.13.6. (Task) Overview “Tasks transferred from an Env”

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Transferred tasks**.
4. Click **Type Reports = “Tasks transferred from...”**

This overview shows all tasks that were transferred from an environment during a period. The “environment-from” can be entered on a continuation screen. Tasks are selected if <period from> LEQ <timestamp task substatus OK> LEQ <period to>. This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.

The tasks that are given on this overview are transferred to the next environment, and have to be tested on this next environment. This overview should be run at the end of each work day in a batch job.

### 37.13.7. (Task) Overview “Task dump status for each environment”

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Dumped tasks**.
4. Click **List option = “Tasks solved during period.”**

This overview shows tasks with a specific dump status. The following values for dump status are possible:

NONE	Tasks that are not yet dumped.
ACTIVE	Tasks that are dumped for test (not final).
DUMPED	Tasks that are dumped finally.
LOADED	Tasks that are loaded (in a target repository)

If a period is entered then a task is selected if:

<period from> LEQ <timestamp dump status> LEQ <period to>.

This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.

### 37.13.8. (Task) Summary “Reported, solved and still active tasks”

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Overview of tasks**.
4. Click **List option = “Summary...”**

Tasks are divided into two main groups: errors and features. A task is an error if it has a task type with kind = error. In all other cases, the task is a feature. (Refer to “37.4 “(Task) Types” for information about the “kind”). This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.

The task summary gives the following information per customer or department:

- Amount of errors reported by <customer or department> during <period>.
- Amount of errors solved for <customer or department> during <period>.
- Amount of active errors of <customer or department> at the end of <period>.
- Amount of features reported by <customer or department> during <period>.
- Amount of features solved for <customer or department> during <period>.
- Amount of active features of <customer or department> at the end of <period>.

Totals for each category are given at the bottom of the overview, and the errors and features are there cumulated too.

The output example gives a summary of reported and solved features during period "October 1995."

### Output Example

COUNTS FOR PERIOD 951001 - 961031

CUSTOMER:	<-ERRORS->			<-FEATURES->		
	ANNOUNCED:	SOLVED:	ACTIVE:	ANNOUNCED:	SOLVED:	ACTIVE:
BAC				1		2
BBL				12	16	30
DIBA				1	1	
ID-EAS						13
ID-FACTO				3	4	
PHOENIX				1	1	1
R&D				53	57	15
SNS						4
WUH				11	5	21
	----	----	----	----	----	----
				82	84	86
						+
ALL	82	84	86			

### 37.13.9. (Task) Overview “Overtime tasks”

1. Click the **Toolbar** function.
2. Click **Reports**.
3. Click **Overview of tasks**.
4. Click **List option = “Overtime tasks.”**

This overview shows all tasks that had to be delivered on a specific date and were still active on that date. This overview is created by the RESPECT/TASK/LIST program that runs on the mainframe.

Tasks without a delivery date are not selected for this list.

A task is overtime if the delivery date is before the selection date.

### 37.13.10. (Task) Overview “History of a file”

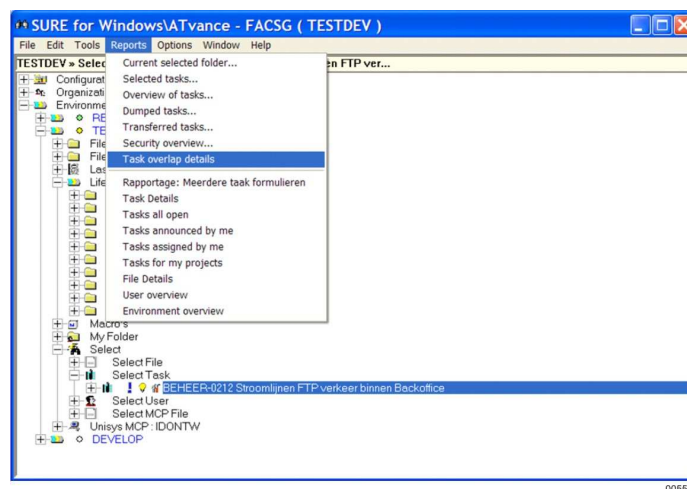
This overview shows a file and all tasks with the reasons why the source was modified in the past. The printed output is similar to the online “HIST” of a file.

### 37.13.11. (Task) Overview of task overlaps

SURE offers an overview that details task overlaps of a specific task with other tasks.

To create the overview, Perform the following steps:

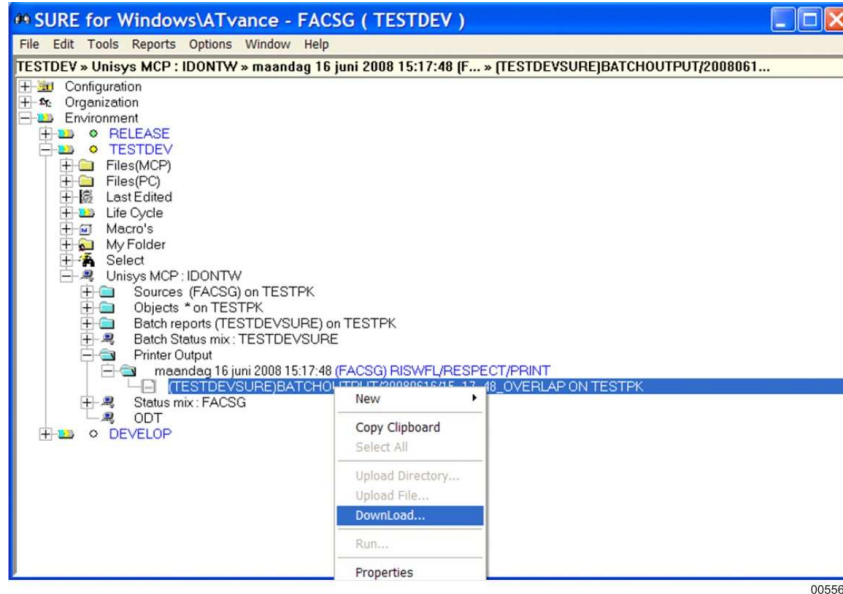
1. Right-click a task in SURE for Windows to make it current.
2. Click the **Toolbar** function.
3. Click **Reports**.
4. Click **Task overlap details**.
5. Click **OK**.



## Task

The report is created on the mainframe. To view the report in MS-Word, perform the following steps:

1. Expand the Unisys MCP folder.
2. Expand the Printer output folder.
3. Expand the <date-time>... RISWFL/RESPECT/PRINT sub-folder.
4. Right-click the **backup file** and select **download to open it in Word option**.



The following is an example overview with the overlaps of task BEHEER-0212.

```
Task overlap report                                     20080616 14:29:14   Page: 1

BEHEER-0212  TESTDEV  TESTGROEP

BEHEER-0212  Reported by  JAN
              Assigned to  TESTGROEP
              Performed by JAN
              Stroomlijnen FTP verkeer binnen Backoffice
TESTDEV      BA\SCRIPT\FB_MAAK_RELBEHEER_LN_REND.WSF      overlap with BEHEER-0186
TESTDEV      BA\SCRIPT\KLNT_ASSIST_INTERFACE.WSF          overlap with BEHEER-0231

BEHEER-0186  Reported by  JAN
              Assigned to  TESTGROEP
              Performed by JAN
              FB_MAAK_RELBEH_REND waitok aanhet einde van de WSF verwijderen
TESTDEV      BA\SCRIPT\FB_MAAK_RELBEHEER_LN_REND.WSF      overlap with BEHEER-0212

BEHEER-0231  Reported by  FACSP
              Assigned to  TESTGROEP
              Performed by FACSP
              Aanpassen van een tweetal script file m.b.t. de naamgeving van FILENAMENETWERK.
TESTDEV      BA\SCRIPT\KLNT_ASSIST_INTERFACE.WSF          overlap with BEHEER-0212
TESTDEV      BA\SCRIPT\KLNT_ASSIST_INTERFACE.WSF          overlap with BEHEER-0212
TESTDEV      BA\SCRIPT\TSP_ASSIST_INTERFACE.WSF           overlap with BEHEER-0212
TESTDEV      BA\SCRIPT\TSP_ASSIST_INTERFACE.WSF           overlap with BEHEER-0212
```

### 37.13.12. (Task) RESPECT/TASK/LIST

The railroad diagram of the RESPECT/TASK/LIST program is as follows:

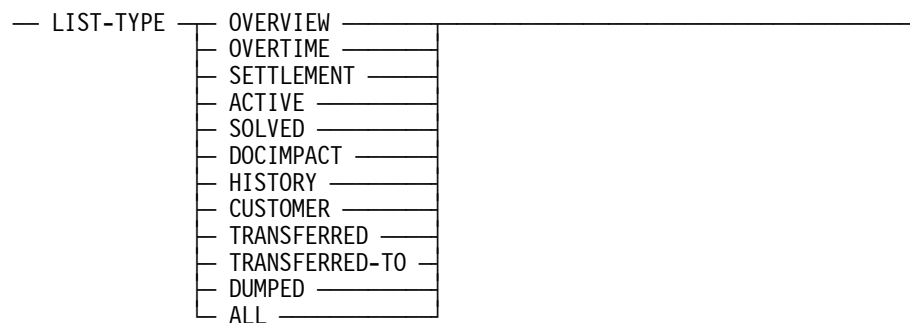
RUN RESPECT/TASK/LIST("<Input parameter>");TASKSTRING="<Environment>"

<Environment> = A repository environment

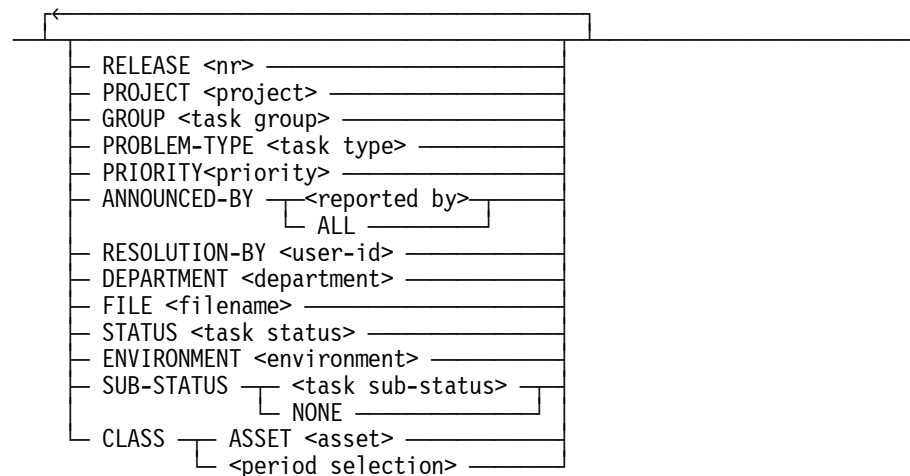
<Input parameter> =

—<overview-selection>—<attribute selection>—<period selection>— /  
/ —<site attributes>—<content selection>—<run options>—

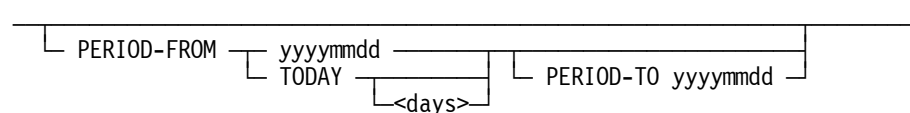
<Overview selection> =



<Attribute selection> =



<period selection>=







TRANSFERRED	This overview shows tasks that are transferred from an environment. MODE = FORMS creates a form per task.
TRANSFERRED-TO	This overview shows tasks that are transferred to an environment. MODE = FORMS creates a form per task.
DUMPED	This overview shows the tasks that are dumped from an environment.  Required parameters: ENVIRONMENT, SUB_STATUS Possible values SUB-STATUS: ACTIVE, DUMPED, LOADED, NONE  Where SUB-STATUS = NONE , selects the opposite tasks, so the tasks that are not dumped from the environment.
ALL	This overview prints all kinds of tasks.
RELEASE	Select only tasks with this release or solved in this release.
PROJECT	Select only tasks with this project.
PROBLEM-TYPE	Select only tasks with this task type.
PRIORITY	Select only tasks with this priority.
ANNOUNCED-BY	Select only tasks that are reported by this customer or user.
RESOLUTION-BY	Select only tasks that are resolved by this user.
DEPARTMENT	Select only tasks that are reported by this department.
FILE	This parameter is only applicable for list-type = HISTORY. It identifies the file name wherefore the history must be listed.
STATUS	Select only tasks with this status.
SUB-STATUS	Select only tasks with this sub-status.
CLASS + ASSET	Select tasks through a variable class/asset.
PERIOD-FROM	The start of the period.
PERIOD-TO	The ending of the period. Default = today.
ATTRIBUTES	This defines a site-specific attribute that must appear on the list.
ALSO-STATISTICS	This shows the task statistics. How many tasks were reported, solved and active in a week? What was the average time to solve a task?
ALSO-DESCRIPTION	This shows for each task the full description.
ALSO-SOLUTION	This shows for each task the full solution.
ALSO-DOCUMENTATION	This shows for each task the full documentation impact.
ALSO-HISTORY	This shows for each task the full task-history.
ALSO-LOG	This shows for each task the log (only forms-mode).
MODE	MODE = FORMS enables form-mode.

BATCHOUTPUT	This puts the overview in directory BATCHOUTPUT.
PRINTER	This sends the overview to the printer.

### 37.13.12.1. How to customize task overviews

Task reports are created by the RESPECT/TASK/LIST program. These overviews can be created in two modes:

1. Task form mode - A task form is printed for each selected task.
2. Overview mode - A list of selected tasks is created.

The following task selections can be made:

- Tasks that are active (during a given period.)
- Tasks that are solved (on a given date.)
- Tasks that are transferred during a period from an environment
- Tasks that are transferred during a period to an environment
- Customer overview: tasks that are announced and solved during a period and still active at the end of that period.

The above mentioned task selections are primarily on task-entry-date and on task-solved-date, and these selections can be fine tuned with extra specific selection criteria. For example, make an overview of solved task, but only the tasks that belong to system XYZ.

You can customize the content of the overview with the following parameters:

#### Task attributes selection

```

— ATTRIBUTES —<attribute> — / —
— / — ALSO-DESCRIPTION — / —
— / — ALSO-SOLUTION — ALSO-DOCUMENTATION — ALSO_HISTORY —

```

- This parameter is ignored in forms mode.
- The selected attributes are printed on the same lines as the task-name.
- The order of the task attributes in the parameter identifies the print order on the overview.
- If ALSO-DESCRIPTION option is NOT used, then the first part of the first line of the task description is printed on the same line as the task-name, just behind the last task attribute.
- If ALSO-DESCRIPTION option is used, then the full task description is printed on the next lines of the task overview.
- If ALSO-SOLUTION option is used, then the full task solution is printed on the next lines of the task overview.

- If ALSO-DOCUMENTATION option is used, then the full documentation impact is printed on the next lines of the task overview.
- If ALSO-HISTORY option is used, then the full task history is printed on the next lines of the task overview.

The task history consists of the following columns:

- The file name that was modified because of the task.
- The solved date/time (when the task got status solved and the file name was placed in the task history).
- The user the transferred the task when it got status solved.
- The version number of the file at that moment.
- The name of the SURE environment where the task got status solved.
- The date/time of the last compilation of the file in that environment.
- The date/time of the last deployment of the file in that environment.

### Task log selection in forms mode

— MODE = FORMS ————  
                                   └─ ALSO-LOG ─┘

The ALSO-LOG option is only applicable in forms mode. It prints the full task log at the end of each task-form.

### Period selection

[— PERIOD-FROM ————  
                                   └─ yyyymmdd ————┘  
                                   └─ TODAY ————┘  
   └─ <days> ─┘ ]    └─ PERIOD-TO-yyymmdd ─┘

- If the PERIOD-TO option is not used, then today is used.
- The TODAY - <days> option can be used to if the program is automatically queued for the next week using a job. For example the following job:

```
BEGIN JOB WFL/TASK/LIST;
START WFL/TASK/LIST; STARTTIME = 22:00 ON + 7;
RUN OBJECT/RESPECT/TASK/LIST
("LIST-TYPE=TRANSFERRED-TO, ENVIRONMENT=PROD, PERIOD-FROM=TODAY - 7");
END JOB.
```

In this example, the job is automatically scheduled for next week, and then it selects all tasks that are transferred to PROD during that week.

A number of additional reports are created that provides word or printer output. These reports may be used as templates for user defined reports.


To view these reports, click the **Toolbar** function and click **Reports**. The following menu is shown.



### Example: RTF File



## Example Output

 09/09/04 1:36 PM All tasks in my (SUREDEMO) projects

**SUREDEMO**

<i>Task</i>	<i>Status</i>	<i>Reported by</i>	<i>Project</i>	<i>Work done by</i>
<b>PRODUCT0021</b>	DEVELOP	SUREDEMO	PRODUCT	SUREDEMO
	Test quarterly release			
	Dependent of PRODUCT0024 4th quarter 2004			
	DEVELOP	PATCH/PROG/SRC/X/PRODUCT0021		
	DEVELOP	PROG/SRC/X		
<b>PRODUCT0022</b>	DEVELOP	SUREDEMO	PRODUCT	SUREDEMO
	Test yearly release			
	Dependent of PRODUCT0025 2005			
	DEVELOP	PATCH/PROG/SRC/X/PRODUCT0022		
	DEVELOP	PROG/SRC/X		
<b>PRODUCT0024</b>	ENTERED	SUREDEMO	PRODUCT	
	4th quarter 2004			
	Controls	PRODUCT0021	Test quarterly release	
	DEVELOP	PRODUCT0021		
<b>PRODUCT0025</b>	ENTERED	SUREDEMO	PRODUCT	
	2005			
	Controls	PRODUCT0022	Test yearly release	
	DEVELOP	PRODUCT0022		
<b>PRODUCT0026</b>	PRODUCTION	SUREDEMO	PRODUCT	SUREDEMO
	Fix on sunday evening			
	PRODUCTION	PATCH/PROG/SRC/X/PRODUCT0026		
	PRODUCTION	PROG/SRC/X		
<b>REFERENTIAL0004</b>	DEVELOP	SUREDEMO	REFERENTIAL	
	Simon test			

005564

### 37.13.14. Multiple Task Forms Using RTF

One of the available report functions is "Multiple Task Forms." This function creates a Word document with multiple task forms: each selected task results in a separate page with task information. The advantage of this report is that all task-forms are combined in one Word document.

The function is implemented using rtf-file ..\Ris\Rtf\MultiTask.rtf

MultiTask.rtf references a macro with technical name MULTI-TASK. This macro must first be added before it is possible to use the MultiTask document. That is why the MultiTask document is not automatically added to the Reports menu.

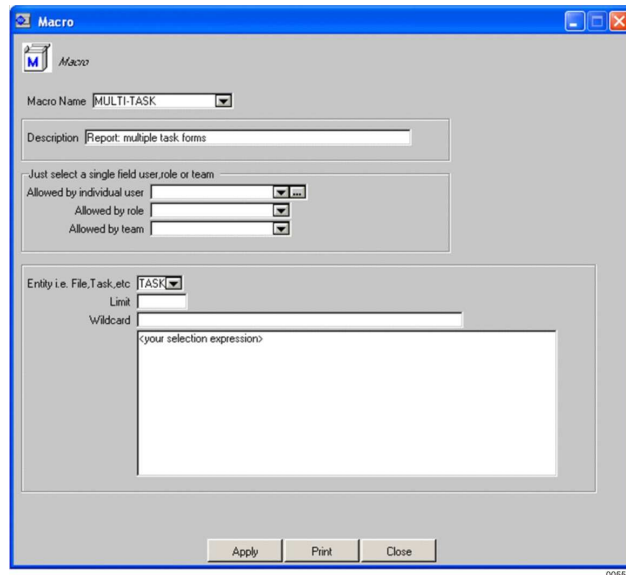
Macro MULTI-TASK selects the tasks that must be reported. The user is free to modify the selection expression of this macro as per the need. Rtf-file MultiTask.rtf creates a task form for each selected task. The task-description, task-solution and task-documentation-impact are printed.

### Configuration

Document MultiTask requires the following configuration:

1. Macro MULTI-TASK:

Add a new macro with macro-name = MULTI-TASK and entity = TASK.



Select the applicable task with your own selection expression.

2. Make document MultiTask.rtf available in the Reports menu.

Open INI-file ...\\AW\_OBJ.INI

Add the following line in paragraph [SUREREPORTS]

```
<installation directory>\\Ris\\Rft\\MultiTask.rtf = Multiple tasks forms
```

Example of a paragraph [SUREREPORTS]:

```
[SUREREPORT]
C:\\SURE\\REND\\RIS\\RTF\\MultiTask.rtf=Multiple task forms
```

### 37.13.15. Reporting Using Microsoft Excel

There are various ways of reporting using SURE:

- Do a macro or query, drag it to a right window, select it, copy it to the clipboard and paste it in a tool such as Microsoft Word or Excel.
- Create a word template file in RTF format and add it to the SUREREPORT section in the INI file.
- Use the available standard reports.

This feature adds another mechanism; it uses Excel to drive the SURE software using a MACRO. Using Excel as a tool allows the user to create customized reports using a generic tool.

### Example

The following Excel sheet retrieves task data from SURE and formats it.

The screenshot shows an Excel spreadsheet titled "Microsoft Excel - TaskRep.xls". The spreadsheet is divided into sections for defining report parameters and displaying task data.

Argument	Value	Field name
Task type	DEVELOP	TaskType
Environment	Environment	Environment

Macro text: PROBLEM-TYPE(<<TaskType>>) AND STATUS(<<Environment>>)

Task Name	Short description	Status	Assigned	Entered	Priority	Delivery
12. BA-P-000076	TEST	DEVELOP	SIMONDEV	20050420		
13. BA-P-00073	Release 52.0	DEVELOP	SGROOT	20050419	A	
14. BA-P-00074	GG	DEVELOP	SGROOT	20050419		
15. BAS00148	short	DEVELOP	FRANK	20050425	HIGH	
16. BB-PROBLEM-00	more patches	DEVELOP	SGROOT	20030414	A	
17. MFC-P-00001	MFC SCC test	DEVELOP	MFC1	20040709		
18. SCCTEST0001	SCCTEST#	DEVELOP	FRANK	20030328	A	
19. SIEMT001	A	DEVELOP	SGROOT	20050421		

The "Examples" section of the SURE internet site contains a zip file with the basic spreadsheet, with documentation, in "Reporting Using Excel." The current SURE software supports all the calls used in the example.

The xls file may be incorporated in the SURE Explorer by adding it to the HELP menu using the following INI file entry:

```
[SUREHELP]
E:\Sure\Support\TaskReport.xls=Excel Task Reports.
```

## 37.14. (Task) Overview of ID that has to be Tested

This paragraph is for RIS-users only.

This overview shows a list of IDs that have to be tested, caused by changes made for a task or a delivered dump-file.

The parameter of the RIS/REPORT/IMPACT batch program determines if the program runs in task-mode or in dump-file-mode.

### 37.14.1. Task-Mode

If the parameter of the RIS/REPORT/IMPACT program is a <task>, then the program runs in task-mode.

The program will read the history of the task and checks all files linked to that task. The impact is checked for each of those files

### 37.14.2. Dump-file-mode

If the parameter of the RIS/REPORT/IMPACT program is empty or a <dump-id>, then the program runs in dump-file-mode.

After the dump-file is loaded in the repository and all the necessary id's are generated, the overview shows a list of id's that have to be tested, caused by changes made by loading the dump-file.

When a task is added into the deliver queue, the task gets a dump-id to determine for which delivery this task is.

This dump-id is defined by the name-standard DELIVERY and looks like:

```
<PARAMETER>'/'{'DELIVERY',4}
```

The parameter is RESPECT/DUMP/TRANSFER.

An example of a dump-id is: RESPECT/DUMP/TRANSFER/0001.

After a final delivery, the number will be increased by one.

When the dump file is loaded into the destination environment, the batch load program adds relation RIS:TO-REPORT(<dump-id>) for every dump-id that is found in the dump file. It is possible to have more than one dump-id per dump file.

The batch program checks if there are TO-REPORT(<dump-id>) relations, and makes an overview per dump-id for all the tasks that have a relation TEST <dump-id>.

At the end, the TO-REPORT(<dump-id>) relation is removed by the batch program.

The impact overview gives a list of all id's that have to be tested for a dump-id.

It is only useful to run the batch program after the to-generate queue is processed; otherwise, if some id's are not generated, an impact overview with wrong information will be printed.

If the RIS/REPORT/IMPACT program is started with an empty parameter, an overview is made for all the TO-REPORT(<dump-id>) relations of the environment for which the program runs.

If the program is started with a dump-id as parameter, an overview is made for that dump-id.



**Example**

Task T1 is added into the deliver queue. The current dump-id number is 8.

T1 will get the following relation.

**Entityowner:class(asset)**

```
Problem|T1:Test(Respect/Dump/Transfer/0008)
```

After task T1 has been dumped and loaded into the destination environment the load program has added the following relation.

**Entityowner:class(asset)**

```
Option|Ris:To-Report(Respect/Dump/Transfer/0008)
```

After generating all the necessary id's the batch program will make an impact overview for the Respect/Dump/Transfer/0008 dump-id.

```
--- Overview of Id's that have to be tested for: RESPECT/DUMP/TRANSFER/0008
```

```
ONLINE-TRN(s)
```

```
In LFI: PINWC03
  AM1211      COPY AUTHORIZATION-BITS
  AM1212      COPY EMP-TERM-FUNC AUTHORIZATION MODEL
  AM1213      Copy authorization bit records global
  AU1202      add-or-update authorization record
  AU2101      inquire authorization record
  AU2103      inquire to change Authorization Details
  AU2104      Delete Authorization Details
  AU2111      browse inquire specific
```

```
In LFI: PINWC04
  ES1101      add entity-vice record
```

```
In LFI: PINWC05
  TS1101      add table-vice record
```

```
SEL(s)
  DTER/02
```

```
SRT(s)
  NPSRT0001
```

## 37.15. (Task) Options

Options for the Task module are available in the “Global Options” dialog.

1. Click the Environment folder.
2. Click **Global Options**.

### **Name standard for newly entered tasks (“task” tab sheet)**

This field can contain a formula that determines the names of newly entered tasks. The default formula is <PARAMETER>{<PROJECT>'PROBLEM',4}. <PARAMETER> is in this case the project. With this formula the task name consists of the project name followed by a four digit sequence number.

Authorized users can override this formula to choose their own task identifier, but only if no naming standards are mandatory for the project.

### **Example**

If a new task is added for project RIS, then the automatically defined task number is in the range RIS0001 - RIS9999.

### **Define tasks per task group (“task” tab sheet)**

This option makes it possible to define multiple task groups per project. If a new task is added then only the task-group field has to be entered. The project and task-type will be inherited from the task-group. The default name-standard formula for this method is <task group>-'<four digit number>. This option can be handy to sub divide projects into smaller task groups, if only a few, very large, projects are defined in the system.

To define Task-groups:

1. Click the Configuration folder.
2. Click **Task Group**.

### **Example**

Consider task-group COMPILE with project SURE and task type FEATURE.

If a new task is added for this task group, then the name of this task will be: COMPILE-0001, the project of this task will be SURE, the task type will be feature, and the task group is COMPILE.

### **Approve newly entered tasks (“task” tab sheet)**

If this option is set, a task has to be approved or denied by an authorized user before the task can be made current. The task is approved by entering the “to handle by” field. The task is denied through DENY command. If this option is not set, all users with authorization ADD or UPD are allowed to enter the “to handle by” field.

**Task verification at each update (“task” tab sheet)**

The terminal emulation interface gives only once per RIS/MENU-session a task-verification. This option enforces that the current task has to be verified at each update.

**Archive solved tasks (“history” tab sheet)**

It is possible to identify the maximum amount of days that a solved task has to be kept in the repository.

If this field is empty, then the solved task will always be kept.

If this field contains a number (0 through 999), then the task will be deleted from the repository after the indicated amount of days that the task got status solved.

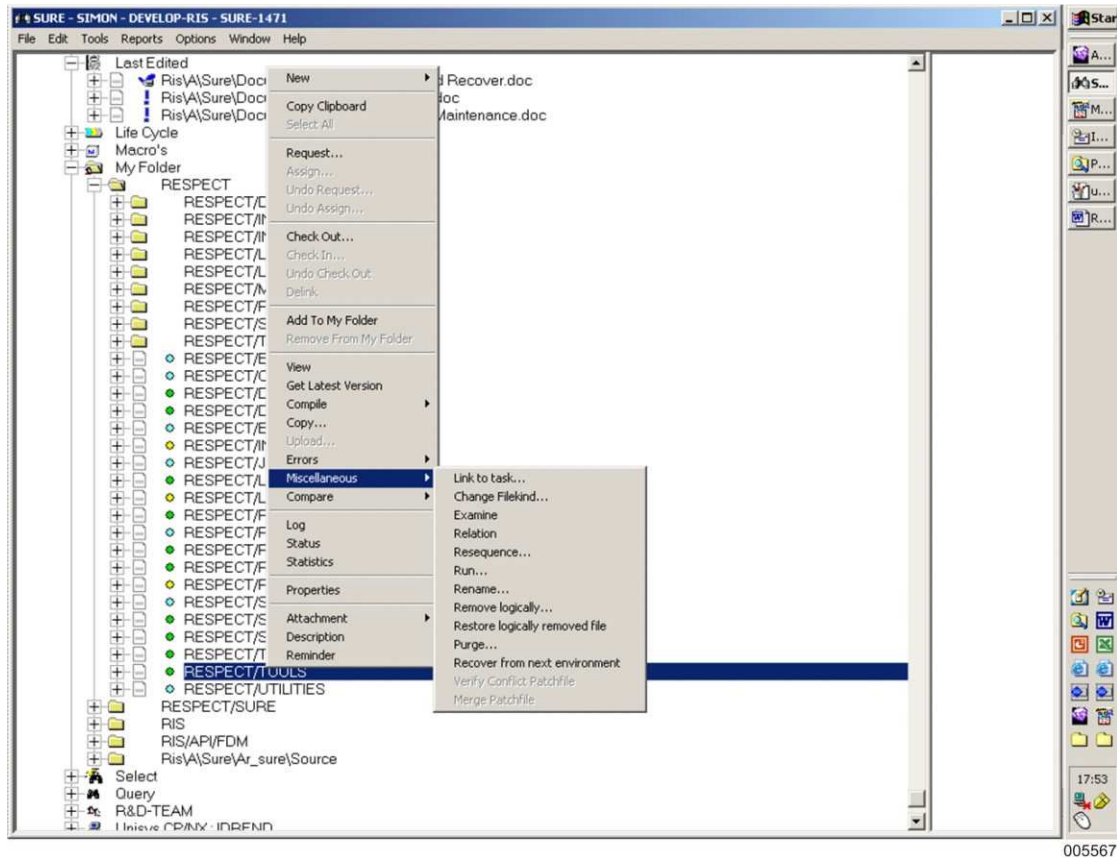
## Task

---

## Section 38

# Remove, Recover and Rollback

It is possible to delete a file physically or logically and to undo that action.



The speed menu of a file contains the following functions:

- Remove logically
- Restore logically removed id
- Purge
- Recover from the next environment

The speed menu of a delta file contains "Rollback" action.

### 38.1.Remove Logically



The “Logical Delete” function can be used to remove a file logically from an environment. A file can only be removed in its highest or lowest environment, unless “Delete in all environments” field is checked, because then the file will be removed from all environments.

A warning is given if the file is still in use as a copy file, or if it is still referenced in a WFL job. It is possible to ignore the warning. In that case the source and attributes are removed, but the file name itself remains in SURE because of the references.

If a file is removed, and this file has a start-job that is not used anywhere else, then this start-job is removed too.

For RIS-users only:

- If a RIS-id is removed, then the corresponding generated source is automatically removed too.
- If a format is removed, then the corresponding copy file is also removed.
- If an FCM-id or RPT-id is removed, then the corresponding script-file is also removed.
- If a DIM-id or FCM-id is removed, then the batch-transaction definitions that are processed by that DIM or FCM are removed too.

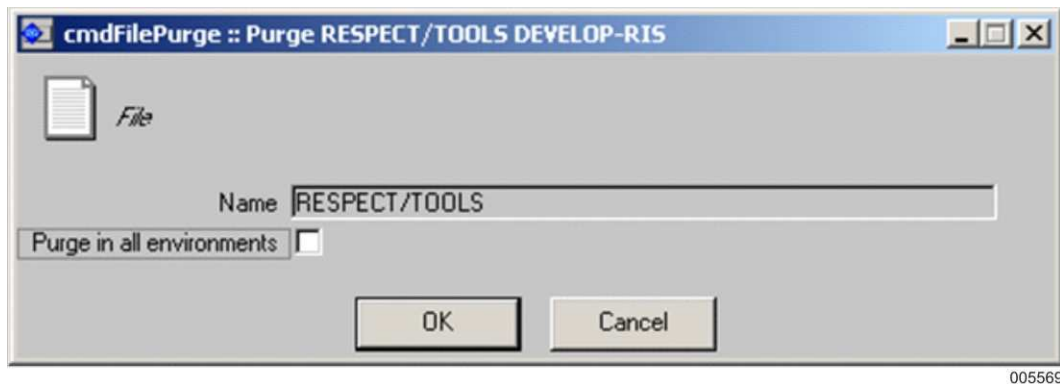
### 38.2.Restore Logically Removed id

The “Restore logically removed id” function can be used to reactivate a logically removed file or RIS-id.

For RIS-users only:

- If a RIS-id is activated, then the corresponding generated source is automatically activated too, and the RIS-entity itself is placed in the generation queue.
- If an FCM-id or RPT-id is activated, then the corresponding script-file is also activated.
- If a DIM-id or FCM-id is activated, then the batch-transaction definitions that are processed by that DIM or FCM are activated too.

### 38.3. Purge



The “Purge” function deletes a file or RIS-id physically from an environment.

A warning is given if the file is still in use as a copy file, or if it is still referenced in a WFL job. It is possible to ignore the warning. In that case the source and attributes are removed, but the file name itself remains in SURE because of the references.

If “Purge in all environments” option is checked, and the source is not referenced as a copy file or by a WFL or anywhere else, then the file name will be removed too.

If a file is removed, and this file has a start-job that is not used anywhere else, then this start-job is purged too.

A user is always allowed to purge a source if that source is empty and added by him (if the source-name was added by mistake).

For RIS-users only:

- If a RIS-id is purged, then the corresponding generated source is automatically purged too.
- If a format is purged, then the corresponding copy file is also purged.
- If an FCM-id or RPT-id is purged, then the corresponding script-file is also purged.
- If a DIM-id or FCM-id is purged, then the batch-transaction definitions that are processed by that DIM or FCM are purged too.

### 38.4. Deleted PC Files are Removed on the Build Server

PC files that are deleted or renamed in SURE are also deleted from the build server. The build process reads in its initialization a deleted-or-renamed queue, and removes all the sources from the build directories that are mentioned in that queue.

SURE keeps a deleted-or-renamed queue for each build-server on each environment.

A source is added to the deleted-or-renamed queue of an environment at the moment that it is deleted or renamed in that environment. That is

- With “Remove logically” functions, Purge, or Rename for a specific environment (when “for all environments” option is disabled).
- With a transfer of a task that contains the “Logical delete” or “Rename function.”
- With “Remove logically” function, Purge, or Rename for all environments, when “for all environments” option is enabled.

The build process performs the following actions:

- It reads the deleted-or-renamed queue of the environment wherefore the build is started.
- For each source in that queue:
  - Delete the source from the build directories on the build server.
  - Remove the source from the deleted-or-renamed queue.

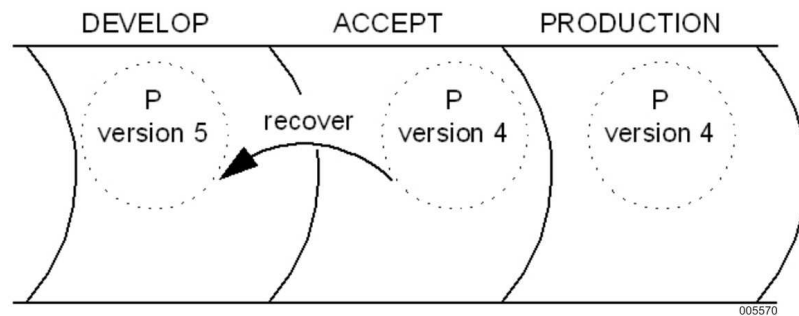
In case of a rename the old file name is removed from the build server. The source is placed with its new name on the build server during the next phase of the build process.

### 38.5. Recover From the Next Environment

It may be necessary that a file or RIS-id, which is adapted wrongly, has to be recovered from a previous version. This can also be the case when a file or RIS-id is purged from a specific environment.

The “Undo all Changes” function can be used to recover a file or RIS-id from a next higher environment to the current environment. The version of the file or RIS-id that is available in the next higher environment will be copied to the current environment.



**Example**

In this picture version 4 of P is recovered from ACCEPT environment to DEVELOP environment. Version 5 of P (which was available in DEVELOP before the recover command) is overwritten by version 4.

Notice that a file can also be recovered to a previous state using the match files of that source. To recover a file to its previous version:

- Click **SURE command <delta-file>**.
- Click **Copy** to create a previous version of the file in CANDE.

It is not possible to recover a file if that file is quick fixed in the environment from where it has to be recovered. The reason is obvious; function recover has to re-install a previous version of the file in the current environment. If the file is quick fixed in the "recover-from" environment then that version was never available in the "recover-destination" environment, so function recover cannot re-install a previous version and is therefore invalid.

Notice that the recover function does not work for a task, but only for individual files and RIS-id's.

## 38.6. RollBack

This function makes it possible to back out the modifications to a source using the delta-files.

To back out the modifications to a source using delta-files:

1. Right-click the **delta file**.
2. Select **RollBack function**.

The modifications to the source of this delta file and all newer delta files will be rolled back. This action requires a valid task.

When a ClearPath server file is rolled back, then a new delta file is created which contains all the reversed actions of the rolled back delta files (see the example). This makes it possible to undo the rollback, just do extra rollbacks of the last delta file (in the example, a rollback of the last delta file make FileVersion 5.1 again current).

## **Remove, Recover and Rollback**

---

When a PC file is rolled back, then the selected delta file and all newer delta files are just removed and no new delta file is created, so a rollback of a PC file is irreversible. Rolling back a PC file requires an extra verification.

# Section 39

## Names Standards

### 39.1.Function

The name-standard option enforces SURE to determine new names according to a predefined naming standard formula. The Electronic Data Processing (EDP) department can define the syntax of these names.

The advantages of the name-standard option are:

- All names are in a form according to the company standards.
- The developer does not need to know these standards because new names are created for him by SURE.

The name-standards option can be set for each application system. This flexibility allows application systems to be developed with or without the option set. However, for new systems it is recommended that this option is set.

### 39.2.(Name-Standard) Implementation

#### 39.2.1. (Name-Standard) Definition

To define name-standard option for each application system:

1. Click **Configuration**.
2. Click **System**.
3. Click **Properties**.
4. Click **System** tab.

The Name Standards field is active.

If this option is enabled, and a new file is created for the system, then the SURE name-standard formula will determine the name of that new file.

### 39.2.2. (Name-Standard) Syntax

A name-standard formula is defined under a key. These keys are used in the software to obtain the correct naming standard formula for a specific RIS module. Some of these keys are fixed, while other naming standard keys are variable.

#### Example

Consider a naming standard formula with "SRT" key. The software in function "Define SRT" uses this naming standard when a new SRT module is created. SRT is a fixed naming standard key.

Consider a naming standard formula with "LIBRARY" key. This naming standard is not used by a standard RIS function. In this case, "LIBRARY" is the key of a naming standard formula that is used when a new file with LIBRARY file-type is entered. "LIBRARY" is a variable naming standard.

The naming standard routine is called with a key, a project, and a variable parameter.

The key is used to obtain the correct naming standard formula.

The project and parameter can be used in the naming standard formula. The project and parameter can be entered on the screen at the moment that a new name is created.

#### Example

Consider a naming standard formula with "LIBRARY" key.

The formula is as follows: 'S/'<PROJECT>/'<PARAMETER>.

In this case, the naming standard expects a project and a parameter. These two variables must be entered on the SURE screen when a new file with LIBRARY file-type is created. The project must be entered in the project-field and the variable parameter must be entered in the file name field.

If the project = "NP" and the parameter (in field file name) = "TEST123" then the new file name will be "S/NP/TEST123".

To define Name Standard formulas for file types in the SURE Explorer "file-type" dialog:

1. Click **Configuration**.
2. Click **FileType**.
3. Click **Properties**.

To define Name Standard formulas for tasks and task groups in the Global Options dialog:

1. Click **Environment**.
2. Click **Global Options**.

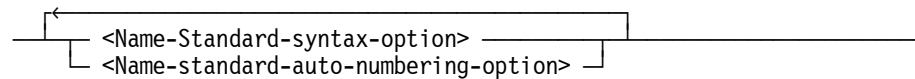
To define other name standard formulas for RIS modules and generated RIS sources in the terminal emulation interface:

1. Click **RIS/MENU** function.
2. Click **RIS Init.**
3. Click **Define Name Standards.**

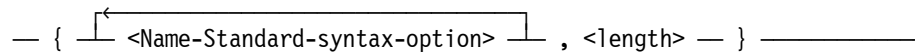
A continuation screen appears where you can:

- Add a new naming standard formula with a unique key.
- Inquire the formula of a naming standard.
- Update the formula of a naming standard.
- Delete a naming standard.
- Inquire an auto-numbering key.
- Update the current number of an auto-numbering key.

A Name-Standard is defined as a string in the form of:



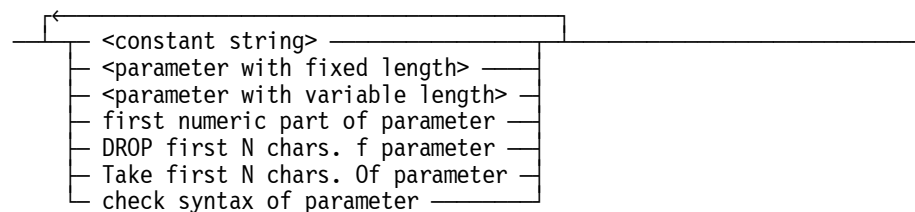
<Name-Standard-auto-numbering option>=

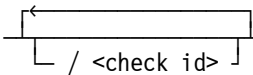


In this case, the <naming-standard-syntax-option> is used to create a unique auto-number key. Each time a new name is created, the current number of this auto-number key is raised by one. The newly created number is added to the new name. The <length> determines the length of the newly created number.

Notice that the auto-numbering key can be inquired on the screen and the current number can be updated.

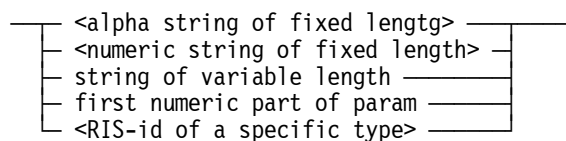
<Name-Standard-syntax-option>=



<constant string>	= An alpha numeric string between single quotes Example: "AAA/" In this case "AAA/" is added to the new name.
<parameter with fixed length>	= A string of "X" characters between parentheses. The amount of "X" character determines the length of the parameter. Example: (XXXX) In this case the parameter must have length 4.
<parameter with variable length>	= (?) or <PARAMETER> Example: (?) In this case the parameter can have any length > 0.
<first numeric part of a parameter>	= <SEQNO> Example: <SEQNO> using "AAA/123/BBB/456" parameter. In this case "123" is added to the new name.
<DROP first N chars. Of parameter>	= <DROP(<number>)> Example: <DROP(4)> using parameter = "ABCDEF" In this case "EF" is added to the new name.
<TAKE first N chars. Of parameter>	= <TAKE(<number>)> Example: <TAKE(4)> using parameter = "ABCDEF" In this case "ABCD" is added to the new name.
<check syntax of parameter>	= — ( — <check id>  ) —

The parameter must contain the same amount of slashes as this option in the naming standard formula.

<check-id> =



<fixed alpha string> = a string of characters "X" between square brackets. The length of the string is determined by the amount of "X" characters.

<fixed numeric string> = a string of nine characters. Between square brackets. The length of the string is determined by the amount of nine characters.

<variable string> = [?]

<first numeric part of param> = [SEQNO]

<RIS-id of a specific type> = [<type>]

	<p>Example:</p> <p>([XX]/[999]/[?]/[SEQNO]/[FORMAT]) will accept parameter 'AB/123/CDE/?/BAF0050' but will deny parameter 'abc/123/cde/?/BAF0050'.</p>
<naming standard key>	<p>= &lt;TYPE&gt;</p> <p>Example:</p> <p>&lt;TYPE&gt; with naming standard key 'LIBRARY'</p> <p>In this case "LIBRARY" is added to the new name.</p>
<project>	<p>= &lt;PROJECT&gt;</p> <p>Example:</p> <p>&lt;PROJECT&gt; with project 'NP'</p> <p>In this case, "NP" is added to the new name.</p>
<system>	<p>= &lt;SYSTEM&gt;</p> <p>Example:</p> <p>&lt;SYSTEM&gt; with project 'NP'.</p> <p>NP is a project of system "BASE."</p> <p>In this case, "BASE" is added to the new name.</p>
<filename>	<p>= Some mnemonics which extract a part of the file name</p> <p>&lt;FILE&gt;</p> <p>&lt;FILENAME&gt;</p> <p>&lt;EXT&gt;</p> <p>&lt;PATH&gt;</p> <p>&lt;FULLNAME&gt;</p> <p>Example:</p> <p>'PROJ\TEST\MYF.CPP'</p> <p>&lt;FILE&gt; = MYF</p> <p>&lt;EXT&gt; = CPP</p> <p>&lt;FILENAME&gt; = MYF.CPP</p> <p>&lt;PATH&gt; = PROJ\TEST\</p> <p>&lt;FULLNAME&gt; = PROJ\TEST\MYF.CPP</p>
<project attribute>	<p>= &lt;&lt;class&gt;&gt;</p> <p>The following relation is inquired:</p> <p>environment = 0</p> <p>RIS-entity = OPTION</p> <p>owner = &lt;project&gt;</p> <p>class = &lt;class&gt;</p> <p>The &lt;asset&gt; of the relation is added to the new name.</p> <p>Example:</p> <p>Consider project 'ABC' with the following relation:</p> <p>environment - entity - owner - class - asset</p> <p>0 - OPTION - ABC - SIGLE - 123</p> <p>In this case, '123' is added to the new name.</p>

The new name is the result of the concatenation of the name-standards-definitions.

### Examples

All examples are done with projects NP and BA. Both projects belong to system BASE

Key	Project	Parameter	Formula	New name	Auto numbering key
SRT	NP	-	<PROJECT>'SRT'{'<PROJECT>SRT,4}	NPSRT0001	NPSRT
SRT	BA	-	<PROJECT>'SRT'{'<PROJECT>SRT,4}	BASRT0001	BASRT
WFL	NP	DB	<TYPE>'/'{'<PROJECT>'/'(?) {WFL,2}	WFL/NP/DB01	WFL
SEL	NP	DTER	<DATASET>'/'{'<DATASET>,2}	DTER/01	DTER
BATCH	NP	My-Name	'S/'<SYSTEM>'/'(?)	S/BASE/My-Name	-
TPE2	NP	ABC/NPDB	'S/'([XXX]/[DATABASE]){'TPE2',3}	S/ABC/NPDB/001	TPE2

### 39.2.3. (Name-Standard) Types

A name-standard formula is stored under a naming standard key. The software uses these keys to obtain the correct formulas. In RIS, a number of pre-defined keys are used. The EDP organization can create other keys for file name and format-name name-standards.

#### 39.2.3.1. Pre-Defined Name-Standards

A pre-defined name-standard has a fixed key, under which the name-standard syntax is stored. The EDP organization can change the syntax but it cannot change the name of the key.

Key	Used by	Reason
CNV	RIS	A naming standard with this key is used to create the name of a new CNV identification.
CNV-DATA	RIS(optional)	A naming standard with this key is used to create the name of a CNV data file.
COPY-FILE	RIS(optional)	The name of the generated copy file of a format is constructed according to this name-standard, if it is present. If this name-standard is in the form of <FORMAT>, the name of the copy file will be constructed from the format name. If this naming-standard is defined then naming-standard RECORD must be defined too.
COPY-PREFIX	RIS(optional)	If the COPY-FILE name standard is present, this name-standard may add a COPY-PREFIX to the file name. If the constructed file name does not start with the evaluated copy-prefix, the copy-prefix is added. An '*' before the copy-prefix is ignored in this evaluation. Example: format      BBS-CC-F001 (project BBS) copy-file   BBS/CC/F001



<b>Key</b>	<b>Used by</b>	<b>Reason</b>
DELIVERY	SURE(optional)	This formula determines the name of dump-groups. A dump-group is a group of tasks that are dumped and loaded simultaneously. The parameter passed to this formula = "RESPECT/DUMP/TRANSFER"
DODDLE	RIS	The name of the formats in DODDLE is checked against this name-standard. This validation is restricted to the size of the auto-number variable.
DODDLE-TO-LFI	RIS	The method to construct the name of an ONLINE-LFI-id from the name of a DODDLE file.
FCM	RIS	The formula to create the name of an FCM-id.
FCT	RIS(optional)	The formula to create the name of an FCT file name.
ID-RPT	RIS(optional)	This name-standard is required to construct a RPT-id from the name of an RPT script file.
LFI-BATCH	RIS	The formula to create the name of a BATCH-LFI-id.
LFI-ONLINE	RIS	The formula to create the name of an ONLINE-LFI-id.
LFI/ALGOLDEFINE	RIS(optional)	The method to construct the name of an ALGOL include file from the name of a COBOL copy file.
LFI-TO-DODDLE	RIS	The method to construct the name of a DODDLE file from the name of an ONLINE-LFI-id.
MODULE-FCM	RIS(optional)	The method to construct the name of an FCM-id from the name of a FDM source.
MODULE-RPT	RIS(optional)	The method to construct the name of an RPT-id from the name of a RPT source.
PROBLEM	SURE	The formula to create the name of a task.
PROBLEM-GROUP	SURE(optional)	The formula to create the name of a task-group.
RECORD	RIS	The formula to create the name of a copy file of a dataset-format.
RPT	RIS	The formula to create the name of an RPT-id.
SCRIPT-FCM	RIS(optional)	The method to construct an FCM-script file name from an FCM-id.
SCRIPT-RPT	RIS(optional)	The method to construct an RPT script file name from an RPT-id.
SEL	RIS	The formula to create the name of a SEL-id.
SELECTION-INFO	RIS	The formula to create the name of a SEL-FUNC.
SEL-DATA	RIS(optional)	The formula to create the name of a selection data output file. The parameter passed to this formula is = <sel-func>/<format>.

## Names Standards

---

Key	Used by	Reason
EL-DATA2	RIS(optional)	The formula to create the name of a selection data output file. If this formula is defined, then it overrules the formula of SEL-DATA. The parameter passed to this formula is = <sel-id>.
SRC-CNV	RIS(optional)	The method to construct a CNV source name from a CNV id.
SRC-DCIS	RIS(optional)	The method to construct a DCIS source name from a DCIS-id.
SRC-DRI	RIS(optional)	The method to construct a driver name from a HST-id.
SRC-FCM	RIS(optional)	The method to construct a FDM source name from a FCM-id.
SRC-FDM	RIS(optional)	The method to construct a FDM source name from a DIM-id.
SRC-FTP	RIS(optional)	The method to construct a FTP source name from a DCIS-id.
SRC-HST	RIS(optional)	The method to construct a HST source name from a HST-id.
SRC-LFI-BATCH	RIS(optional)	The method to construct a LFI source name from a LFI-BATCH-id.
SRC-LFI-ONLINE	RIS(optional)	The method to construct a LFI source name from a LFI-ONLINE-id.
SRC-REM	RIS(optional)	The method to construct a REM source name from a HST-id.
SRC-RPT	RIS(optional)	The method to construct a RPT source name from a RPT-id.
SRC-SEL	RIS(optional)	The method to construct a SEL source name from a SEL-id.
SRC-SER	RIS(optional)	The method to construct a SER source name from a SER-id.
SRC-SRT	RIS(optional)	The method to construct a SRT source name from a SRT-id.
SRC-TPP	RIS(optional)	The method to construct a TPP source name from a DCIS-id.
SRT	RIS	The formula to create the name of a SRT-id.
SRT-DATA	RIS(optional)	The formula to create the name of a SRT data file. The parameter passed to this formula = 'SRT/'<input format>.

### **39.2.3.2. Variable Name-Standards**

The variable name-standards are defined for file-types and format-types.

#### **File-Type**

To define a file-type using the SURE Explorer:

1. Click **Configuration**.
2. Click **FileType**.
3. Click **Properties**.

A name-standard formula can be defined with the file-type as a key. If a new file is added in SURE, then the name of that new file is determined according to the name-standard formula of the entered file-type.



# Section 40

## Repository

### 40.1.(Repository) Environment

The repository is a term for a general storage and retrieval system. In the SURE software on ClearPath Enterprise Servers, the repository is implemented as a DMSII database. This database is surrounded with a software layer called RESPECT.

The SURE repository contains all information from the application systems. The maintenance copy of a source may be out on disk. For this reason, a dump of the repository database is a safe copy of all application systems.

#### 40.1.1. (Repository) Implementation

The repository is implemented in a DMSII database called INFDB. At installation time, the preferred location of this repository is defined. This definition causes the DASDL to be recompiled matching this defined location.

The database timestamp synchronization combined with the database version that is registered in the object, are sensitive parameters in the Unisys software. A mismatch will cause DMOpen errors in programs.

For these reasons it is advised:

- Copy the newly created description file to the database location.
- Dump the tailored software and the description file after installation or reorganization.
- Install the original description file visible at a place where SURE programs are compiled.

The DASDL of the INFDB is stored in the installation repository as DASDL/INFDB. Using this DASDL you may create as many repositories as required. If the specified conditions are met, all compiled SURE programs will run against all installed repositories.

### 40.1.2. (Repository) Location

Every SURE program opens a file called RESPECT/TITLES, which is created during installation. This file contains the location of the repository. The file system on Unisys ClearPath Enterprise Servers locates this file in a specific order, with or without usercode. The visible file RESPECT/TITLES determines the repository accessed by the programs.

If multiple repositories are present, multiple versions of the RESPECT/TITLES file must be present, each one visible for a specific group of users.

### 40.1.3. (Repository) Why Multiple Repositories

A single repository can maintain multiple environments. Therefore, creating an ACCEPTANCE environment does not require multiple repositories.

Other repositories may be required if:

- An application system does not have the same environments than the regular development.
- The release mechanism of an application system differs from other systems.
- A repository with a single slot must be delivered.
- For test purposes.

Creation of a repository may occur by:

- Copy a user repository.
- Initialize INFDB database and run program.
- RESPECT/REPOSITORY ("INITIALIZE-REPOSITORY.")
- Initialize INFDB database and run the program RIS/COPY/ENVIRONMENT.

### 40.1.4. (Repository) Dump/Load Procedures between Repositories

SURE software supports DUMP/LOAD procedures between two repositories using a TASK or a RELEASE indication. The SURE/Delivery option allows to queue a TASK or a RELEASE in the TRANSFER-REQUEST queue.

The SYSTEM definition for an environment contains a CURRENT RELEASE definition. If this field is entered, every task will get a SOLVED-IN-RELEASE (<Current-Release>) attribute. Delivery of a release will queue all tasks with the SOLVED-IN-RELEASE (<release>) attribute in the transfer queue.

Delivery can be issued for a SOLVED task from an environment that has the <solved> indication defined.

**Dump Procedure**

- Run RESPECT/REPOSITORY("DUMP-REPOSITORY")
- Run RIS/EDITOR;VALUE=9996
- Both these programs create files in the directory RESPECT/DUMP/=

**Load Procedure**

- Run RESPECT/REPOSITORY("LOAD-REPOSITORY");
- If RESPECT/DUMP/SCREENS/TRANSFER IS RESIDENT THEN
- Run RESPECT/REPOSITORY("LOAD-FORMATS TRANSFER");

**Note:** The directory RESPECT/DUMP/= might contain hundreds of files (including source files) to be loaded.

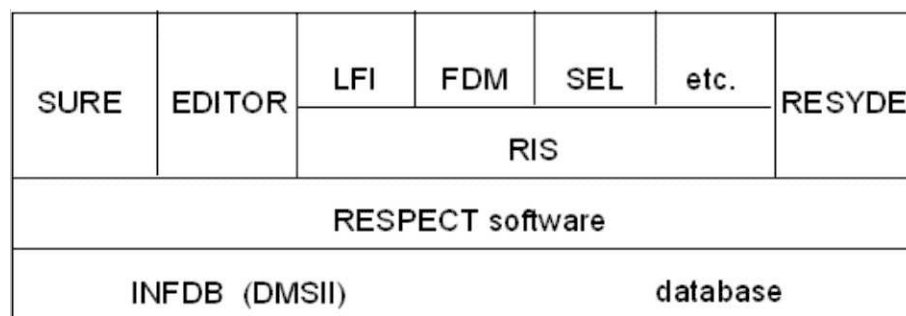
## 40.2. (Repository) Structure

RESPECT database is the core of the RESPECT software (SURE, RIS, EDITOR, and so on). This paragraph describes briefly how information is stored in the RESPECT database and the way information is used by the various RESPECT software tools.

RESPECT is an acronym for: **RE**lational **SPEC**ification **T**ool

Synonyms for the RESPECT database used in documentation for the various RESPECT software tools are: INFDB, data-dictionary or repository.

The following diagram shows the basic modules of the RESPECT software.



005571

### 40.2.1. (Repository) The Structure of the Database (INFDB)

The RESPECT database (INFDB) was originally designed with one basic thought, create a database that can contain ANY type of information, without re-organizing this database when a new type of information has to be stored.

One should compare this idea with a traditional database design: consider a database for a customer registration system. This database has a dataset DCUS with the following data items: CUST-NR, CUST-NAME, and ADDRESS. The example dataset DCUS can only contain customer information of the type customer number, name and address. If we want to store another type of customer information (for example, CUST-ACCOUNT-NR), then we have to re-organize dataset DCUS. If we want to store a new type of non-customer information, then we have to define a new dataset.

The RESPECT database is set up with the DMSII system. The design of the RESPECT database is not based on the principle of a fixed record layout, but on the principle of an inverted file technique: One dataset with item names and another dataset with relations between these item names. An item name itself gives no information, it is a combination of characters (example, AMSTERDAM, HOLLAND, BELGIUM, CAPITAL, BRUSSELS are only words). The real information comes when the item names are combined using relations. Relation *AMSTERDAM:CAPITAL(HOLLAND)* tells us that Amsterdam is the capital of Holland. An identical relation may be established for Brussels and Belgium. When a new item is added (example, NEIGHBOUR) it can use the existing names to form a relation *HOLLAND:NEIGHBOUR(BELGIUM)*.

Notice that the previous example with the customer registration system can be implemented in the RESPECT database without any DMS-reorganizations. Add records in the item name dataset with name = CUST-NR, CUST-NAME and ADDRESS. Enter the variable names too in this dataset: UNISYS, 500, USA. Using some relations between these item names (= records of the relation dataset), the necessary information is captured. Relation *UNISYS:TYPE(CUST-NAME)* tells us that Unisys is a customer name. Relation *UNISYS:CUST-NR(500)* tells us that Unisys has customer number 500. Relation *UNISYS:ADDRESS(USA)* tells us that the address of Unisys is in the USA.

It is obvious that the performance of the system can decrease if the above described inverted file technique is used. In the customer example, only one database access was necessary in the traditional database design and three database accesses were necessary in the database design with the inverted file technique, to obtain the same information.

Still we did not notice a real performance degradation of RESPECT software. The main reason for this is that the amount of users that work simultaneously with the repository is normally rather low (less than 25). Application databases are normally accessed much more often (a hundred transactions for each second is not unusual) and in that case, the inverted file technique is not a good database design due to performance reasons.



A benefit of the inverted file technique is that new information can be stored in the repository without the necessity to reorganize the database first. The open character of the data model allows all kinds of information to be stored in the repository. The only software that has to be developed is an extra on-line function to maintain this newly kept information. Especially for software distributors like Infra Design it is important that the database data model is as stable as possible. Installing a new software release becomes much more difficult if database reorganization is involved.

More details about this relational principle are given in the following sections.

### 40.2.2. (Repository) Datasets in Database INFDB

The INFDB repository consists of the following datasets and items.

#### Dataset DITEM

Fields: ITEM-OWNER, ITEM-NAME, ITEM-VERSION and ITEM-REF-OWNER

This dataset contains a record for each item-name. Each item name is unique and has a unique number, the item-owner.

Item-version and item-ref-owner are only meaningful if the item-name is renamed and the new item name is not yet valid in all environments. Item-ref-owner refers to the owner of the new name. Item-version indicates in which environment the new name is available and in which environment the old name is still available.

Dataset DITEM can be accessed using the set SITEO (key is ITEM-OWNER) and set SITEN (key is ITEM-NAME). Both sets do not allow duplicates.

#### Example

Dataset DITEM

Owner	name	Version	ref-owner
1	DEVELOP		
2	ACCEPT		
3	PRODUCTION		
4	OLDNAME	3 (PRODUCTION)	5 (NEWNAME)
5	NEWNAME	1 (DEVELOP)	4 (OLDNAME)

In this example, item "OLDNAME" is renamed as "NEWNAME" OLDNAME is still available in environments PRODUCTION and ACCEPT. NEWNAME is only available in environment DEVELOP. If NEWNAME is transferred using ACCEPT to PRODUCTION, then the record with OLDNAME is removed from the repository and the owner of NEWNAME is changed to the old owner of OLDNAME. Thus, NEWNAME will get owner 4.

Since the fields ITEM-VERSION and ITEM-REF-OWNER are only used for technical purposes, (technical solution of a renamed item) they will not be mentioned anymore in this document.

### **Dataset DFIL**

Fields: FIL-OWNER, FIL-TITLE, FIL-TITLE-DISP and FIL-VERSION

This dataset contains a record for each FileName. Each fil-title is unique and has a unique number: the fil-owner.

Fil-version is only meaningful if the fil-title is renamed and the new name is not yet valid in all environments.

Functionality for fil-title-disp is not yet implemented.

Dataset DFIL can be accessed using the set SFILO (key is FIL-OWNER) and SFILN (key is FIL-TITLE). Duplicates are not allowed.

### **Dataset DUSER**

Fields: USER-OWNER, USER-NAME, USER-INFO, and USER-TIME

This dataset contains a record for each user-name. The difference between user-names and item-names is that an item-name is often the technical abbreviation of a user-name. For example, item name "coy-nr" has username "company number." Each user name is unique and has a unique number, the user owner.

User-time is the timestamp that the record was updated.

User-info contains the actual user-name. Field user-name is replaced to uppercase and is used as a key. This solution makes it possible to store user names that are a combination of characters in uppercase and lowercase.

Dataset DUSER can be accessed using the set SUSEO (key is USER-OWNER) and set SUSEN (key is USER-NAME).

### **Dataset DFIL and DITEM and DUSER**

If an item-name in dataset DITEM is equal to a fil-title in dataset DFIL, then the corresponding owners are also equal in both datasets.

In all other cases, there is no overlap between the owners in DITEM, the owners in DFIL and the owners in DUSER.

**Dataset DREL**

Fields: REL-VERSION, REL-GROUP, REL-OWNER, REL-CLASS, REL-ASSET, REL-TIME, REL-VALUE (1 through 4).

There is no direct link between datasets DITEM, DFIL and DUSER. Links between these datasets are created by records of the DREL structure. These records are called relations.

- Rel-version identifies the environment for which this relation is valid.
- Rel-group identifies the RIS-entity for this relation (for example, FILE, FORMAT, DATAMODEL, SRT, and FDM).
- Rel-owner corresponds to the fil-title or item-name to which this relation belongs (the owner of the relation).
- Rel-class is the type of the relation.
- Rel-asset is the value of this relation (this can be a file name, an item name or a user name).
- Rel-time is the timestamp the relation was created.
- Rel-value (1 through 4) is four extra fields to maintain technical information.

Dataset DREL can be accessed using the following three sets:

SRELO	(keys: REL-VERSION, REL-OWNER, REL-GROUP, REL-CLASS, REL-ASSET, REL-TIME)
SRELA	(keys: REL-VERSION, REL-ASSET, REL-CLASS, REL-GROUP, REL-OWNER, REL-TIME)
SRELC	(keys: REL-VERSION, REL-CLASS, REL-OWNER, REL-GROUP, REL-ASSET, REL-TIME)

**Dataset DMSG**

Fields: MSG-KEY, MSG-LANG, MSG-VERSION, and MSG-TEXT

This dataset contains (error) messages that are used in the RESPECT software, but also error messages that belong to the application system built with RIS. All RESPECT error messages have a prefix "E>".

- Msg-key is the key of the error message.
- Msg-text contains the error text.
- Msg-lang indicates for which language this message is valid.
- Msg-version indicates for which environment this message is valid.

### **Dataset DINFO**

Fields: INFO-OWNER, INFO-CLASS, INFO-LANG, INFO-VERSION, INFO-GROUP, INFO-INDEX, INFO-TIME, and INFO-DESC

Keys: INFO-VERSION, INFO-OWNER, INFO-GROUP, INFO-CLASS, INFO-LANG, INFO-INDEX

This dataset contains all kinds of variable information, such as source-documentation, operator information, log-information, task description, and coding schemes.

- Info-owner is the item name or file name to which the information belongs.
- Info-class is the type of information (example, loginfo or task description).
- Info-lang indicates for which language the info is valid.
- Info-version indicates for which environment the info is valid.
- Info-group identifies the RIS-entity for this info (for example, FILE, FORMAT, SRT, DATAMODEL, and FDM).
- Info-index is used as a sequence number to identify records with the same owner, class, lang, version and group.
- Info-time is the timestamp that this info-record was updated the last time.
- Info-desc is an alphanumeric field of 320 characters that contains the actual info.

Dataset DINFO can be accessed using the set SINFO.

### **Dataset DSTOR**

Fields: STOR-OWNER, STOR-CLASS, STOR-VERSION, STOR-INDEX, and STOR-DATA.

This dataset contains all source records. A compressing algorithm is used when a file is loaded in DSTOR, which decreases the size of the file with approximately 70%.

- Stor-owner refers to the file name to which the source lines belong.
- Stor-class indicates the type of file: a total file or a delta-file.
- Stor-version gives the version of the file (example, version 34.1 or version 34.2)
- Stor index is used as a sequence number to identify records with the same owner, class and version.
- Stor-data is an alphanumeric field of 2580 characters that contains multiple compressed source lines.

Dataset DSTOR can be accessed using the set SSTOR (Keys: STOR-OWNER, STOR-CLASS, STOR-VERSION, STOR-INDEX).

**Dataset DFMT**

Fields: FMT-TITLE, FMT-OWNER, FMT-VERSION, and FMT-DATA

This dataset contains format descriptions.

- Fmt-title is the name of the format. This Fmt-title is also added as an Item-name in dataset DITEM.
- Fmt-owner refers to the item-name in DITEM (equal to the Fmt-title).
- Fmt-version indicates the version number of the format.
- Fmt-data is an alphanumeric field of 1920 characters, which contains all kinds of format attributes (such as amount of pages, amount of fields, amount of fields for each page, and field sizes).

Dataset DFMT can be accessed using the following two sets:

SFMT (Keys: FMT-TITLE, FMT-VERSION)

SFMT0 (Keys: FMT-OWNER, FMT-VERSION)

**Dataset DNAME**

Fields: NAME-TITLE, NAME-FIELDS, NAME-VERSION, NAME-DATA (occurs 128)

This dataset contains the field names of formats.

- Name-title is the name of the format. This name is equal to item Fmt-title in dataset DFMT.
- Name-fields give the amount of fields of the format.
- Name-version identifies the version number of the format.
- Name-data is a table with field names.

**Dataset DSCR**

Fields: SCR-KEY, SCR-LANG, SCR-DATA

Keys: SCR-KEY, SCR-LANG

This dataset contains the layout of the formats.

- Scr-key refers to the format-owner, the page of the format and the environment of the format.
- Scr-lang is not used at the moment.
- Scr-data is an alphanumeric field of 1920 characters, which contains the actual format layout.

Dataset DSCR can be accessed using the set SSCR

### 40.2.3. (Repository) The Use of Relations

Information is created by relations between data items. A data item is a unique entity that is stored in the RESPECT database. The following data items are separately recognized:

<item name> = variable name for a data item

<file name> = variable name which can be given to a source

Each item name and file name appears only once in the database. The information is built through creating relations between file names and item names. Relations are built as follows:

version (environment)	=	item name
group (RIS entity)	=	item name
Owner	=	file name or item name
class (type relation)	=	item name
asset (value of the relation)	=	item name or file name or user name

The way in which information is stored is explained in the following example.

#### Example of the use of relations

The fields "version" and "group" are not mentioned in this example.

Dataset DITEM (item names)		Dataset DFIL (file names)	
nr	Name	nr	name
1	AUTHOR	3	RESPECT/SURE
2	LIBRARY	8	RESPECT/LIBRARY
4	COPY FILE	9	RESPECT/DEFINES
5	INFDB		
6	INFRA		
7	STATUS		
10	PRODUCTION		
11	DATABASE		
12	TYPE		

Dataset DREL (relations)

	<b>Owner</b>		<b>class</b>		<b>asset</b>
3	(RESPECT/SURE)	1	(AUTHOR)	6	(INFRA)
3	(RESPECT/SURE)	2	(LIBRARY)	8	(RESPECT/LIBRARY)
3	(RESPECT/SURE)	4	(COPY FILE)	9	(RESPECT/DEFINES)
3	(RESPECT/SURE)	7	(STATUS)	10	(PRODUCTION)
5	(INFDB)	12	(TYPE)	11	(DATABASE)
6	(INFRA)	12	(TYPE)	1	(AUTHOR)
8	(RESPECT/LIBRARY)	1	(AUTHOR)	6	(INFRA)
8	(RESPECT/LIBRARY)	4	(COPY FILE)	9	(RESPECT/DEFINES)
8	(RESPECT/LIBRARY)	7	(STATUS)	10	(PRODUCTION)
8	(RESPECT/LIBRARY)	11	(DATABASE)	5	(INFDB)
9	(RESPECT/DEFINES)	1	(AUTHOR)	6	(INFRA)
9	(RESPECT/DEFINES)	7	(STATUS)	10	(PRODUCTION)
9	(RESPECT/DEFINES)	11	(DATABASE)	5	(INFDB)

The above diagrams give an example of the contents of the RESPECT database.

All item names are uniquely stored in dataset DITEM. The total set of item names consists of pre-defined item names (for example, STATUS, PRODUCTION, DATABASE, AUTHOR, LIBRARY, TYPE, COPY FILE) and user defined item names (for example, INFRA, INFDB.) Every item name has a unique number.

All file names are uniquely stored in dataset DFIL. Every file name has a unique number. These numbers do not overlap the item numbers.

All source records are stored in dataset DSTOR.

The information belonging to a source is built up by a set of relations. These relations are stored in dataset DREL.

A relation consists of an <owner>, a <class> and an <asset> of which only the numbers are stored in DREL (hence the brackets in the diagram).

- <owner> = The file name or item name to which this relation belongs (the owner of the information).
- <class> = The type of this relation (the type of information).
- <asset> = The value of this relation (this can be a File name, an item name or a timestamp).

### Relations

In the above example the following information is built up through relations:

Source RESPECT/SURE has AUTHOR INFRA

Source RESPECT/SURE uses COPY-FILE RESPECT/DEFINES

Source RESPECT/SURE uses LIBRARY RESPECT/LIBRARY

Source RESPECT/SURE uses LIBRARY RESPECT/LIBRARY

Source RESPECT/SURE has STATUS PRODUCTION

INFRA is an AUTHOR

INFDB is a DATABASE

Source RESPECT/LIBRARY has AUTHOR INFRA

Source RESPECT/LIBRARY uses COPY FILE RESPECT/DEFINES

Source RESPECT/LIBRARY uses DATABASE INFDB

Source RESPECT/LIBRARY has STATUS PRODUCTION

Source RESPECT/DEFINES has AUTHOR INFRA

Source RESPECT/DEFINES uses DATABASE INFDB

Source RESPECT/DEFINES has STATUS PRODUCTION

Dataset DREL has three sets:

1. Set SRELO with primary key OWNER
2. Set SRELC with primary key CLASS
3. Set SRELA with primary key ASSET

The information given is presented with the use of set SRELO. However, if we use set SRELA, the following information is also available:

- INFRA is the AUTHOR of RESPECT/SURE, RESPECT/DEFINES and RESPECT/LIBRARY. DATABASE INFDB is used by RESPECT/DEFINES and RESPECT/LIBRARY.
- RESPECT/DE FINES is a COPY FILE of RESPECT/SURE and RESPECT/LIBRARY.
- PRODUCTION is the STATUS of RESPECT/SURE, RESPECT/DEFINES and RESPECT/LIBRARY.
- RESPECT/LIBRARY is a LIBRARY of RESPECT/SURE.



**Information**

Where a relation exists, the information is available. When information is not available, no relation will be present. For example, Source RESPECT/DEFINES has no copy file relations (relations with class COPY-FILE), so it uses no copy files.

**Inquiry relations**

All relations that point to a file or item name can be made visible with function "inquire relations."

**40.2.4.(Repository) Detailed Information about Relations**

Information is stored in the repository using relations.

**Example**

The indication that the RESPECT/SURE file is placed in the compile-queue of environment DEVELOP is stored in the following relation.

---

**Version\*Group|Owner:Class (Asset)**

---

DEVELOP\*FILE-CONTROL|RESPECT/SURE:COMPILE-STATUS(TO-COMPILE)

The version of the relation identifies an existing environment, or it is zero. If the version is zero, then the relation is valid for all environments.

The group of the relation can be one of the following items.

SEL	SEL-CONTROL
SRT	SRT-CONTROL
FCM	FCM-CONTROL
FDM	FDM-CONTROL
FCT	FDM-CONTROL
LFI-BATCH	LFI-CONTROL
LFI-ONLINE	LFI-CONTROL
CNV	CNV-CONTROL
RPT	RPT-CONTROL
HST	HST-CONTROL
REM	REM-CONTROL
TPP	TPP-CONTROL
SER	SER-CONTROL
FTP	FTP-CONTROL
DCIS	DCIS-CONTROL
FILE	FILE-CONTROL

FORMAT	FORMAT-CONTROL
DATABASE	DATABASE-CONTROL
DATAMODEL	DATAMODEL-CONTROL
WINDOWS	WINDOWS-CONTROL
SEL-FUNC	SEL-FUNC-CONTROL
SER-FUNC	SER-FUNC-CONTROL
ONLINE-TRN	ONLINE-TRN-CONTROL
BATCH-TRN	BATCH-TRN-CONTROL
MESSAGE	MESSAGE-CONTROL
ITEM	ITEM-CONTROL
USER	
PROBLEM	
REQUEST	
OPTION	
TYPE	
PROJECT	

All groups with a corresponding CONTROL group identify a RIS entity.

All relations with group = <RIS entity> and with the same owner and version, form together the definition of that module.

All relations with group = <RIS entity>-CONTROL and with the same owner and version, control that module (owner) in that environment (version).

**Example**

Consider file RESPECT/SURE written by INFRA. This file has version 25.1 in DEVELOP environment and version 24.1 in ACCEPT environment. The file is placed in the transfer-queue of DEVELOP environment and is compiled in ACCEPT environment.

This results in the following relations.

Version*Group Owner:Class (Asset)	
DEVELOP*FILE	RESPECT/SURE:AUTHOR ( INFRA )
DEVELOP*FILE	RESPECT/SURE:STOR ( 25 . 1 )
DEVELOP*FILE-CONTROL	RESPECT/SURE:TRANSFER-STATUS ( TO-TRANSFER )
ACCEPT*FILE	RESPECT/SURE:AUTHOR ( INFRA )
ACCEPT*FILE	RESPECT/SURE:STOR ( 24 . 1 )
ACCEPT*FILE-CONTROL	RESPECT/SURE:COMPILE-STATUS ( COMPILED )

In this example, the source is RESPECT/SURE; the group is FILE and the definition of source RESPECT/SURE is formed by the STOR and AUTHOR relations.

It is possible to transfer a source from an environment to a higher environment. In that case, only the relations that form together the definition of that source are transferred. In the above example these are the relations with group = FILE. It is obvious that the CONTROL relations cannot be transferred, since they control the source in a specific environment.

### Example (Continued)

Consider source RESPECT/SURE of the previous example.

If this source is transferred from DEVELOP to ACCEPT, then the changed relations with group = FILE are copied to environment ACCEPT.

Action	Version*Group Owner:Class (Asset)
DELETE	ACCEPT*FILE RESPECT/SURE:STOR(24.1)
ADD	ACCEPT*FILE RESPECT/SURE:STOR(25.1)

Notice that the author-relation is not copied, because it has the same value in DEVELOP and ACCEPT.

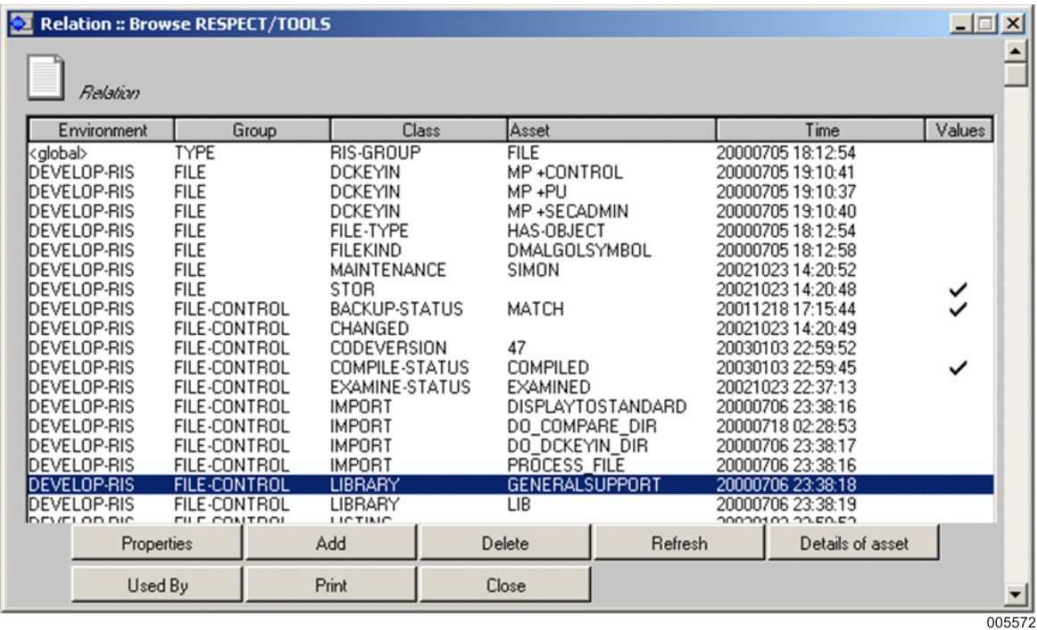
Notice that the FILE-CONTROL relations are not copied. For example, a file that is placed in the transfer queue of DEVELOP, does not have to be placed automatically in the transfer queue of ACCEPT if this file is transferred.

## 40.2.5.RELATION Functionality in SURE Browser

The following "RELATION" functions are available in the SURE browser.

Function	How
Show all relations of a file	<ol style="list-style-type: none"> <li>1. Right click the <b>filename</b>.</li> <li>2. Click <b>Miscellaneous</b>.</li> <li>3. Click <b>Relation</b>.</li> </ol>
Show all relations of a task	<ol style="list-style-type: none"> <li>1. Right click the <b>task name</b>.</li> <li>2. Click <b>Relation</b>.</li> </ol>
Relate: modify a relation of a file or task	<ol style="list-style-type: none"> <li>1. Select a <b>file/task</b>.</li> <li>2. Click <b>Tools</b></li> <li>3. Click Multi <b>Relate</b>.</li> </ol> <p>OR</p> <ol style="list-style-type: none"> <li>1. Show relations of file/task</li> <li>2. Click <b>Properties</b>.</li> <li>3. Click <b>Modify</b>.</li> </ol>
Multi Relate: modify relations of a group of files or tasks	<ol style="list-style-type: none"> <li>1. Select a group of <b>files/tasks</b>.</li> <li>2. Click <b>Tools</b>.</li> <li>3. Click <b>Multi Relate</b>.</li> </ol>

Example: Show All Relations of a File/Task

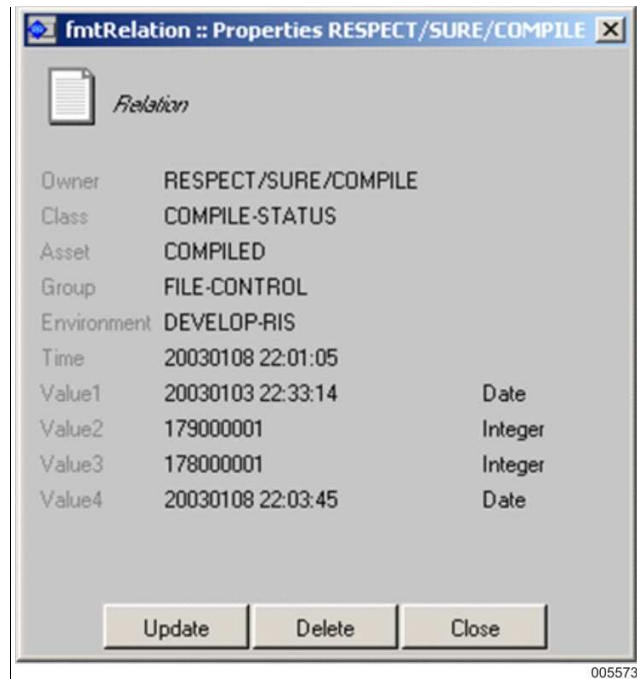


Column	Description
Environment	The environment where this relation is used. <Global> = visible for all environment.
Group	Relations are grouped. Relations with group FILE will be transferred to another environment. Relations with group FILE-CONTROL will not be transferred.
Class	What kind of information is kept in the relation?
Asset	The value of the info.
Time	Each relation has its own timestamp.
Values	Four fields with variable info.

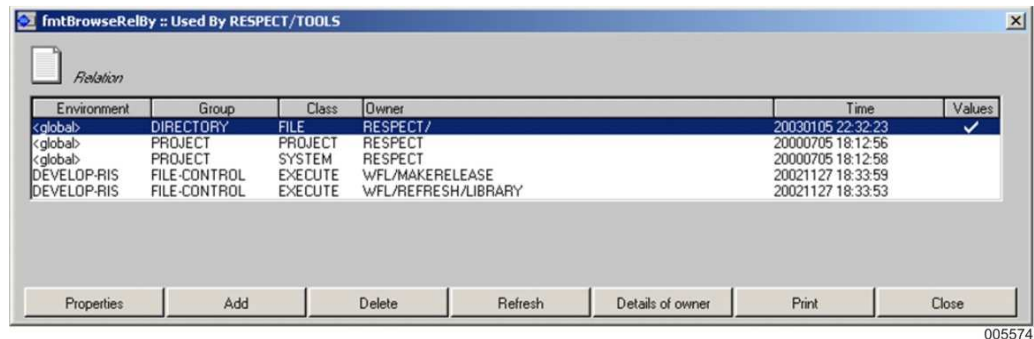
Button	Function
Properties	Show all details of the selected line.
Add	Add a new relation.
Delete	Delete the relation on the selected line.
Refresh	Refresh the screen.
Details of Asset	Show the relations of the 'asset' in the selected line (in this case library GENERALSUPPORT).
Used by	Show how the file or task in the title bar (in this case RESPECT/TOOLS) is referenced.

The **Properties** button gives the following screen:

It shows four variables "relation values."



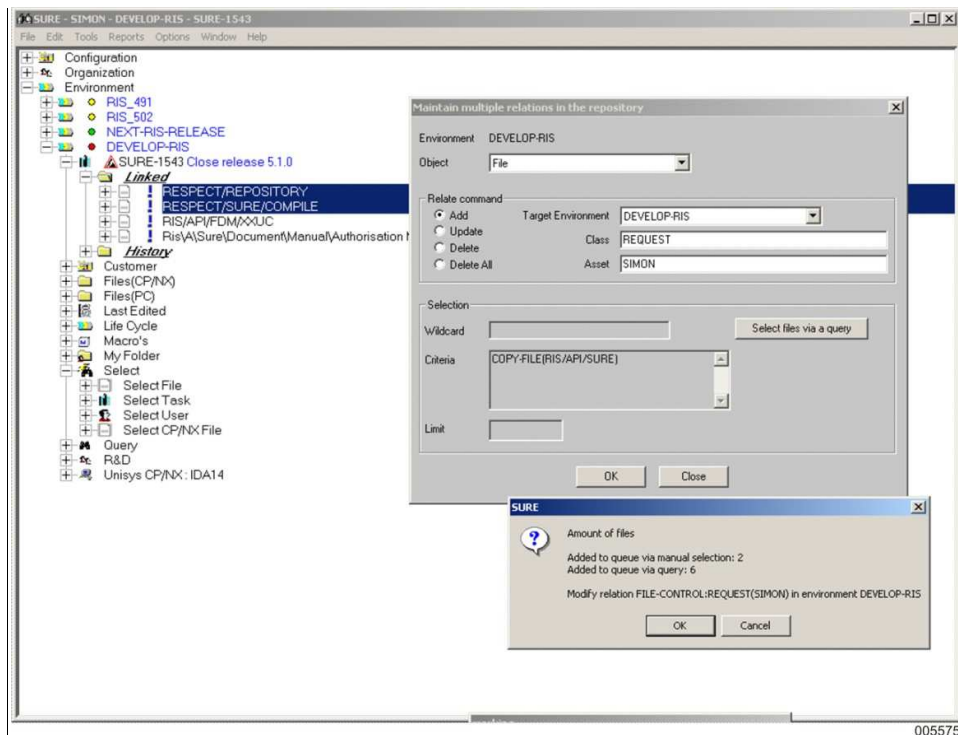
The **Used By** button gives the following screen.



Column	Description
Owner	The file, item or task that references the id in the title bar (in this case RESPECT/TOOLS).

### Modify Relations through “Multi Relate” Dialog

1. Click **Function Tools**.
2. Select **Multi relate**.



This example adds a relation with

Group =	FILE-CONTROL (automatically determined by SURE)
Class =	REQUEST
Asset =	SIMON

to eight files in environment DEVELOP-RIS. Two of the files were selected in the browser; the other six files were selected using the “files with COPY-FILE RIS/API/TASK” query.

The “Object” field on the MultiRelate dialog identifies the group: files or tasks. If one or more tasks are selected in the SURE browser, then “Object” field is pre-filled with “Task”; otherwise, the field is pre-filled with “File.”

It is possible to switch between “File” and “Task.” The selected tasks in the browser are also added to the relate-queue if the object is “Task.” The selected files in the browser are also added to the relate-queue if the object is “File.”

Field “Environment” is pre-filled with the environment of the current selected line in the browser. It is possible to change the environment, but not if files or tasks are selected in the browser.

## **Technical Details and Considerations**

Most information on all screens is stored in the repository using relations. Modifying a relation will change the properties of a file or task on a screen in an uncontrolled way (this is similar to the case that a user is changing information in a dataset using ERGO or DMS-inquiry, instead of doing that using a regular application program). Therefore, it is not recommended to add, delete or modify relations if you are not familiar with the concept of relations.

The "Multi Relate" function is secured with "Multi Relate" authorization bit.

Modify a relation using the "Relations" screen is secured with "Relate" authorization bit.





# Section 41

## API Interfaces

### 41.1. Overview

The SURE package provides four different API interface implementations.

For Customer use:

- DDE.
- On Unisys ClearPath Enterprise Servers using SURE RPT module.
- The SITE library.

Used by Infra Design:

- On Windows using C++ declared entry points.
- On Unisys ClearPath Enterprise Servers using the RESPECT interface.

Each different API implementation has its own flexibility and advantages. The API interfaces for customer use are described here in detail.

### 41.2. OLE and DDE Interfaces to SURE for Windows

All objects and methods in SURE can be controlled from other applications, such as WORD, EXCEL, or a VISUAL BASIC application. SURE for Windows supports OLE automation and DDE allowing this remote control. Although, the syntax is different between OLE automation and DDE, the principle for both mechanisms is the same. Therefore, the principle is discussed first and thereafter the syntax for OLE and DDE.

SURE for Windows supports a DATA structure for each opened object. However in case of remote control, such as OLE and DDE, a single DATA structure is used. This means that one application can use the same variables and results as another application. For example, if a Visual Basic application does an INQUIRE of a FILE, a WORD application can see the results of this inquire.

#### Connecting to SURE for Windows

Connection to the SURE system is achieved by connecting to the configured environment. For the SURE installation, this environment is called **SURE Browser** and this name is used in the DDEInitiate (DDE) or the Connect (OLE) function.

### **SURE for Windows Supports Three Categories of Commands:**

- SETDATA  
This command allows you to set variables in SURE for Windows. This command is used to set variables before executing a method.
- GETDATA  
Variables from SURE for Windows are obtained with this command. This command is used to obtain result variables after executing a method.
- EXECUTE\_A\_METHOD\_FOR\_AN\_OBJECT  
This command is used to perform a method in SURE for Windows. This command is executed synchronously. Therefore, the calling application gets control back after the execution of the method is complete.

### **SURE for Windows Supports Two Categories of DATA:**

- DATA  
This is the default where a variable can have a single value.
- TABULAR DATA  
This mode supports tabular forms where a number of variables may occur multiples times. To obtain a variable, a format name and an index must be supplied.

### **How Do I Know Which Method to Execute or What the Field Name Is?**

In SURE for Windows you can define various objects, methods and field names. In the SURE product, the major objects are "File" and "Task." However, various weak entities are defined, such as "Patch Files" or "Statistics" and each has its own variables.

The included help files will show the names of the objects, methods, field and their correlation. Exploring this help file is one way of knowing the names to use. Another way of exploring these names is executing the functions using the SURE for Windows SURE interface. Setting the DEBUG option will show all executed methods under the "BROKER" category. Note that setting this DEBUG option requires setting of the option Windows/MDI enabled. This option opens the MDI window that allows setting of the option Windows/Debug.

## **41.2.1. DDE Interface**

The DDE interface implemented in SURE for Windows uses three commands. This implementation supports the programmatic interface. However, SURE for Windows also supports a dynamic interface including "hot links." These interfaces are created on user request through "Field/Copy" or "Field/Paste."

### 41.2.1.1.DDEExecute

Parameters

<Object>:<Method>:<Option>

The <Option> may be empty or it may have the value of "complete." The "complete" indication instructs SURE for Windows to complete a BROWSE function. The WAN and SQL implementations of browse functions supports partial retrieval. The user submits a NEXT request using the scroll bar. The "complete" indication instructs SURE for Windows to obtain all records and then return control to the user. If this option is not used, the caller is responsible for obtaining next requests. The option may also include a calling object name. For weak entities, it is required to set the calling object name to the entity object.

#### Example

```
DDEExecute("File:_ris_BrowseShowD")
DDEExecute("File:_br_Project:Complete")
```

### 41.2.1.2.DDEInitiate

Parameters

<Service>	(= AW_OBJ)
<Topic>	(= <environment> [,PRIVATE])

The initiate function is used to establish a connection between SURE for Windows and the calling program. This function returns a handle that is used in the subsequent calls, depending on the DDE implementation. The <Topic> equals an <environment>, which must be opened in the SURE for Windows system. The default installation creates an environment called SURE. The PRIVATE option allows you to create a private datamap used for this connection only.

The Name AW\_OBJ is default installed. However, if multiple SURE systems are running on the client, the SERVICENAME in the [GLOBAL] section of the configuration file may overwrite this default name.

#### Example

```
DDEInitiate("AW_OBJ","SURE Browser")
```

### 41.2.1.3.DDEPoke

Parameters

<FieldName>

<FieldValue>

#### Example

```
DDEPoke("FileName","RESPECT/SURE")
```

The poke command is used to SETDATA in SURE for Windows. The DDE interface does not support tabular data in this command.

### 41.2.1.4. DDERequest

Parameters

<FieldName>	for non tabular data
<FormatName>.<FieldName>[index]	for tabular data
<FormatName>..more	a more indication
<FormatName>..count	a count indication

Return Value

<FieldValue> concatenated with Cr Lf

#### Example

```
s = DDERequest("FileChanged")
s = DDERequest("Statistics.RunET[1]")
s = DDERequest("Statistics..more")
s = DDERequest("Statistics..count")
```

The Request command is used to GetData in SURE for Windows. The DDE interface supports tabular data using specific syntax.

## Example Source in WORD

### Macro Listing

```
Sub MAIN
wordWin$ = "Microsoft Word - " + WindowName$()

channel = DDEInitiate("AW_OBJ", "SURE Browser")
DDEPoke channel, "UserPassword", "TTESTUSER"
DDEPoke channel, "UserCode", "PASS"
DDEExecute channel, "User:Logon"

DDEPoke channel, "FileName", "MY/FILE/NAME"
DDEExecute channel, "File:_cmd_Property"

User$ = DDERequest$(channel, "SourceUserCode")
End Sub
```

## 41.2.2. OLE Interface

The SURE for Windows system can be used as OLE automation server. A number of generic methods are used to drive the SURE for Windows application. The SURE for Windows system registers itself and its methods in the windows registry database. For this reason it is required to run SURE for Windows once before using OLE automation.

For all of the methods exported by SURE for Windows is given an example in Visual Basic syntax.

### 41.2.2.1.CreateObject

This is standard Visual Basic syntax that creates an OLE object. This object is thereafter used as a stub for executing methods or interrogating properties. The SURE for Windows system exports one type of object, which is used to drive the SURE for Windows system. The object name registered in the registry by SURE for Windows is called "RIS.AUTOMATION."

If multiple SURE versions can run concurrently, a parameter can be configured in the configuration file that sets the name to be used in the CreateObject function. The [GLOBAL] section in the configuration file may contain an entry OLESERVER=<n>, which extends the name RIS.AUTOMATION with the number .<N>.

#### Example

```
DIM obj AS OBJECT
SET obj = CreateObject("RIS.AUTOMATION")

[GLOBAL]
OLESERVER=4

DIM obj AS OBJECT
SET obj = CreateObject("RIS.AUTOMATION.4")
```

### 41.2.2.2. SetValue

Parameters

pszItemName	LPCSTR
pszItemValue	LPCSTR

This method is used to set the value of a non tabular data item in SURE for Windows.

#### Example

```
obj.SetValue "FileName" , "RESPECT/SURE"
```

### 41.2.2.3. GetValue

Parameters

pszItemName	LPCSTR
-------------	--------

Return value

pszItemValue	BSTR
--------------	------

This method is used to obtain the value of a non tabular data item in SURE for Windows.

#### Example

```
s$ = obj.GetValue "FileName"
```

### 41.2.2.4. MethodForObject

Parameters

pszMethodName	LPCSTR
pszObjectName	LPCSTR
pszOption	LPCSTR

Return value	short
--------------	-------

0 - not successful  
1 - successful

The MethodForObject causes SURE for Windows to execute a method and it suspends the caller until a response (or timeout) is received. All error conditions, including an application error response are translated to a return value of zero (not successful). The calling application can obtain error information by GetData methods for the error fields used by SURE for Windows.

ErrorText	contains a description of the error
_ris_ErrorType	contains a two character error category
_ris_ErrorNr	contains a four digit error number

The pszOption parameter is either empty or it contains the constant "Complete." For browse methods, this option instructs SURE for Windows to complete the browse method before returning control. The user interface gets a browse results (for example 10 rows) and get the next browse result as a result of scrolling by the user. The "Complete" option instructs SURE for Windows to obtain all rows before returning control. The pszOption parameter can also contain a calling object name. This is true for weak entities where the calling object name equals to the entity object.

**Example**

```
if obj.MethodForObject("ris_BrowseStart","FileTaskCurrent","File,Complete")
    rem process result
else
    s$ = obj.Getdata "ErrorText"
end if
```

**41.2.2.5. Connect**

Parameters

pszConnect	LPCSTR
------------	--------

Return value	short
--------------	-------

0 - not successful

1 - successful

Connect establishes a connection between the caller and the SURE for Windows system. This pszConnect string contains an open environment. For the installed SURE systems, this environment is called SURE Browser. Optionally, the PRIVATE option may be provided. This instructs the SURE system to create a private datamap for this connection.

**Example**

```
obj.Connect "SURE Browser"
obj.Connect "SURE Browser,PRIVATE"
```

**41.2.2.6. Disconnect**

The Disconnect method terminates a connection. A Connect may re-establish the connection using the same object with another repository.

**Example**

```
obj.Disconnect
```

### 41.2.2.7. InitTab

Parameters

pszFormatName	LPCSTR	
nFields	short	number of columns or fields
nOccurs	short	number of rows
bMore	short	if tab may be extended

The InitTab instruct SURE for Windows to create a Tabular Data Structure. The caller must set the field names using SetTabName and set the data by SetTabdata.

#### Example

```
obj.InitTab "MyData" , 2 , 5 , 0
obj.SetTabName "MyData" , "FileName" , 0
obj.SetTabName "MyData" , "FileChanged" , 1
for n = 0 step 1 until 4 do
    SetTabValue "Mydata" , "FileName" , fn[n] , n
    SetTabValue "Mydata" , "FileChanged" , fc[n] , n
endfor
```

### 41.2.2.8. SetTabName

Parameters

pszFormatName	LPCSTR
pszItemName	LPCSTR
nField	short

This method is used to set the field names for a tabular data structure.

#### Example

```
obj.InitTab "MyData" , 2 , 5 , 0
obj.SetTabName "MyData" , "FileName" , 0
obj.SetTabName "MyData" , "FileChanged" , 1
```



#### 41.2.2.9. SetTabValue

Parameters

pszFormatName	LPCSTR
pszItemName	LPCSTR
pszItemValue	LPCSTR
nOccurs	short

This method is used to set data in a tabular data structure.

##### Example

```
obj.InitTab "MyData" , 2 , 5 , 0
obj.SetTabName "MyData" , "FileName" , 0
obj.SetTabName "MyData" , "FileChanged" , 1
for n = 0 step 1 until 4 do
    SetTabValue "Mydata" , "FileName" , fn[n] , n
    SetTabValue "Mydata" , "FileChanged" , fc[n] , n
endfor
```

#### 41.2.2.10. GetTabValue

Parameters

pszFormatName	LPCSTR
pszItemName	LPCSTR
nOccurs	short

Return Value                      BSTR

value for format.item[nOccurs]

This method returns the value for an item in a tabular data structure.

##### Example

```
for n=0 step 1 until obj.GetTabOccurs("MyData") - 1
    fn[n] = GetTabValue("MyName" , "FileName" , n)
endfor
```

### 41.2.2.11. GetTabOccurs

Parameters

pszFormatName	LPCSTR
---------------	--------

Return value	short
--------------	-------

returns the number of rows in a tabular data format

#### Example

```
for n=0 step 1 until obj.GetTabOccurs("MyData") - 1
    fn[n] = GetTabValue("MyName" , "FileName" , n)
endfor
```

### 41.2.2.12. GetTabFieldCount

Parameters

pszFormatName	LPCSTR
---------------	--------

Return value	short
--------------	-------

returns the number of fields or columns in a tabular data format

#### Example

```
for n=0 step 1 until obj.GetTabFieldCount("MyData") - 1
    fn[n] = GetTabFieldName("MyName" , n)
endfor
```

### 41.2.2.13. GetTabFieldName

Parameters

pszFormatName	LPCSTR
nField	short

Return value	BSTR
--------------	------

the name of the field

#### Example

```
for n=0 step 1 until obj.GetTabFieldCount("MyData") - 1
    fn[n] = GetTabFieldName("MyName" , n)
endfor
```

### 41.2.2.14.Example Source

This Java Script performs the following actions:

- Logon to SURE
- Inquire a File (and show a file property)
- Inquire actual task list for this file
- Inquire linked entities for every actual task
- Inquire copy file references for every linked entity

```

/*****
*
*   SURE interface utility
*
*****/
*   Description:
*   -----
*   This sample script
*       Logon to SURE
*       Inquire a File (and show a file property)
*       Inquire actual task list for this file
*       Inquire linked entities for every actual task
*       Inquire copy-file references for every linked entity
*
*   To Run:
*   -----
*       sure.js
*
*   Used Constructs: for the Sure object
*   -----
*       Connect          - Make a connection with the SURE subsystem
*       Disconnect       - Terminate the connection with the SURE subsystem
*       SetValue         - Set a single property value (normally a http request variable)
*       GetValue         - Get a single property value (normally a http response variable)
*       GetTabValue      - Get a property in a multi row output (normally a http response tab
variable)
*       GetTabOccurs     - Get the number of rows for a particular response
*       MethodForObject  - Execute a method on an object by the SURE subsystem
*
*   Used Constructs: for the WScript object
*   -----
*       CreateObject     - Make a connection with the SURE application
*       Echo             - Show information in a message box
*       quit             - terminate
*
*****/

var nTask;
var nLink;
var nRef;
var Sure;

/-- Make a logical connection to the SURE executable on the client

Sure = WScript.CreateObject("RIS.AUTOMATION");
Sure.Connect("SURE Browser,private");

/-- Logon to the system

```

```
//-- Set Value UserCode and PassWord
//-- normally these are http request variables

Sure.SetValue("UserPassword","PAZZS");
Sure.SetValue("UserCode","SIMONDEV");
Sure.SetValue("Product","SfW");

if (Sure.MethodForObject("LogonEx","User",""))
    WScript.Echo("Logon OK");
else
{
    WScript.Echo("Logon failed:" + Sure.GetValue("ErrorText"));
    Sure.Disconnect();
    WScript.quit(0);
}

// inquire file with a name

Sure.SetValue("FileName","S/FI/ALP/FIALP0015");
Sure.MethodForObject("_cmd_State","File","");

// retrieve one of the file properties

WScript.Echo(Sure.GetValue("FileName") + " FileKind      = " + Sure.GetValue("FILEKIND"));

// inquire actual task list for this file and loop through them

Sure.MethodForObject("_ris_BrowseStart","FileTaskCurrent","File,Complete");
for (nTask = 0; nTask < Sure.GetTabOccurs("FileTaskCurrent"); ++nTask)
{
    WScript.Echo("actual linked task " + Sure.GetValue("FileName") +
        " -> " + Sure.GetTabValue("FileTaskCurrent","TaskName",nTask));

    // search all linked entities for this task and loop through them

    Sure.SetValue("TaskName",Sure.GetTabValue("FileTaskCurrent","TaskName",nTask));
    Sure.MethodForObject("_tsk_Linked","Task_Func","Task,Complete");
    for (nLink = 0; nLink < Sure.GetTabOccurs("TaskItemBrowse"); ++nLink)
    {
        WScript.Echo("linked to task " + Sure.GetValue("TaskName") +
            " -> " + Sure.GetTabValue("TaskItemBrowse","_ObjectItem",nLink));

        // search all references for this file and loop through them

        Sure.SetValue("LinkName",Sure.GetTabValue("TaskItemBrowse","_ObjectItem",nLink));
        Sure.SetValue("RefDirection","0");
        Sure.MethodForObject("_ris_BrowseRefC","FileReference","Complete");
        for (nRef = 0; nRef < Sure.GetTabOccurs("FileReference"); ++nRef)
        {
            // Just select copy-file type of references

            if (Sure.GetTabValue("FileReference","RefClass",nRef) == "COPY-FILE")
                WScript.Echo("copy file " + Sure.GetValue("LinkName") +
                    " -> " + Sure.GetTabValue("FileReference","RefAsset",nRef));
        }
    }
}

Sure.Disconnect();
```

# Section 42

## Terminal Emulation Interface

### 42.1. Overview

This chapter describes the standard screen functions for all RESPECT online programs. This chapter includes the screen functions for SURE, RESYDE, RESPECT/EDITOR, and RIS.

### 42.2. Screen Handling for the On-line Functions

The name of the online program is RIS/MENU. All available online functions are bound in this object.

The online program is a mainframe program that accesses the terminal in the emulation mode (TD830 compatible). A terminal emulation program, such as InfoConnect, ICC, and Unisys Terminal Emulator, is required to use the SURE software from a PC.

As the software uses most of the TD830/ET1100 terminal functions, the user should be familiar with these terminals or emulator operations. The TD830/1100 keyboard functions include:

- Transmit: A button to transmit the data.
- Receive: A key to put the terminal into receive mode.
- Specify: A key used to specify a certain position on the screen. This function is similar to the mouse click PC programs.

#### **Ctrl nn Transmit Commands**

The Ctrl commands can be used for general functions in every screen. These functions are similar to the accelerator keys on PC programs.

Since the online program uses a terminal emulator, actions to the mainframe can only be initiated by one of the following functions:

- **Transmit**

The Transmit key sends all information that has been keyed into the mainframe.

- **Specify**

The specify key is used as a pointing device. It can be used only at specific locations on the screen.

A number of screens are presented in forms mode. The forms mode allows data to be entered within field delimiters only. Except for specify functions, the cursor cannot be moved to any position outside these delimiters. Only the information within these delimiters is transmitted; information outside the delimiters is not sent.

The terminal has an option that allows two modes for transmitting fields in the forms mode.

- Transmit to Cursor

This mode transmits all data up to the cursor location. This option can confuse new users and should be reset for the terminal emulator.

- Transmit all Fields

This mode transmits all fields between forms delimiters, irrespective of the cursor position. This option should be set for the terminal emulator.

If the presented screen contains forms delimiters, the terminal is set in the forms mode by the program after every message is received. The emulator software takes the terminal out of the forms mode after each transmit. If two transmits are performed without receiving data in between the first transmit and the second transmit, the second transmit will not be in the forms mode. As a result, all the characters that appear on the screen up to the cursor position or the total screen (if the cursor is at the home position), are sent and this results in an error message.

Transmitting a screen with forms delimiters out of the forms mode can lead to unexpected errors and should be avoided. To avoid errors, check the cursor position before transmitting.

Some screens do not contain forms delimiters and must be transmitted out of the forms mode. All data from the home position to the cursor position will be sent to the mainframe. The functions associated with these screens are listed below:

- Paint a new format (RESPECT/EDITOR)
- Change the format layout (RESPECT/EDITOR)
- Add a new field (RESPECT/EDITOR)
- Report window utility (RESPECT/EDITOR)
- Add/update documentation (ALL)

**Screen: Example**

```

InterCom Terminal Emulation - [Env 1 - TCP_49]
Spcfy back      <Short screen description (line 1)>      infra design
ITEM INFO RELATION HELP
<STATUS INFORMATION >      (optional)
<FUNCTION FIELDS (line 3 - 20)
<Error message field (line 24) >

1  5  Pg=1  TCP_49      Norm Loc LTAI
005576

```

All RESPECT screens have the following standard layout.

**42.2.1. Function Line: Top Line**

This line contains a description of the function being performed.

The first few characters on this line are "Specify back." If the cursor is placed at this location and the Specify key is hit, the program will return to the previous screen or finish.

**42.2.2. Command Line: Second Line**

The command line has a set of general buttons that represent corresponding commands. Commands are selected by moving the cursor to one of these commands and pressing Specify.

There are a number of general commands that appear in most screens. Certain screens also contain function-specific commands on this line. The general commands are listed below.

### 42.2.2.1. Specify on ITEM

#### Function

Requests item names that resemble an incorrectly entered item name. This function can be used to find the desired item name and to browse through a list of item names.

#### Usage

- The function can be used on all screens where an item can be inquired on by using the function "INQ."
- The function can also be used if an error message indicates that an item name is not present in the database.
- By <Specify> on "ITEM," a continuation screen is displayed with a maximum of 18 lines.
- Item-names are given in alphabetical order. Each line displays one item name. The line with the item name most closely resembling the incorrectly entered item (or current item) is highlighted.
- The following functions are used to page through the continuation screens:
  - <Specify> on NEXT
  - <Specify> on PREV
  - <Specify> on FIRST (original) page
  - <Specify> on LAST if selected, the first (original) page is displayed
- After the correct name is found, it is possible to <specify> on this name. The previous screen will be presented. In certain cases, the specified item is automatically entered in the error field.



## Screen: Inquire Item Names

InterCom Terminal Emulation - [A40NTW - TA14021]

Spcfy back RESPECT ( RELational SPEcification Tool ) infra design

NEXT PREV FIRST LAST HELP

COMPILER-DATE  
COMPILER-FAST  
COMPILER-STATUS  
COMPILER-SYNTAX  
COMPILER-TEST  
COMPILER-TYPE  
COMPILED  
COMPILERUSER  
COMPILING  
COMS-SYNC-RECOVERY  
COMS-USERCODE  
CONVERSION  
COPY-FILE  
COPY-PREFIX  
COPYFILE-PACK  
COPYFILE-USERCODE  
COUNT  
CUSTOMER

1 1 Pg=1 TA14021 Norm Rcv LT01

005577

## Help Screen

InterCom Terminal Emulation - [A40NTW - TA14021]

Spcfy back HELPSCREEN INQUIRY infra design

It is possible to SPCFY on a line.  
. In case of inquiry file-, item- or user-name, the specified name will be entered automatically on the master-screen.  
. In case of inquiry relationships, a continuation screen with more information about the specified item will be given, but: if the spcfy is given on columns 26-48 or 50-78, the specified file-, item- or user-name will be copied back to the current master screen (if possible).  
A specify on columns 79-80 gives the time-stamp of the corresponding relationship.

1 1 Pg=1 TA14021 Norm Rcv LT01

005578

### 42.2.2.2. Specify on FILE

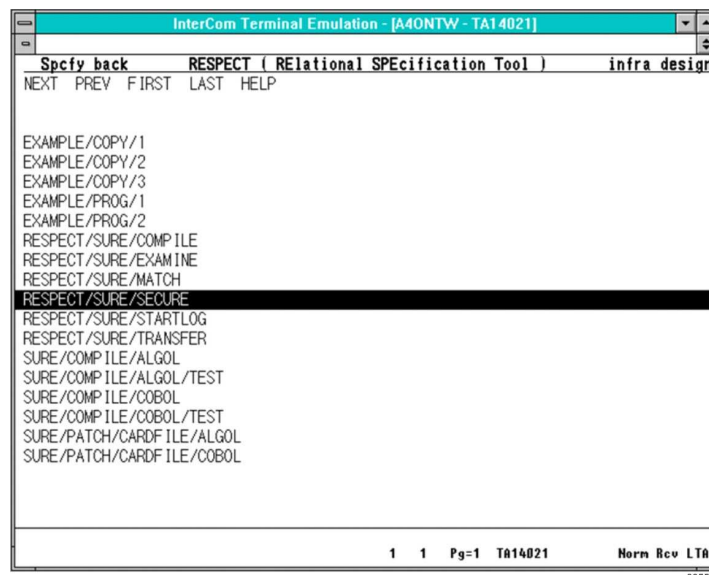
#### Function

Requests file names resembling an incorrectly entered file name. This function can be used to search for the correct file name or to browse through a list of file names.

#### Usage

- The function is available from all screens where a file name can be inquired on by using the function "INQ."
- The function can also be used if an error message indicates that a file is not present in the database.
- By <specify> on "FILE," a continuation screen is displayed with a maximum of 18 lines.
- File names are listed in alphabetical order. File name, input date, and input time are presented on each line. The line with the name most closely resembling the incorrectly entered file name is highlighted.
- The following functions are used to page through the continuation screens to find the correct file:
  - <Specify> on NEXT
  - <Specify> on PREV
  - <Specify> on FIRST (original) page
  - <Specify> on LAST if selected, the first (original) page is displayed.
- Once the correct name is found, it is possible to <specify> on this name. The previous screen will be presented. In some cases, the specified file name is entered automatically in the error-field.

#### Screen: Inquire Item Names



#### **42.2.2.3. Specify on TYPE**

Sometimes the online program requires an item to be of a specific type. If the entered item exists but is not of the correct type, the following error will be generated: "522, This ITEM is not of type <type>." When this error occurs, the TYPE button can be specified and a list of the valid items of the required type will be presented.

This function can be used to search for specific types by deliberately making an error. Note that the item must exist because this error has a higher priority than type verification. After the type error, all valid types will be shown by specifying the TYPE button.

The selected types may be browsed and certain screens allow the selection of a type by specifying on a line. This name is then returned in the error field.

#### **42.2.2.4. Specify on NEXT**

A response from the online program can comprise multiple pages of information. The NEXT button returns the next page of information.

#### **42.2.2.5. Specify on PREV**

A response from the online program can comprise multiple pages of information. The PREV button returns the previous page of information.

#### **42.2.2.6. Specify on FIRST**

A response from the online program can comprise multiple pages of information. The FIRST button returns the original (first) page of information.

#### **42.2.2.7. Specify on LAST**

A response from the online program can comprise multiple pages of information. The LAST button returns the last page of information. Note that "LAST" does not work for the <specify> item and <specify> file buttons.

#### **42.2.2.8. Specify on RELATION**

##### **Function**

Requests relations:

Show all relations of the current item name or file name in the current environment.

Attributes of files or RIS-id's are linked to these files (RIS-id's) using relations. Relations are records of the INFDB dataset DREL. Each DREL-record contains one relation and each relation corresponds with one attribute.

### Example

Consider a file EXAMPLE/COPY/1.

EXAMPLE/COPY/1 is written by Infra Design, so attribute AUTHOR is Infra Design.

EXAMPLE/COPY/1 has status production, so attribute STATUS is PRODUCTION.

The results are in the following relation.

FILE	EXAMPLE/COPY/1	AUTHOR	INFRA-DESIGN
FILE-CONTROL	EXAMPLE/COPY/1	STATUS	PRODUCTION

A relation is stored in the INFDB dataset DREL and consists of the following fields:

REL-GROUP	: The RIS-entity to which this relation belongs
REL-OWNER	: The owner of the relation
REL-CLASS	: The owner of the relation
REL-ASSET	: The asset or value of the relation
REL-VERSION	: The environment where this relation is valid

In the example above, the relation groups are FILE and FILE-CONTROL, indicating the owner of the relation is in this case a file name; the relation-OWNERS are in both cases EXAMPLE/COPY/1; the relation Classes are AUTHOR and STATUS; and the relation ASSETs are INFRA-DESIGN and PRODUCTION.

All relations that have the item or file as an asset or owner are displayed on the screen. Relations where the item or file is used as an asset are displayed in reverse order with the asset in the first column and the owner in the last column.

### Usage

- Inquire the desired item or current file with function "INQ."
- <Specify> on "RELATION" displays a continuation screen listing all the existing relations with the item or file.

### Usage on the Continuation Screen

- It is possible to <specify> on a relation line to obtain the following information:
  - The timestamp of the relation when the <specify> is performed at positions 79-80 of the line.
  - The relations of the item in column 3 when the <specify> is performed at positions 0-25 of the line.
- The following functions can be used to page through the continuation screens:
  - <Specify> on FIRST = first page
  - <Specify> on NEXT = next page

- <Specify> on PREV = previous page
- <Specify> on LAST = last page

Example of a “<specify> RELATION” output screen.

Specify back RESPECT ( Relational SPECification Tool ) infra design		
NEXT PREV FIRST LAST HELP		
EXAMPLE/COPY/1		VERSION: DEVELOP
FILE	AUTHOR	INFRA-DESIGN
FILE	FILE-TYPE	INCLUDE
FILE	FILE-IND	DMALGOLSYMBOL
FILE	MAINTENANCE	SIMON
FILE	STOR	96-01-04 20:28:24
FILE-CONTROL	ASSIGNED	SIMON
FILE-CONTROL	CHANGED	96-01-04 20:28:25
FILE-CONTROL	EXAMINE-STATUS	TO-EXAMINE
FILE-CONTROL	PREVIOUS	96-01-04 20:28:24
FILE-CONTROL	STATUS	PRODUCTION
FILE-CONTROL	TRANSFER-STATUS	TRANSFERRED
0 TYPE	RIS-GROUP	FILE
.. FILE-CONTROL	COPY-FILE	S/NP/DBP/NPDBP0171
.. FILE-CONTROL	COPY-FILE	S/NP/DBP/NPDBP0172
0 PROJECT	PROJECT	NP
0 PROJECT	SYSTEM	BASE

1 1 Pg=1 TCP\_49 Norm Loc LT81 005580

The “target” name is placed in the upper-left corner of the screen. All relations with this target as an owner or asset are displayed. If the target is the asset of the relation, then the line is highlighted.

### Example

In the example screen, the target name is EXAMPLE/COPY/1.

The current environment is displayed in the upper-right corner of the screen. All displayed relations are valid only in this environment, except those relations with a “0” (zero) at line position 3. These last relations are defined globally and are valid for all environments.

### Example

In the example screen, the current environment is DEVELOP, and the project and system relations are defined globally (0 at line position 3).

The first column shows the group of the relation. The second column shows the class of the relation. The third column shows the asset (if not highlighted) or the owner (if highlighted) of the relation.

### Example

The first mentioned relation in the example screen is the “author” relation. In this case, the group is FILE, the class is AUTHOR, and the asset is INFRA-DESIGN. The last four relations on the screen are highlighted, and in these cases, the last column represents the owner.

### 42.2.2.9. Specify on INFO

#### Function

Shows information on the current RIS-id or in the file that is available in the current environment (no update functionality).

It is possible that an item or file contains different types of information. An example is an item in RESYDE that contains "normal" information and "technical" information. A record example is a file in SURE that contains file documentation. Using <specify> on INFO presents the "normal" information.

All types of information are presented using the same functions that are described in this section.

Other types of information are

- RESYDE technical information
- HELP problem solution
- SURE customer information
- SURE problem information
- SURE problem tasks

Information is presented in the inquiry-mode by default. When an <xmit> is given on an information screen in the inquiry-mode, the program will return to the previous (master) screen without updating the information. When the information function is entered in the update-mode (this can only be done with a specific function) and an <xmit> is given on the information screen, all information from the home position to the cursor position will be updated and the next page will be presented.

Information is presented in the current system language by default. However, with SURE and RESYDE, it is possible to override the system language by a temporary session language.

This is achieved by using the RESYDE functions INQINFO on INQTECH with one or more language-codes.

Examples of language-codes are

E: English

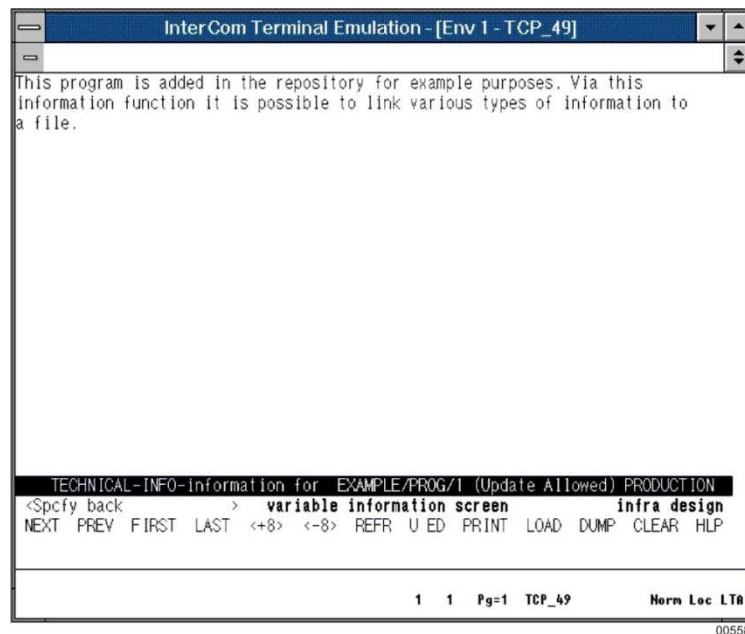
N: Dutch (Nederlands)

D: German (Deutsch)

In RESYDE, it is possible to execute some commands with one or more language codes.

INQINFO E,N	Show the information for this file in English and Dutch
INQINFO N,E	Show the information for this file in Dutch and English
INQINFO All	Show the information for this file in all languages

### Screen: Maintain Documentation



The first 20 lines of the current information are displayed on the continuation screen. If the INFO function is entered in the UPDATE mode, there are three ways to maintain the information.

1. Use the function '<specify> on U ED' to start the Unisys System/Editor to edit the information. After using the editor, the new text appears on the information screen. You should use this method if the information is longer than one page.
2. Dump the information into a file by specifying on DUMP.
  - Edit the information through CANDE.
  - Load the information back into the repository by specifying on LOAD. Once you specify the DUMP or LOAD, a continuation screen is displayed where the file name is entered.
3. Type the new information directly on the screen and position within the information using specify on NEXT, PREV, FIRST, LAST, +8, -8, or REFR. Information can be entered on lines 1-20. It is possible to enter multiple pages of information: an <xmit> will update the present page up to the cursor position in the repository. The last four updated lines will be displayed again at the top of the screen, after which new information can be entered.

+<number of lines>	For example, 10 lines forward is +10
-<number of lines>	For example, 10 lines backward is -10
NEXT+ or N+	20 lines (1 page) forward
NEXT- or N-	20 lines (1 page) backward
PA	Show the first 20 lines (1 page)
PA END	Show the last 20 lines (1 page)
SAM or REFRESH	Refresh the page

You can use the specify function at the bottom of the screen.

<specify> on NEXT	20 lines (1 page) forward
<specify> on PREV	20 lines (1 page) backward
<specify> on FIRST	Show the first page
<specify> on LAST	Show the last page
<specify> on <+8>	8 lines forward
<specify> on <-8>	8 lines backwards
<specify> on REFR	Refresh the page
<specify> on U ED	start the Unisys System/Editor to edit the documentation using that utility
<specify> on PRINT	Print the documentation
<specify> on LOAD	Load the documentation from a file into the repository
<specify> on DUMP	Dump the documentation from the repository into a file
<specify> on CLEAR	Clear the documentation

You can <specify> on each word of the documentation. If a word contains the "normal" information, the information will be presented; otherwise, an error message is given.

### Example

```
item "SURE" contains the information:    "a product of Infra Design".
item "Infra" contains the information:   "Infra Design
                                         Netherlands          ".
```

When <specify> is given on the word "Infra" (part of the information for item SURE), the information for item Infra will be presented.



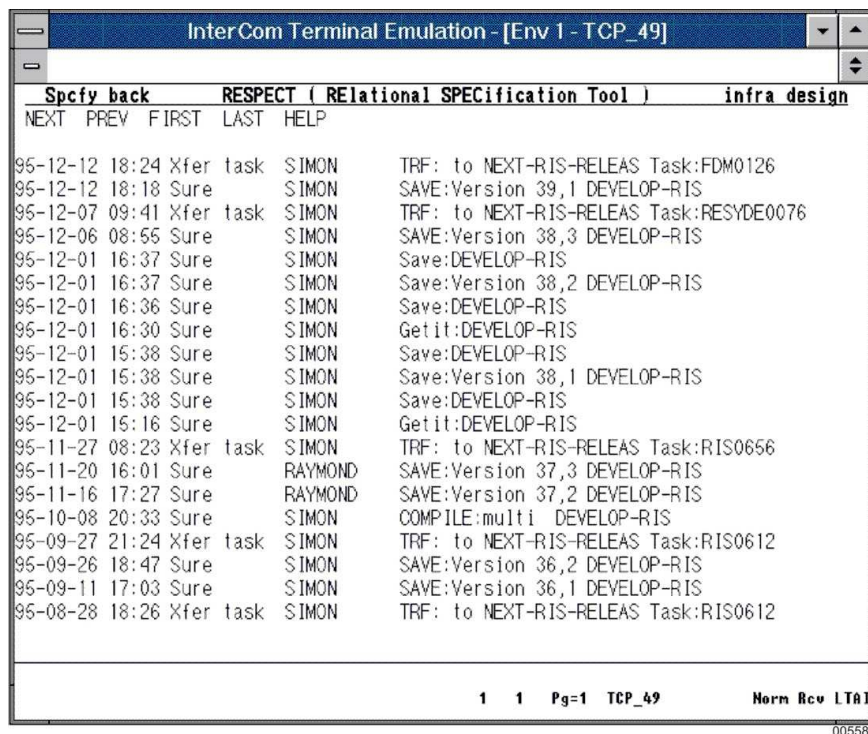
### 42.2.2.10. Specify on LOG

#### Function

Shows LOG information about the current RIS-id or source.

Each time a source or RIS-id is updated, the timestamp of this update and the user who made the update is logged. There is only one log available for all environments. This makes the flow of the adaptations and transfers to other environments of this source or RIS-id visible.

#### Screen: Show Log



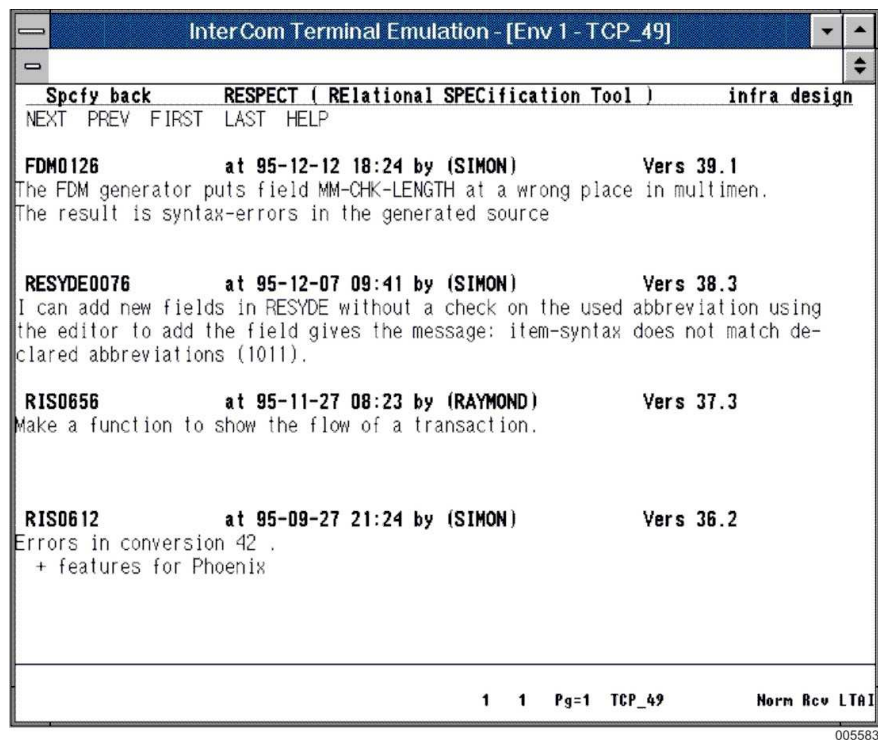
### 42.2.2.11.Specify on HIST

#### Function

Shows the history of the current RIS-id or source.

A source or RIS-id can only be updated if a task is linked to that source or RIS-id. The adaptations are transferred to the next environment by transferring the task to that environment. If the task is transferred to a "solved" environment, the status of that task will be changed to SOLVED, and the task will be added to the history of the sources and RIS-id's that were adapted due to the task. A specify on HIST shows all tasks why the source or RIS-id was adapted in the past.

Screen: Show History

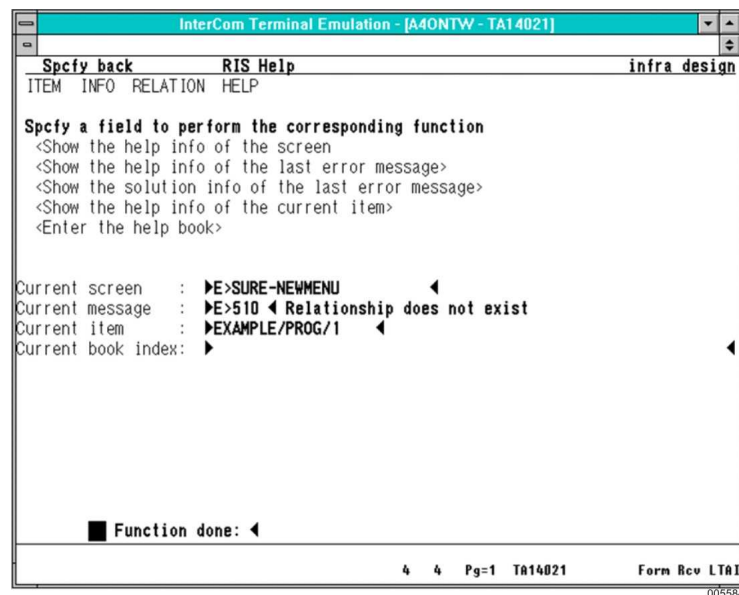


You should specify the task names to get the detailed information about that task.

42.2.2.12. Specify on HELP

The HELP function gives the user extra error message information, solutions to errors, help information belonging to online functions, and access to the documentation.

When the HELP function is selected, a help screen is presented. The current error message, the current screen, the current item, and the current BOOK chapter are automatically entered on the help screen. It is possible to overwrite this current information and to choose another error message, screen, item, or chapter.

**Screen: On-line Help**

The following help information is available:

- The screen: provides information about the commands that can be used on the current screen.
- The last (current) error-message: provides extended information about the error situation (why the error occurred).
- The last error-message: provides information on resolving (or correcting) the error.
- Help info on the current item: provides "normal" information about the item.
- Enter the help book.
- The documentation is loaded in the repository. It is possible to read the on-line documentation using this function.

The current book index gives the chapter of the documentation where the book is entered. First a tree-structure of all chapters and paragraphs is given. With the function <specify>, it is possible to select a paragraph or to move to a lower chapter level.

When a chapter is chosen and presented on the screen, it is possible to <specify> on each word of this chapter.

- If the specified word is mentioned in the index of the documentation, then all chapters where this word is used will be displayed. Each chapter can be chosen with the specify key.
- If the specified word is not mentioned in the index, but contains the "normal" information, this information will be presented.

You can overwrite the "current" keys on the help-screen by your own keys.

### 42.2.2.13. Specify on Tasks

#### Function

Shows an overview of tasks linked to the current RIS-id or source.

A specify on TASKS shows all the tasks for which an adaptation has been made to the source or RIS-id for the current environment.

A specify on one of these task names shows detailed information about that task.

### 42.2.2.14. Specify on PRINT

#### Function

Prints information that is selected by SURE on a remote printer.

Various screens contain a print button. A specify on such a button will result in creating a backup file. A continuation screen will be presented where a remote printer name can be entered. The backup file will then be printed on that remote printer. If "NONE" is entered instead of a printer name, then the backup files will be created, but the files will not be printed.

### 42.2.3. Status Information

It is possible to keep multiple versions of sources or RIS-id's in the repository. Each environment that is defined in the repository can contain a different version of a source or RIS-id. Therefore, it is important to know what the current environment is; otherwise, one might look into a wrong version of a file or RIS-id.

Definition of environments will result in a standard layout of the heading for most of the SURE online screens. The top of these screens contains a standard layout with the following attributes as shown in the example.

#### **Curr.Environment**

The current assigned environment for the user.

#### **Current Task**

All updated definitions or sources during a SURE session will be linked to the current task. If there is no current task available, then it is not possible to update definitions or sources.

Updates to definitions or sources are only possible in the environment that is defined for the current task.

An on-line SURE-session will start automatically in the user's default environment, unless the program is started with a task string to enforce that the session begins in the environment that is indicated with the task string.

A specify on the name of the current task gives all relations of that task along with the RIS definitions and sources that were changed due to this task.

**Environment and Status Summary**

On the top right portion of the screen, an environment summary is given. This environment summary is formed from each first character of every defined environment, from the lowest hierarchy (development) up to the highest hierarchy (production). It is possible to switch temporarily from the current environment by specifying on one of the characters of the environment summary. As a result, the definition can be viewed at the specified environment or a task can be chosen defined for this environment. Be aware that changes can only be made in the environment assigned to the current task. If the environment is changed this way, it will be set to default when the current function is left through <specify back>.

**Example**

Consider a repository with three environments: DEVELOP, ACCEPT, and PRODUCTION. This results in an environment summary DAP. Here, D stands for DEVELOP, A for ACCEPT, and P for PRODUCTION. If a specify is given on the A of the DAP, then the current environment is changed temporarily to ACCEPT. From that moment, all actions will be made for ACCEPT environment.

The temporarily chosen current environment remains active when a "deeper nested" on line function is activated, but the original current environment will be reactivated if the current function is left though specify back.

Notice that the environment summary is dependent on the system of the current file or RIS-id. If a file or RIS-ID is current, the environments that are excluded for the system of the file or RIS-id will not be mentioned in the summary.

**Example**

Consider the same repository with environments DEVELOP, ACCEPT, and PRODUCTION.

File P/1 belongs to system PSYS and environment ACCEPT is excluded for this system.

If file P/1 is inquired, the environment summary will be "DP" instead of "DAP."

For each environment, the status of the definition or source is given through a specific character just below the first character of the environment. The combination of these specific characters for all environments is called "status summary." The possible values of these specific characters are given below.

## Terminal Emulation Interface

- \* This definition is different from the definition at the next higher environment (closer to production)
- space There is no definition present for the environment.
- . There is a definition for this environment, which is equal to the definition at the next higher environment (closer to production).
- Q A quick fix has been made for the definition on this environment, but it has not been transferred to the next higher environment (closer to production).
- q A quick fix has been made for the definition on this environment and the definition is transferred to the next higher environment.
- R The definition is removed from this environment; however, it still exists in the higher environments.
- G The definition or source is generated.
- This definition is not available in this environment, but it is available in a higher environment.

The screenshot displays the 'InterCom Terminal Emulation - [Env 1 - TCP\_49]' window. The main menu includes 'Specfy back', 'SURE (Software Update REtrieval) (c) 1986 infra design by', and a list of tabs: 'ITEM', 'INFO', 'RELATION', 'HISTORY', 'RUNINFO', 'TASKS', 'ERRORS', 'LOG', 'FILE', 'TYPE', and 'HELP'. The 'TASKS' tab is selected, showing a table with columns 'Environments' and 'Status'. The table lists 'DEVELOP-RIS' with a status of 'DAP' and '.\* DEVELOP-RIS'. Below this, the 'Curr.Environment' is 'DEVELOP-RIS' and the 'Current Task' is 'RIS0666', which was 'Changed' on '95-11-06 16:40:06'. The 'Copy/Object' section shows 'Author' as 'RESPECT', 'System' as 'SURE', 'Project' as 'ONLINE', 'Function' as 'ONLINE', and 'File type' as 'ONLINE'. The 'DMALGOLSYMBOL' section shows '(FileVersion 113.2)'. The 'Object usercode' section shows 'Compiled by Sure' and 'Listing'. The 'Processor' section shows 'Mixnr', 'Core', and 'Eol'. The 'Last prod.run' section shows 'with object crea.date'. The 'Function done' section shows 'INQ: RESPECT/SURE'. The bottom status bar shows '1 1 Pg=1 TCP\_49 Norm Loc LTRI' and a small number '605585'.

### Status Indication

The interpretation of the status summary for a single environment can be difficult. You cannot say "this definition is in MAINTENANCE" or "it is in PRODUCTION." As the example screen shows, the RESPECT/SURE file is changed in the DEVELOP-RIS environment and resides in the ACCEPT-RIS environment, equal to the file in PRODUCTION. The "status summary" gives an overview of the status of the source or definition for all environments. Right of the "status summary" is a status indication that gives the exact status of the source or definition in the current environment.

The possible status indications are as given below.

<Environment>	Defines the highest (= nearest to production) environment that contains a version of this definition or source that is equal to the version in the current environment.
	In the example screen:
	The status of RESPECT/SURE in the environment      develop-ris:      DEVELOP-RIS
	The status of RESPECT/SURE in the environment      accept:      PRODUCTION
	The status of RESPECT/SURE in the environment      production:      PRODUCTION
GENERATED	If the definition or source is generated in the current environment.
REMOVED	If the definition or source is removed in the current environment.

#### 42.2.4. Current Task

Adaptation to sources or RIS definitions can be made only if the programmer is assigned to a "current task." This current task is presented on many screens, and specify on the current task name will give detailed information about this task.

#### 42.2.5. Function Specific Part: Line Three to Twenty-Three

The function "specific part" is different for each function. The following general methods may be used as a guide:

- RIS menu specify selection  
The RIS menu only allows commands to be selected using the Specify key. All text on this screen represents a function that can be selected by the Specify function.
- Other functions  
The majority of other screens are 'forms mode' screens. The values of the items must be entered between forms mode delimiters. These values must be sent to the mainframe by the Transmit key.

Forms mode screens often contain options that can be selected by the specify key. These options are easily recognizable because they are enclosed in angular brackets, < >.

For example, <option which can be specified>.

If the option is set, it will be shown in reverse video.

**Note:** *If the forms mode and specify options are both present in a screen, the Transmit function as well as the Specify function must be used. The Transmit function has higher priority to make variables current in the program.*

For many fields, it is possible to obtain extra information about the name that is placed in the field by giving a specify on that field. The following extra information is available:

- If the field contains a file name, and this file is loaded in the repository, then the contents of the file is shown (like the LIST command in SURE).
- If the field contains a format name, and this format is loaded in the repository, then the technical layout of that format is shown (like function 19 in the format editor).
- If the field contains a transaction code, then the flow of that transaction is given (like function FLOW on the SURE screen).
- If the field contains an attribute of the current file or RIS-id, then all attributes of that type are shown.

### Example

If a specify is given on the author-field in SURE, then all available authors are presented.

- If no extra information is available for a field, then the error message "You can not specify on this place" is returned.

## 42.2.6. Error Field: Last Line

All errors are indicated by an error field on the bottom line of the screen. The field that caused the error is shown in REVERSE video and the cursor will be positioned in that field.

Each error message has a unique number that is also displayed on line 24 (just before the error text). The number is used in the key of the error message. This error message key can be used in the HELP function (specify on HELP) to obtain extended information about the error.

Consider the following error message:

```
500  You are not allowed to use this function
```

In this case, the error number is 500 and the corresponding error key is E>500. The error key is built up as follows:

- First position is the language code (in this case "E" for English).
- Second position is always a ">" (used as a separator).
- Positions 3 to 6: a unique error number.



If the error number is 500 and the language code is "E," then message E>500 will be shown. If the language code is "N" (Nederlands) and message N>500 are available in the system, the message N>500 will be shown:

```
500  U mag deze functie niet gebruiken
```

If key "E>500" is entered on the help screen in field "current message," then it is possible to obtain the extended help information and solution info of this message, by specifying on the corresponding fields (Refer to "42.1.2.12 Specify on HELP" for more information).

### 42.2.7. Ctrl Codes

Some general commands may be entered using control codes. A control code consists of the following actions:

- Press the Ctrl key.
- Press a two-digit numeric code.
- Press the Transmit key.

The available control codes and their meanings are listed below.

Ctrl 00 Transmit	Refresh screen
Ctrl 01 .. 20 Transmit	Go to page 1 .. 20 of a multi-page screen
Ctrl 21 Transmit	Go to the next page of multi-page screen
Ctrl 22 Transmit	Go to the previous page of multi-page screen
Ctrl 28 Transmit	Last screen of multiple pages
Ctrl 29 Transmit	Multiple page exit
Ctrl 30 Transmit	First page of a multiple information screen
Ctrl 31 Transmit	Next page of a multiple information screen
Ctrl 32 Transmit	Previous page of a multiple information screen
Ctrl 33 Transmit	Last page of a multiple information screen
Ctrl 37 Transmit	Show myself.jobnumber, myself.tasknumber, and myself.usercode
Ctrl 38 Transmit	Show repository title + library-mixnumber
Ctrl 39 Transmit	Show current screen title
Ctrl 40 Transmit	Return to previous menu
Ctrl 88 Transmit	Go to the SURE home menu
Ctrl 92 Transmit	Refresh screen
Ctrl 94 Transmit	Print the current screen

Ctrl 95 Transmit	Print the current screen in uppercase
Ctrl 96 Transmit	Dump the current screen in the SCREEN/DUMP/<mix number> file
Ctrl 99 Transmit	Help screen

### 42.2.8. GO Command

The GO command is available to jump directly from one SURE function to another SURE function, without passing through the SURE menu screen.

#### Syntax

— GO —<destination function>—  
      |  
      | END

GO <destination function> is available only in the SURE menu program.

GO END is in all online programs available.

The GO command can be used on every online screen from the home position. If the screen contains form delimiters for various fields, then the GO command can be entered using multiple fields.

#### Example

Consider a screen with the following fields:

```
command  [      ]  
File name [      ]
```

Then, the GO command can be entered as follows:

```
command  [GO R]  
File name [ESYDE      ]
```

As a result, the RESYDE screen will be presented.

### 42.2.9. Printing Information

Various online functions are available that result in the creation of a backup file. If a backup file is going to be created, then an intermediate screen will appear, where the name of the remote printer (the printer destination) can be defined. If a default remote printer is declared for the user, then that printer name is automatically pre-filled by SURE. SURE waits for 15 seconds before it accepts the default printer name. In the mean time, the user can enter another printer name.

If "NONE" is entered instead of a remote printer name, then the backup file will be created, but it will not be printed.

## Section 43

# Windows Interface

### 43.1. Logon

The logon screen appears once you start the SURE application. The file/logon menu option shows the following screen.

The screenshot shows the SURE Logon window. The title bar says "Logon". The header area has the SURE logo and the text "Task Tracking, Source Management, Application Deployment ...". The main area is divided into several sections. On the left, there's a "User" section with input fields for "User", "Password", and "AccessCode", and a checkbox for "Save Password and AccessCode". To the right of this is a column of buttons: "Logon", "Change Password", "Start SURE Explorer", and "Cancel". Below the "User" section is the "Current Environment" section with dropdown menus for "Environment", "Task", and "Language", and labels for "Host", "Pack", and "Printer". To the right of this is a section titled "modify SURE explorer layout according to role" which contains a list of roles with radio buttons and dropdown menus for each: "None", "Development CP/NX", "Development PC/UNIX", "Help Desk", "Operations", "Project leader", "Security Administrator", "Support", "Task routing", and "Test". The dropdown menus are labeled "in environment" or "for team".

To logon to the system:

- Enter the values for the "User," "Password," and "Accesscode" (optional) fields.
- Press **Logon**.

SURE returns the current environment parameters: task, language, host, pack, and printer.

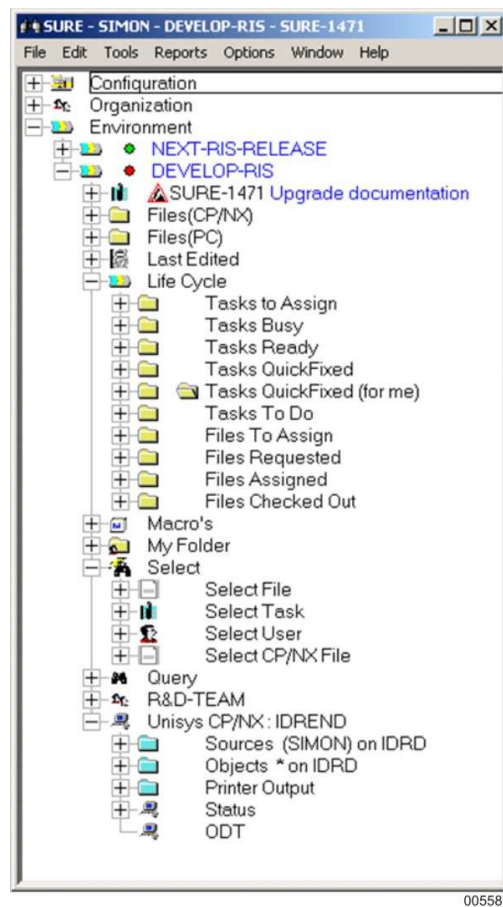
- Press **Start SURE Explorer** to start the SURE Explorer.

You can customize the SURE Explorer according to your role within the organization. This customized explorer interface simplifies the user interface by reducing the number of choices and folders. If this user role is defined in the user definition, then it is not selectable in this window. In other words, the user is enforced to work in this mode. Otherwise, the user can select his role.

## 43.2.SURE Explorer

### 43.2.1. Explorer Folders

The SURE Explorer contains many folders. The example window shows a repository with three environments and all main folders opened for the DEVELOP environment. Each environment will show the same set of folders when expanded.



#### Configuration

This folder defines the options and parameters for SURE. This includes allowed drop-down box values, task-types, and groups.

#### Organization

This folder defines the users, securities, and teams.

### **User**

This folder includes a list of defined users in the system.

### **User Team**

This folder includes a list of user teams defined in the system.

### **Employee Function**

This folder includes a list of employee functions defined in the system.

### **Environment**

The environment is the top-level node for all configured environments in the repository. The popup menu for this folder allows setting of Global Options for the SURE system and allows creating new environments.

<**NEXT-RIS-RELEASE**> and <**DEVELOP-RIS**> are configured environments.

### **Files (CP/NX)**

This folder shows the ClearPath Enterprise Server files present in the SURE repository of the DEVELOP-RIS environment.

### **Files (PC)**

This folder shows the PC files present in the SURE repository environment DEVELOP-RIS.

### **Last Edited**

This folder shows the names of the edited files. To configure the number of entries:

1. Click **Toolbar**.
2. Click **Options**.
3. Click **Number Entries**.
4. Click **Last Edited**.

### **Life Cycle**

This folder shows the files or tasks in a specific workflow status. The different folders in the Life Cycle folder can be considered as a work queue.

### **Tasks to Assign**

This folder lists all the tasks that have not been assigned yet. Therefore, the tasks in this list are entered but are not yet assigned to a user, a team, or an employee-function.

A task manager or project leader inspects this list and assigns the tasks.

### **Tasks Busy**

This folder lists all the tasks that are assigned to one or more developers. If a task is transferred, it moves to the BUSY list in the destination environment.

Acceptance or quality assurance employees inspect this queue and perform test scripts.

### **Tasks Ready**

A task can be marked ready and thereafter it will be present in this list.

A task manager, project leader, or configuration controller inspects this queue and transfer these tasks to a next environment.

### **Tasks QuickFixed**

A quick fix task or a partial reprocessed task is present in this list.

A programmer inspects this queue and applies the quick fix changes.

### **Task To Do**

This folder lists all the tasks that are in the To Do list.

### **Task Quick Fixed**

This folder lists all the tasks that are assigned to the user and must be reprocessed due to a quick fix.

### **Files To Assign**

This folder lists all the files that are requested but are not yet assigned. This folder is used only if the request method is enforced by the definition of the authorization scheme.

A technical manager or project leader inspects this list and approves the request.

### **Files Requested**

This folder lists all the requested files that are not yet assigned.

### **Files Assigned**

This folder lists all the files that are assigned but not checked out.

### **Files Checked Out**

This folder lists all the checked out files.

### **Macro's**

This is a list of defined macros in the system.

### Select

By default, the explorer shows information in a tree view. This means that you can open the nodes until you reach the actual name of a file or task. In some cases, this is not the fastest way to access information. Therefore, the select facility offers the possibility to select objects by name.

#### Select File

Select a file from the repository.

#### Select Task

Select a task.

#### Select CP/NX file

Select a ClearPath server file in the work-environment.

#### Select User

Select a user.

### Query

This folder allows initiation of the query facility of SURE.

#### <host-name> (IDREND on the Example Screen)

This is the BNA name of the ClearPath Enterprise Server where the SURE server system is running. From this folder, you can access this server for actual work on this ClearPath server.

#### Sources <usercode> ON <packname>

This folder contains the files on the ClearPath server in your work-environment. Note that the work-environment is configured on the system level.

#### Objects <usercode> ON <packname>

This folder contains the files on the ClearPath server in your object environment. Note that the work-environment is configured on the system level.

#### Printer Output

This folder contains the print files produced on the ClearPath server for the user listed in the work-environment.

#### Status

This folder shows the mix status for the current work usercode.

#### ODT

This folder allows performing ODT commands on the ClearPath server system.

### 43.2.2. Hiding Main Folders

The main folders described in the explorer folders might be hidden so that the user interface can be individually customized. The **hide** command is accessible through the popup menu for each folder that can be hidden. This command does not directly adapt the user interface.

To see the hidden folders, do the following:

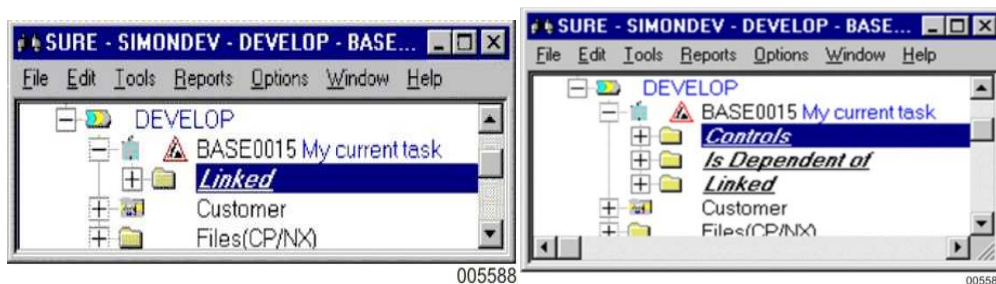
1. Click **Options**.
2. Select **Show hidden folders**.

Setting this option allows you to reset the hide indication for a folder.

### 43.2.3. Expanding Objects

Many of the objects in the explorer can be expanded with sub-folders, as shown by the + (plus sign) or collapsed as shown by the - (minus sign). The sub-folders that are shown for a particular object are content-sensitive. That means, if there is no information available for a sub-folder, then that sub-folder is not shown.

The following two example screens show the BASE0015 task, both with different detailed folders. This is because the status of the task has been changed between the two snapshots of the window.



The expansion is performed based on the type of object. There is no limit on the level of the nested expansions and on the check on recursion.

This next list is a summary of the types of objects and the possible sub-folders.

#### Environment

Each environment shows the standard folders as explained in Section 43.2.1, "Explorer Folders." The hide facility allows customization of this standard list.

#### Directory

A directory will show sub-directories and thereafter the actual files in a directory. For ClearPath server files, a directory can also be a file, and in this situation, two entries are present: the directory and the file.



**File (Repository)**

A file (repository) object is the name of a file in the SURE repository. This can be either a ClearPath server file or a PC file.

**References**

This is a list of all the objects referenced by this file. Which references are listed here may be configured through the Configuration/References folder.

**Used By**

This is a list of all the objects that reference this object. Therefore, this is the reversed reference list.

**Delta Files**

This is a list of all the delta files present for this file. The environment definition contains the archive period for these delta files.

**Task**

This folder is present if there are actual tasks or historical tasks linked to this file.

**Actual**

This is a list of tasks currently linked to this file in this environment. Therefore, this folder is present only if the file is or has been requested.

**History**

These are the historical tasks linked to this file. This is a logical log for the reason of the changes for this file.

**Task**

A Task object is a name of a task in the SURE repository.

**Controls**

This is a list of tasks that are controlled by this task. Therefore, these tasks are all dependent tasks.

**Is Dependent On**

This is a list of controlling tasks for this task. Therefore, this task is dependent on the list of tasks.

**Linked**

The objects (mostly files) that are currently linked to this task in this environment.

**History**

The objects (mostly files) that were once linked to this task and which are now in the task-history.

### **System**

This is a defined system name or an application system name.

### **Project**

This is a list of projects defined for this system. The system name is included in this list because project definitions may be defined for this system.

### **Files**

This is a list of files related to this project. So, instead of file names organized by physical name as shown in the directory folders, this folder can show file names based on logical organization.

### **User**

This is a defined user in the SURE repository.

### **Team**

This is a list of teams in which this user is configured.

### **Employee Function**

This is a list of employee-functions that are configured for this user.

### **Projects**

This is a list of projects for which this user is allowed to work.

### **Files Checked Out**

This is a list of files that are checked out by this user.

### **Tasks To Do**

The tasks that are in the To Do list for this user.

### **Printer Output**

This is the printer output present in the print request queue on the ClearPath server for this user. Through this folder, you can view printer output for any user on the ClearPath server.

### **Status**

This folder allows you to query the mix status for this user.

### **Employee Function**

The defined employee-functions in the SURE repository.

#### **Users**

A list of users with the selected employee-function.

#### **Team**

The defined teams in the SURE repository.

#### **Users**

A list of users that are in the selected team.

### **Print Output**

A list of current print requests for a user.

#### **Backup Files**

The backup files in a print request. These files can be downloaded and opened in a word processor.

### **File (CP/NX)**

The files at ClearPath Enterprise Servers.

#### **Status**

The mix status for users on ClearPath Enterprise Servers.

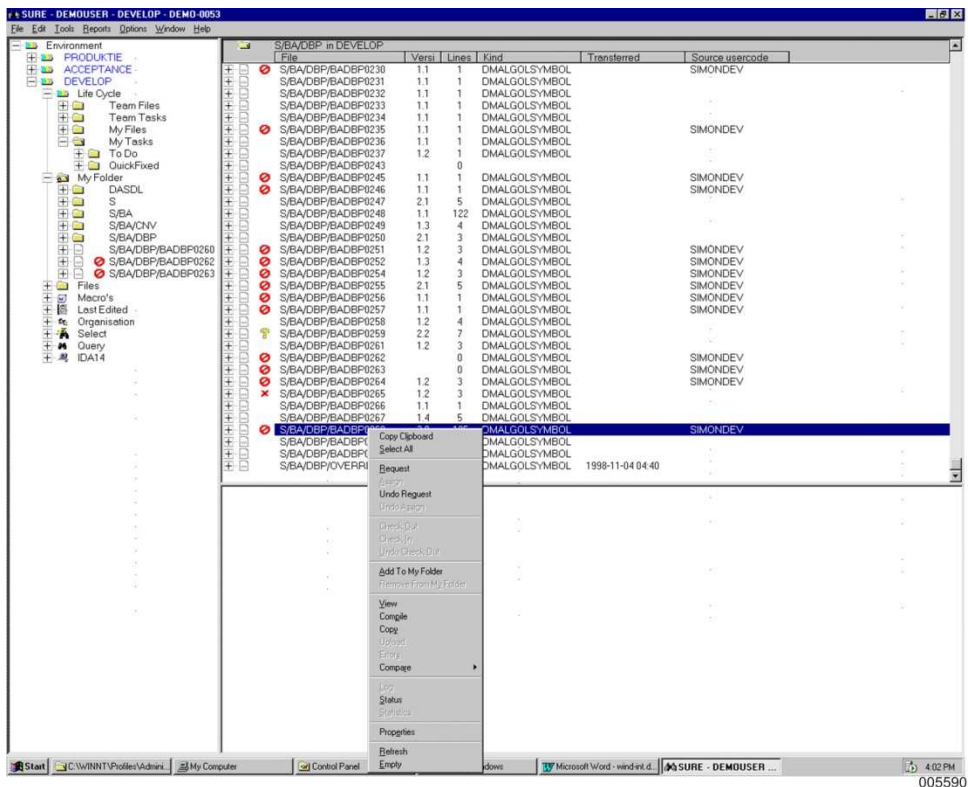
#### **ODT**

An entry that allows ODT commands using the popup menu.

### 43.2.4.Context-Sensitive Popup Menus

The SURE Explorer shows a different context-sensitive popup menu for the different object types. The popup menu of a file object differs from the popup menu for a user object. However, there are many common entries in the popup menus.

To open the popup menu, click **F1** for a selected object or right-click an object.



The **properties** entry in the popup menu opens a dialog with the properties of the selected object. The bottom line buttons might contain additional commands. This is the case for the file, user, and task objects. Most other objects include only a modify button allowing changing of the shown properties.



The dialogs shown allow a change if the fields are enabled. If the fields are in grey, the modify button must be pushed, which will start a new dialog with the adaptable fields enabled.

The security in the SURE repository may prohibit certain actions. However, the information is always shown to the user.

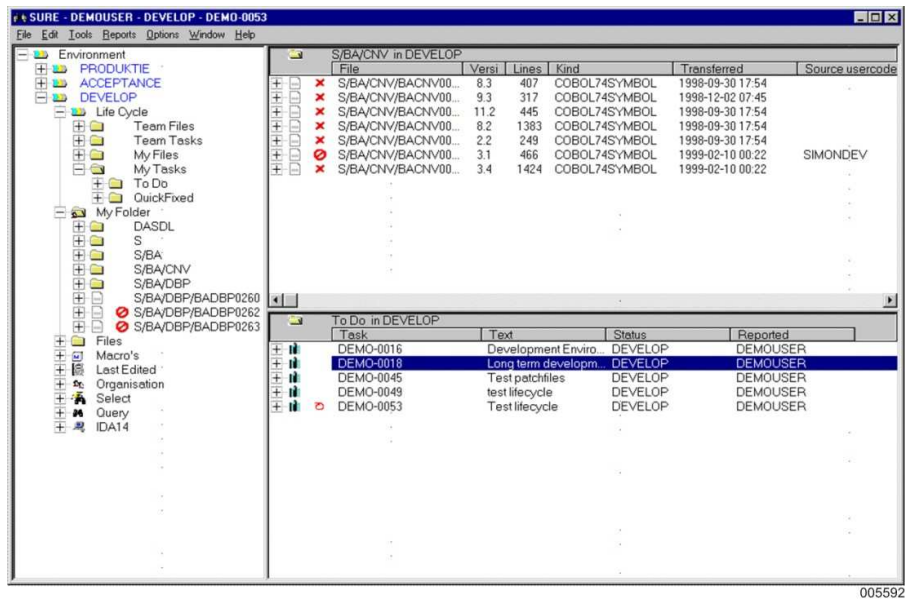
### 43.2.5.Windows Layout

The SURE Explorer consists of a left window with a list of the defined environments. The current environment is opened. To check the right side window entries, perform the following steps:

1. Click **Window**.
2. Click **New**.
3. Click **toolbar**.
4. Click **Window**.
5. Click **Close**.

The user can resize the windows and drop any folder on a right window.

## Windows Interface



The example shows the SURE Explorer with three environments listed in the left window: PRODUKTIE, ACCEPTANCE, and DEVELOP. On the top-right window, the S/BA/CNV folder has been dropped. On the bottom-right window, the To Do list is dropped.

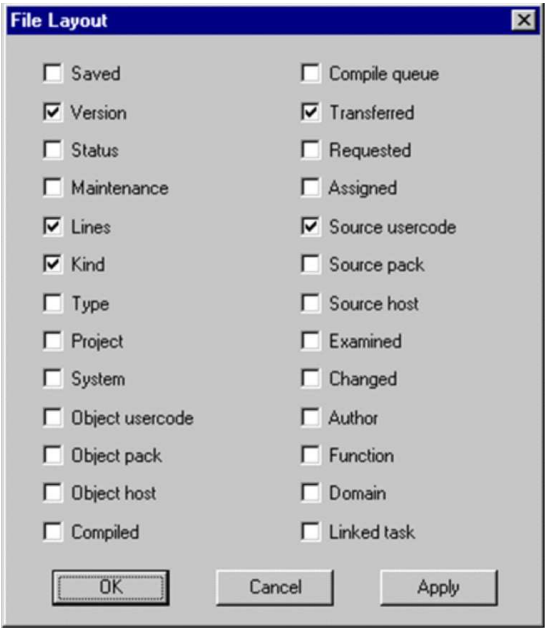
You can drop on an empty window.

You can drop on a non-empty window by pressing the **Ctrl** key.

A right window contains attributes for the type of object dropped. To define the attributes, perform the following steps:

1. Click **Menubar**.
2. Click **Options**.
3. Click **Attributes**.
4. Click **<ObjectType>**.

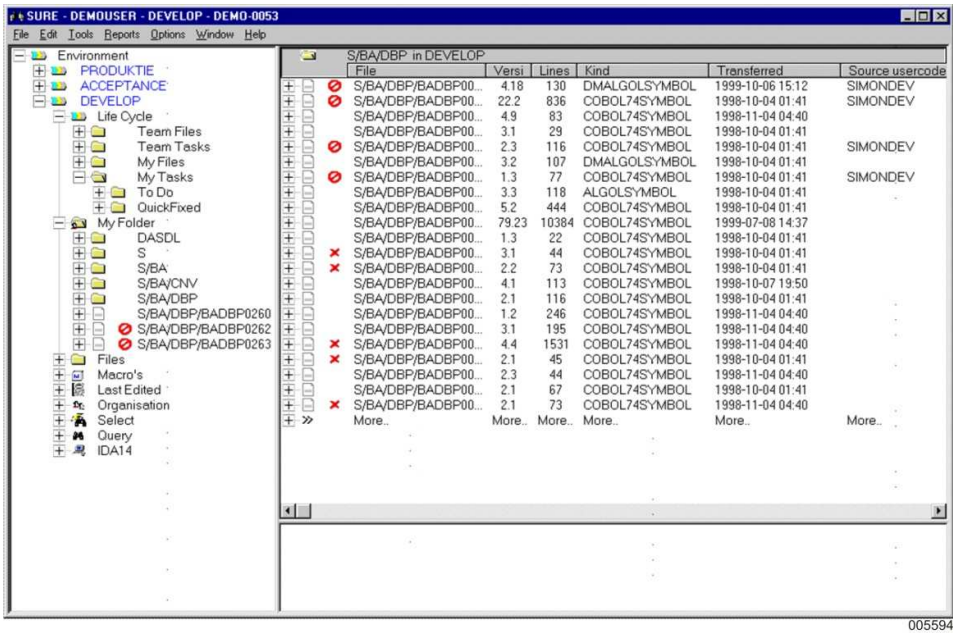
The Apply button will redo the query for the active screen and apply the changes. In the right window, you can sort the columns by pressing the column header.



005593

The queries on ClearPath Enterprise Servers may result in a “more” indication if the result contains more entries than the defined entries as shown in the next screen. The number of entries, which should be returned by ClearPath Enterprise Servers, can be configured per <ObjectType> through Options/Number Entries/<ObjectType>. Note that setting this number to “Maximum” may influence the performance because all entries will be retrieved from ClearPath Enterprise Servers before completing the function.

Pressing <<More>> retrieves the next browse-page from the ClearPath server.



005594

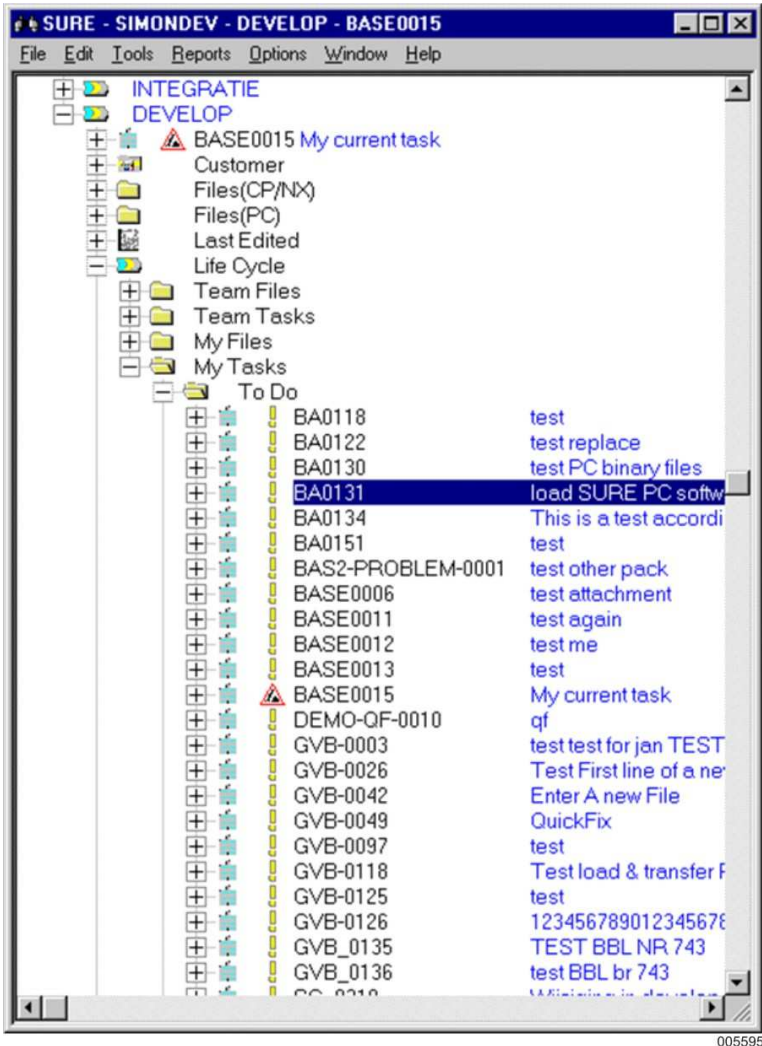
43.2.6.Refresh Strategy

The SURE system is a true client and server implementation that may even be run over the Internet with TCP/IP as a transport protocol. Therefore, an optimized refresh strategy has been chosen.

The “expand” and “collapse” functions cache the information in the GUI application. Therefore, expansion of a previously collapsed structure does not need to access the server for displaying information. This has the following side effect: changed information on the server is not automatically refreshed on the clients.

A “manual refresh” enforces execution of the server request, therefore updating the information in the GUI. Manual refresh is performed by selecting an entry in a list that needs to be refreshed, and thereafter pressing the **F5**.

In the next example, the To Do list is refreshed. Select the BA0131 entry, and press **F5** to refresh the list with the selected entry.





There is a difference in the refreshing strategy for the left window and the right window. If a function is performed in the GUI, it has an effect on one of the workflow lists (a list from the life cycle folder). If the affected list has been dragged to a right window, then the list is automatically refreshed. So workflow lists dragged in a right window will reflect the actual situation if a function is performed on the same workstation.

### **43.2.7. Clipboard Data**

The popup menus contain two options related to the clipboard.

The right windows contain a **Select All** choice that allows selection of all the objects. This includes execution of the "more" function until all the objects are loaded into the GUI. The selection of all the objects is disabled for the left window because it often contains different object types. To execute the select all function, select a single entry, invoke the popup menu, and then press **Select All**.

The **Copy Clipboard** function copies all the selected entries into the clipboard. For selected entries in the left hand window, it only copies the names. However, for selected entries in the right hand window, it copies the object names including the attributes. For this reason, the right hand windows can be loaded into an excel or a database system such as access.

### **43.2.8. Tools**

The Tools menu shows a number of SURE-related tools that can be used. These tools include the find, replace, and query facilities. Second, it can access the compile interface for the SURE repository. For ClearPath Enterprise Servers and the PC, separate sub-menus are available with applicable functions.

### **43.2.9. Queues**

Through the File menu, a number of specific SURE queues are accessible. Depending on the configuration, these queues can be persistent or just valid within a session.

#### **Compile**

This queue contains the history of remote compilations started through the SURE GUI.

#### **FileXfer**

This queue contains the outstanding and completed file transfers with ClearPath Enterprise Servers. These file transfers are mostly initiated by the GUI because of specific actions.

#### **E-mail**

This queue contains the e-mail queue maintained in the SURE repository.

### 43.3. Customizing the SURE Explorer

This chapter defines the specific actions that need to be taken for customizing the user interface. Customization is performed by setting the options in the configuration (AW\_OBJ.INI) file. That can be the private configuration file (on the user's PC) or the shared configuration file (on the server).

Refer to Section 8, "Installation," for information on private and shared installations.

#### 43.3.1. Help Menu

The SURE help menu may be extended with a number of specific entries. The entry [SUREHELP] in the configuration file may contain a number of entries in the form <file>=<text>. The text is shown in the menu text, and the file is started by the SURE system when choosing the option.

```
[SUREHELP]
TASK.HLP=Task Repository
FILE.HLP=File Repository
```



### 43.3.2. Report Menu

The SURE report menu may be extended with a number of specific entries. The entry [SUREREPORT] in the AW\_OBJ.INI configuration file may contain a number of report-entries.

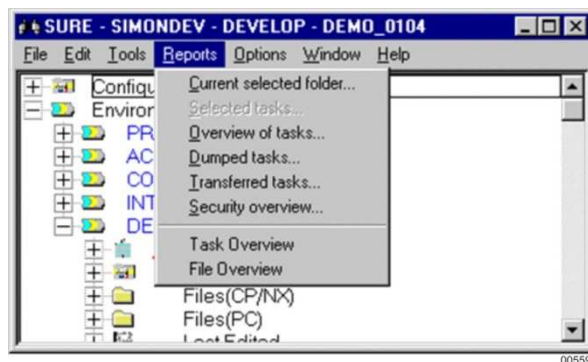
```
[ SUREREPORT ]
<report-entry>
<report-entry>
```

Each report-entry defines a report function. A report-entry can have two layouts:

- A simple layout: <RTF-file>=<Description>

<description>    The text that is shown in the Reports menu.  
 <RTF-file>        The name of the RTF file that does the processing.

**Example**        [ SUREREPORT ]  
                   TASK.RTF=Task Overview  
                   FILE.RTF=File Overview



- An extended layout: <Description>=[<section>]

<description>    The text that is shown in the Reports menu.  
 [<section>]       The name of another section in the AW\_OBJ.INI file.  
                   That section describes  
                   The report input parameters  
                   The selection processing method  
                   The name of the RTF file that does the processing

The key text of the selected line is set in the variable nRef for the report.

**Detailed Description of the Extended Layout <Description>=[<section>]  
(With Examples)**

The syntax of a [<section>] is as follows:

```
[ <SECTION> ]  
SELECT=<selection-type>  
STEP1=<command step>  
STEP2=<command step>  
STEPn=<command step>
```

**<selection-type>**      The selection type determines if and how the current selected items in the SURE browser are used.

The following selection types are available:

- |       |  |
|-------|--|
| NONE  | Indicates that the report does not use the browser's current selection; each command step will be executed once.   |
| GROUP | Groups the browser's current selection and runs the command steps once with this group of nodes as input; selected node names are passed in the array ReportSelection.Names[0..#selected-1]. |
| EACH  | Runs the command steps, in order, for each of the browser's selected nodes; the node name is passed in the variable nRef; this is the default if no selection type node is defined.          |

**<command step>** Command steps must be defined in the order in which they have to be processed, and will be run one time or once per node in the browser's current selection, depending on the value of SELECT.

The following <command steps> are available:

MACRO:<macro name> Runs a predefined macro; macro input variables are processed beforehand, which might result in an input dialog before the macro is run.

INPUT:<parameter>; Shows an input screen where selection parameters can be entered. It is possible to define multiple parameters. Each selection parameter has the following syntax:

<technical name>,<type>,<description>;

<technical name> is the internal name of the item in the SURE browser; the value chosen or typed in the input dialog will be stored in the datamap under this name.

<type> can be DATE, BOOLEAN, or any type in SURE. A DATE results in a calendar button on the input screen; BOOLEAN results in a check box and SURE-type results in a drop-down menu.

<description> is the name of the parameter on the screen.

Each parameter must be terminated with a semicolon.

RTF:<report file name> Interprets an RTF report; the report is started through a method that may show different input formats depending on the entity type associated with the report.

METHOD:<method name>,<object name> Starts a method-for-object; the input format of the method will not be shown prior to the call, and the output format will not be shown when the method finishes.

### Examples

The following is an example of report definitions in the AW\_OBJ.INI file. The blue lines are only comments and should not be placed in the INI file.

```
[SUREREPORT]
Task Details=[REPORT_TASKDETAIL]
Tasks all open=[REPORT_TASKALLOPEN]
Test release letter=[REPORT_TESTRELEASELETTER]
GroupOfTasks=[REPORT_GROUPOFTASKS]
Sep1=
Overview with manual parameter choise=[REPORT_CUSTOM]
```

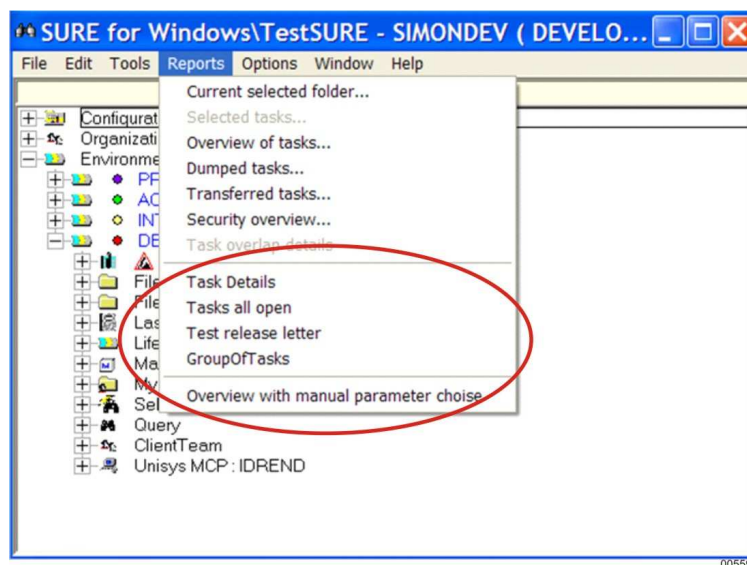
### Explanation

<description>=[<section>] Each line in the [SUREREPORT] section results in an entry in the Reports drop-down box.

<description> is placed in the drop-down box, and should be a clear description of the function of the report.

[<section>] must be a section in the AW\_OBJ.INI file, and that section describes the report parameters. See further.

This [SUREREPORT] section results in the following Reports menu.



- The Reports drop-down menu shows the lines that are defined in the INI file.
- Notice the extra separator, which is defined through the "Sep1=" command: a line without a [<section>] results in an extra separator in the drop-down list. Each separator must have a unique name.

```
REPORT_TASKDETAIL]
SELECT=EACH
STEP1=RTF:C:\SURE\RIS\RTF\Detail.rtf
```

### Explanation

SELECT=EACH	The following STEPs are executed for each item that is selected in the browser.
STEP1=RTF:<filename>	Execute the mentioned RTF file. The RTF file must be available on disk.

```
REPORT_TASKALLOPEN]
SELECT=NONE
STEP1=RTF:C:\SURE\RIS\RTF\TaskAll.rtf
```

### Explanation

SELECT=NONE	The following STEPs are executed once and the browser selection is ignored.
STEP1=RTF:<filename>	Execute the mentioned RTF file. The RTF file must be available on disk.

```
REPORT_TESTRELEASELETTER]
SELECT=NONE
STEP1=MACRO:RELEASE-LETTER
STEP2=RTF:c:\SURE\RIS\RTF\ReleaseLetter.rtf
```

### Explanation

SELECT=NONE	The following STEPs are executed once and the browser selection is ignored.
STEP1=MACRO:<macroname>	Execute the mentioned macro to select items. This macro must be defined in SURE.
STEP2=RTF:<filename>	Execute the mentioned RTF file. The RTF file must be available on disk.

```
REPORT_GROUPOFTASKS]
SELECT=GROUP
STEP1=RTF:c:\SURE\RIS\RTF\GroupOfTasks.rtf
```

### Explanation

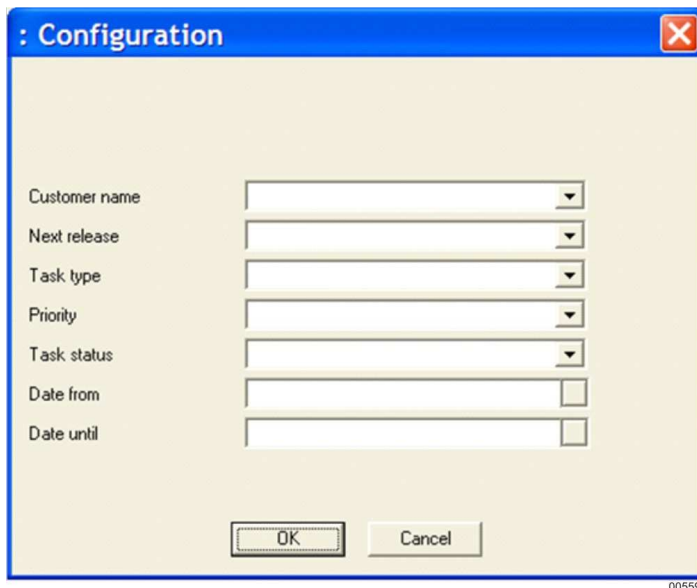
SELECT=GROUP	The following STEPs are executed once with all tasks that are selected in the browser as tabular input.
STEP1=RTF:<filename>	Start the mentioned RTF file. The RTF file must be available on disk.

```
REPORT_CUSTOM]
SELECT=NONE
STEP1=INPUT:ByReported,ANNOUNCED,Reported by;NextRelease,NEXT-
RELEASE,Next release;ByType,PROBLEM-TYPE,Task
type;ByPrio,PRIORITY,Priority;ByStatus,STATUS,Task
status;DateFrom,DATE,Date from;DateTo,DATE,Date until
STEP2=RTF:C:\SURE\RIS\RTF\TaskByQuery.rtf
```

### Explanation

SELECT=NONE	The following STEPs are executed once and the browser selection is ignored.
STEP1=INPUT:<field>;<field>	Show an input screen with the defined fields. Each field must be defined as follows:  <technical name>,<type>,<display text>;
STEP2=RTF:<filename>	Start the mentioned RTF file. The RTF file must be available on disk.

This INPUT step results in the following input screen.



- A calendar button for the DATE fields.
- Drop-down menus for the fields with type CUSTOMER, NEXT-RELEASE, PROBLEM-TYPE, PRIORITY, and STATUS.

The RTF files that are used in these examples are placed in the ...\\RIS\\RFT\\ directory. These RPT files can be used as a starting point to create your own report. Notice that you must give your own RFT-file a unique name; otherwise, it will be overwritten with an installation of a future release.



### 43.3.3.Email Server

The SURE system can use email as an external communication system. For this reason, a single computer in the network functions as a SURE email server. However, from the email side, this is just an ordinary email client connected with, for example, Outlook Express. It is possible to configure a workstation as a "SURE eMail server" during the initial SURE client installation on that workstation. The IP address and the port number must be configured in the Global Settings (through the popup menu under environment). If the email address is not set, most formats will not contain an email indication.

### 43.3.4.Multiple Concurrent Versions

The SURE system allows for multiple concurrent applications against multiple different server systems. You can run different SURE applications against different servers on a single workstation. By default, these applications run without any conflict, if each installation has its own directory and configuration file.

Only during compilation on the ClearPath server hardware, a conflict arises. Because the AS\_COMP program interfaces through DDE with the SURE system, it cannot distinguish the different SURE versions. For this reason, one has to identify the SURE version by configuring a SERVICENAME in the [GLOBAL] section of the configuration file. This requires an additional parameter to the AS\_COMP program called /S<Server>

```
[GLOBAL]
SERVICENAME=MYSURESERVER
OLESERVER=<2::9>
```

Parameters for AS\_COMP in MultiEdit

```
AS_COMP /SMYSURESERVER <LFN> <FILE>.EXT>
```

### 43.3.5.Task Entry Form

The Task entry form can be customized using the popup menu under the environment folder called "Customize Task Form." In the [TASK] section of the configuration file, you can set the focus on the text field if required.

```
[TASK]
Text80_1=FOCUS
```

### 43.3.6.Alternative Task Definition Screens

You can implement a site-specific task screen that overrules the default task screen. There are several examples of programs that support site-specific task screens. These programs are written in Visual Basic.

The default task screen.

Task T\_0632

Task Name: T\_0632  
Status: DEVELOPMENT  
ASSIGNED  
no dependency

Task Definition

Project: SIMON  
Type: DailyChanges  
Environment: DEVELOPMENT  
Reported by: SIMONDEV  
Short description:  
Description: Changes for environment DEVELOP

Attachment  
Solution

To handle by: user SIMONDEV  
Inform assigned to handle by via eMail ☐

Date value ☐  
Boolean value ☐  
Numeric value ☐  
String value ☐

Send email to customer  
When entered ☐  
When solved / closed / denied ☐

24/06/2008 14:01:49 by SIMONDEV  
new  
24/06/2008 13:59:17 by SIMONDEV  
My reminder

Open Modify Info New Print Close

005600

The preceding default task screen can be customized by adding site-specific fields or by deleting standard fields. On this screen, it is not possible to define cross checks between fields.

The main reason to implement a site-specific task screen is to enforce standard workflows. Customization of the generic task screen that is included with SURE is only limited. Writing a custom task entry and update form may integrate better with the defined workflow and the specific customer requirements. For example, the workflows for task-type CHANGE: in the case that a CHANGE-task is entered, it must be linked to a predefined RELEASE-task.

Following are the various site-specific task screens:

An example of a site specific task screen (through a Visual Basic plug-in program).

### The Selected Task

The field with the selected task allows the user to select a task list in the drop-down list below. You could select the To Do list and then all your To Do tasks are selectable. Furthermore all task macros are selectable. If the required task is not present in a list, you can enter the name of the task and press the tab that shows the entered task.

### Master Task

This field allows selecting a master task. To fill this drop-down list, click one of the radio buttons underneath.

### To Handle By

This allows the selection of the one who is going to do this task. The "I start working for this task" option makes the task your current task.

### Information

The information box shows one of the available information carriers attached to the task. For a new task, the description is enabled; however, after the task is created, the other information is available.

### Attachment

The attachment works based on a template file. In the same directory7 as the TaskMnt.exe file, a file named TaskAtt.ZIP may be present. If this is the case, then pressing on the paperclip opens this ZIP file allowing the user to alter information. The zip file is used as a template.

### Extended Data

The extended data shows some task attributes that are not commonly used.

### Site-Specific Data

The site-specific data shows the site-specific task fields defined in the Customize task.

### Ready

This button readies the task.

### Approve

This button APPROVES the task.

Another example of a site-specific task screen.

The screenshot displays the 'SURE for Windows\TestSURE - T\_0632' window. It is divided into several sections:

- Definition:** Includes fields for Identification (T\_0632), Project (SIMON), Type (DailyChanges), Priority, Environment (DEVELOPMENT), Reported by (SIMONDEV), Short description, Reference, and Next release.
- Status:** Shows 'Status' as DEVELOPMENT and ASSIGNED. It includes checkboxes for 'Opened', 'Ready', 'Solved in release', and 'Performed by'.
- Planning:** Includes 'Received' (6-24-2008), 'HH:MM' (12:17), 'Delivery', and 'Effort estimated'.
- To handle by:** Includes 'Usercode' (SIMONDEV) and checkboxes for 'Not assigned', 'Usercode', 'Function', and 'Team'. There is also an 'Inform assignment via email' checkbox.
- Email reporting:** Includes checkboxes for 'Send email to reporter when task entered in SURE' and 'Send email to reporter when task is solved / closed / denied'.
- Reminder:** A yellow box containing the text '24/06/2008 13:59:17 by SIMONDEV My reminder'.
- Information:** A list box containing 'Changes for environment DEVELOP'.
- Bottom Bar:** Includes buttons for 'New', 'Ready', 'Current', 'Update', 'Exit', and 'Attachment' (with a paperclip icon).

The window title bar shows 'SURE for Windows\TestSURE - T\_0632'. The bottom right corner of the window has the text '005602'.

- In this case, the site-specific attributes are placed on the main screen (on the right side).
- The reminder is placed on the screen, and Solution and Documentation are bold if the task has such info.

The “Examples” section of the SURE internet site contains a zipped VB project, “Task Maintenance Tool,” that creates a customized task form.

The following lines in the AW\_OBJ.INI file enforce the usage of a site-specific task screen:

```
[ SURE ]
TaskAdd=C:\SURE\REND\RIS\BIN\TaskMnt.exe /O <OLEID> /A
TaskUpd=C:\SURE\REND\RIS\BIN\TaskMnt.exe /O <OLEID> /U <TaskName>
```

The name of the executable is TaskMnt.exe and it is placed in the ..\RIS\BIN directory.

If multiple SUREforWindows-sessions are open, then the OLESERVER option must be set in the paragraph [GLOBAL] to locate the correct SURE session. This is also set in the INI file according to the next entry:

```
[ GLOBAL ]
OLESERVER=2
```

### Hyperlinks to a URL or SharePoint Page

You can place hyperlinks in the variable task information blocks: task description, solution, documentation impact, and reminder. This can, for example, be an URL of a SharePoint page, where more information about this task (for example, a detailed analysis) can be found.

This functionality is available only on alternative Visual Basic Task definition screens.

### Example

The screenshot shows a Windows application window titled "SURE for WindowsVEND - SURE-2451". The window contains a form with various fields and sections. At the top, it says "The selected task: SURE-2451 TaskMaintenance: make clicking on URLS in text fields". Below this, there are several dropdown menus and text boxes for task details. The "SURE status" is "SOLVED". The "Master task of this task" is "Show no task". The "Reported by" is "SNS". The "Entered" date is "20-8-2009". The "Delivery" is empty. The "Effort estimated" is empty. The "Temporary employee role" is empty. The "Reference" is empty. There is a section "To handle by" with radio buttons for "I start working for this task", "Not assigned", "Usercode", "Function", and "Team". The "Not assigned" option is selected. There is a checkbox "Inform assignment via email" which is unchecked. There is a section "Information" with radio buttons for "Description", "Solution", "Documentation", and "Reminder". The "Description" option is selected. The description text is "TaskMaintenance: make clicking on URLS in text fields route to web browser". Below this, there is an example URL: "Example: http://www.unsys.com/unsys/". At the bottom of the window, there are buttons for "New", "Ready", "Approve", "Apply", "Close", and "Attachment".

- The full URL-name must be placed in the text.
- Double-click the URL-name to open the Web page.

### 43.3.7. Configuring as BUILD Server

The SURE server can run in unattended mode to build applications using the build support facility. To achieve this, the following two criteria must be met:

1. The system should logon automatically to the SURE system (set the <Save password and Accesscode> indication in the logon screen).
2. The following settings must be there in the INI file to define the automatic behavior for the build server:

```
[ BATCH ]
UNATTENDED=TRUE
ENVIRON MENT=<environment>
SERVER=<server>
```

If the SERVER is not specified, the Windows computer name is used as the server name.

### 43.3.8. User Pictures

The SURE system allows pictures to be shown for the configured users. This option requires two configuration settings that define where the images can be found.

```
[ SURE ]
BITMAPPREFIX=<directory>
BITMAPSUFFIX=<suffix>
```

For example:

```
[ SURE ]
BITMAPPREFIX=C:\TEMP\
BITMAPSUFFIX=.BMP
```

These settings locate a file called C:\TEMP\SIMON.BMP for a user named SIMON. The example screen illustrates the user properties screen.



### 43.3.9. Open With

To configure the Utilities that are used to open the files through the SURE Explorer, perform the following steps:

1. Click **Option**.
2. Click **SURE**.
3. Click the **Local options** tab.

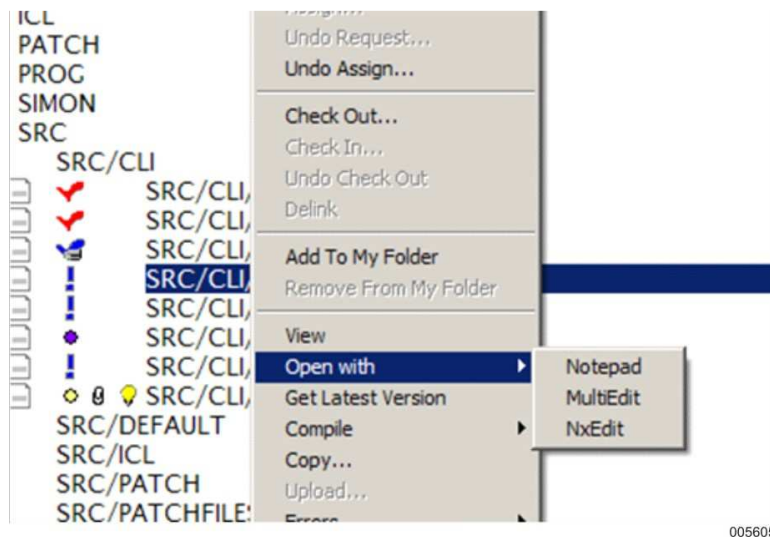
An editor for MCP files, a default editor for PC files, and a DIFF utility are configured. For PC files, the Windows Explorer setting of the file-extension has priority over the default PC-file editor.

SURE allows an additional method to define extra PC-editors to open files. It uses entries in the INI file according to the next example.

```
[OPENWITH]
MultiEdit=C:\MEW\MEW32.EXE
Notepad=notepad.exe
NxEdit=C:\Program Files\Unisys\MCP\ProgWrkbench\NXEdit.exe
```

- The entries must be defined in the paragraph [OPENWITH].
- Each editor is on a single line: the logical name of the editor, followed by the name of the executable.
- The list with defined editor appears in the “Open With” menu.

The “Open with” function is located in the popup menu of a file.

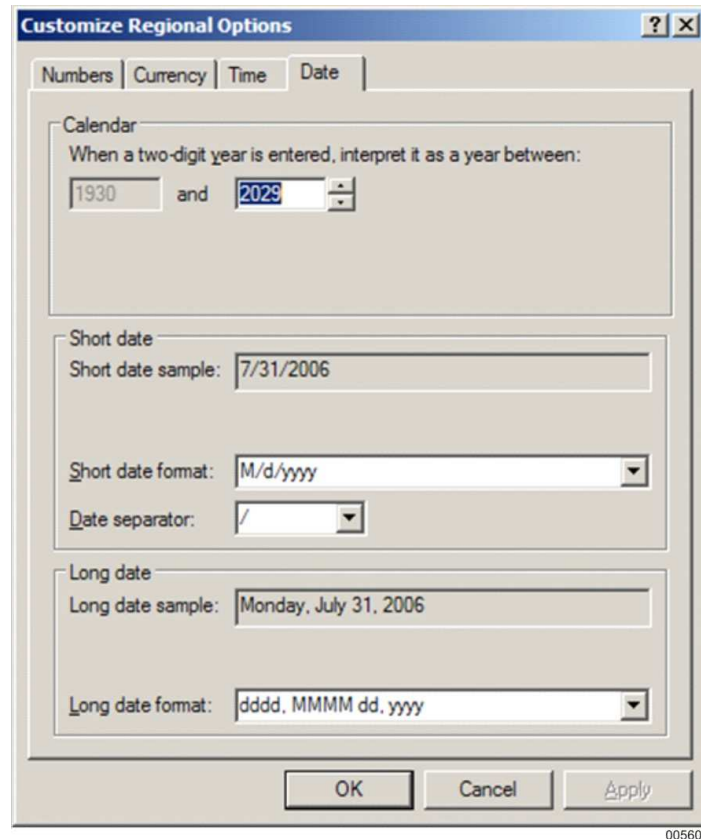




### 43.3.10. Date Format According to the Regional Settings

SURE presents and accepts the dates according to the Windows regional settings.

The Windows regional options contain the settings for the short and the long formats.



The short date format is used for all entry fields.

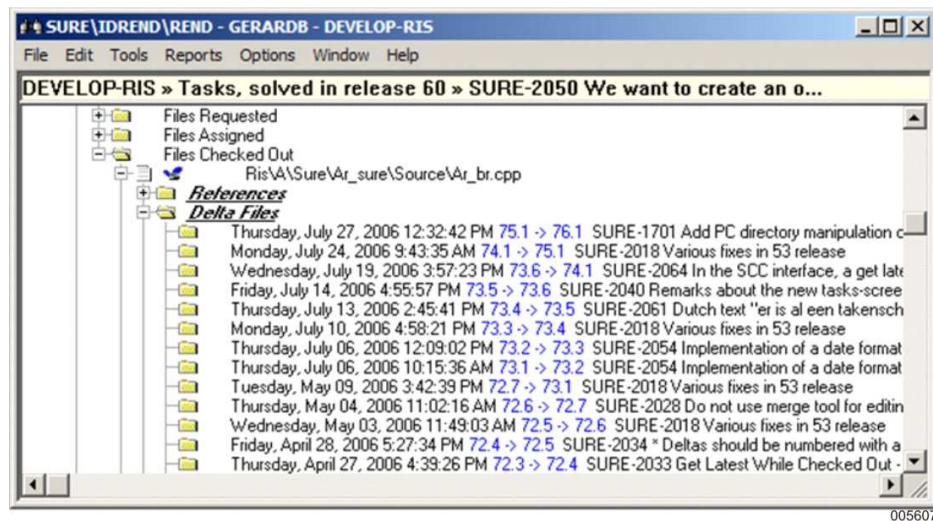
The long date format is used to show date information.

These settings are read and used by SURE. The INI file contains the used settings in the following entries:

```
[GLOBAL]
CURRDATELONGFORMAT=dddd, MMMM dd, yyyy
CURRDATESHORTFORMAT=M/d/yyyy
CURRTIMEFORMAT=h:mm:ss tt
```

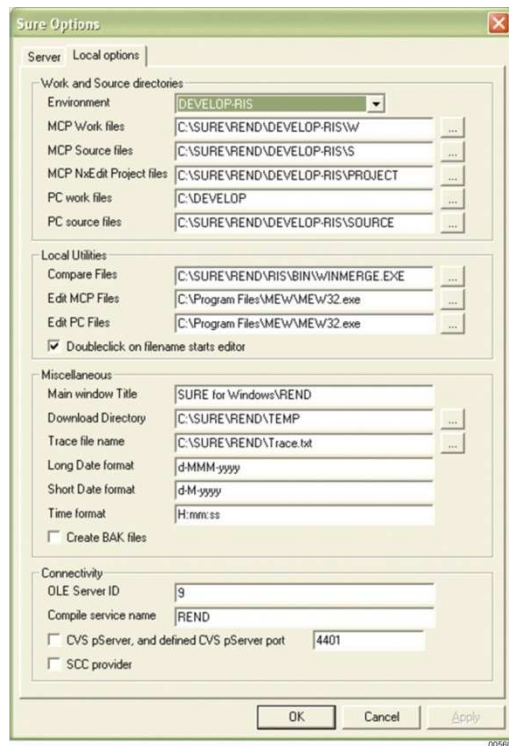
## Windows Interface

This shows information in the following layout.



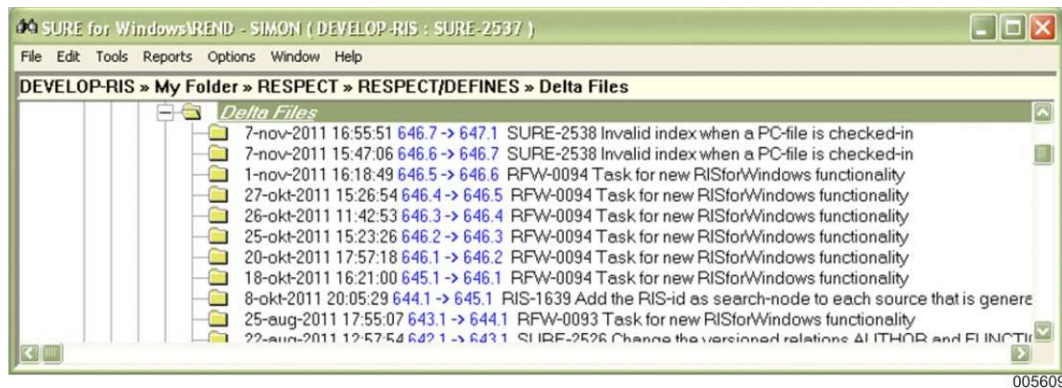
To overwrite the settings in the INI file:

1. Click **Options**.
2. Click **SURE options**.
3. Click the **Local options** tab.



**Note:** The preceded syntax is a valid windows date format.

This shows the information in the following layout.



## 43.4. Keyboard and Mouse Support

The SURE Explorer has of two types of desktop windows:

- The Explorer style browser with multiple sub-windows
- Data-entry windows

Keyboard and mouse support is described for the data entry screens. These screens are similar to Windows dialog screens. However, there are a number of special remarks.

### 43.4.1. Keyboard Support on Data Entry Windows

Arrow	Using the up arrow and down arrow, you can change from the field.
Right Button	Allows for a popup menu that may show restricted value lists.
Esc	Cancel.
Enter	OK or Apply.
F1	Show value list.
F12	Switch to next TAB format.
Ctrl+Alt+Insert	Insert a line in tabular format.
Ctrl+Alt+Delete	Delete a line in tabular format.










**43.4.2.Keyboard Support in the SURE Browser**

ArrowRight	Scroll right one position or drag the current selected line to the top right window.
ArrowLeft	Scroll left one position or close the top right window if this entry is dragged on it.
ArrowUp	Select next line.
ArrowDown	Select previous line.
BackSpace	One hierarchical level up.
Ctrl {Arrow Up, Arrow Down, Home, End, Page Up, Page Down}	Position only.
Alt Enter	Execute property function.
Ctrl Enter	Drag the current selected line to the top right window.
Ctrl Insert	Copy selection to clipboard.
End	Select the last line.
F1	Show the popup menu.
F5	Refresh the selected level.
Home	Select the first line.
Page Up	Select the first line on the visible page.
Page Down	Select the last line on the visible page.
Ctrl {Arrow Up, Arrow Down, Home, End, Page Up, Page Down}	Select and do not deselect.
Shift Tab	Set focus to the next window.

**43.4.3.Mouse Support**

Left Button Down	Select line.
Left Button Down (object)	Invoke the property command.
Left Button Down (status)	Invoke the log command.
Left Button Down (info)	Start view information.
Right Button Down	Show the popup menu for line
Ctrl+Left Button Down	Select line and do not deselect line.
Shift+Left Button Down	Select all entries between the previously selected line and the current one.

The SURE Explorer allows support for clicking the following objects.

Object	Invoke properties
Status	Invoke Log
Indicator	Start view/update indicator contents
<b>Example</b>	     SURE_QUALITY_0039 OLE automation
	<i>is the task object</i>
	<i>is the status object</i>
	<i>is an indicator</i>
	<i>is an indicator</i>

#### 43.4.4. Drag and Drop Operations

Drag an entry from any window and drop it in an empty window on the right. The entry expands in the window on the right. If the entry type contains columns (files tasks and users), these columns are filled according to the user-defined attributes.

For more information, do the following:

1. Click **SURE**.
2. Click **options**.
3. Click **attributes**.

A right side window may be emptied through the empty command (menu choice in windows/empty or in the popup menu on the target window). If the window is not empty, the above described drag and drop operation can also be done using the Ctrl key.

Drag Object	Drop Object	Action
Directory	My Folders	Add a directory to My Folders
File	My Folders	Add a file to My Folders
File	Task in my To Do list	Check out file
File	Task in any other list	Assign a file to a task
File in linked list	Task	Move a file to a task
Task	File	Assign a file to task
Task	Task	Make the drag task dependent on the Drop task
Task	EmployeeFunction	Assign a task to EmployeeFunction
Task	User	Assign a task to User
User	EmployeeFunction	Add a user to EmployeeFunction
User	Team	Add a user to Team

### 43.4.5.Explanation of Icons

The various objects are:



File



Task



User



Environment



Various configuration objects



Organization , user team



Macro



Employee function

A file name can be marked with one of the following status icons.



The file is requested.



The file is assigned to me.



The file is checked out by me on the server.



The file is checked out by me with the local edit option.



This referenced file has been changed for another task as the master file.



The file has been logically removed.



The file is assigned to another user.



The file is checked out by another user.



The file is only present as a name.



The file is stable and transferred to the next environment. The color is defined in the environment option



The file is anchored.

A task name can be marked with one of the following status icons.



The task is marked as ready.



The task is the current task.



The task is assigned.



The task is solved.



The task is quick fixed.



The task is entered and not yet assigned.

A file can be marked with one of the following information icons.



The file has an attachment.



The file has reminder information.



The file has a textual description.

A task can be marked with one of the following information icons:



The task has an attachment.



The task has solution information.



The task has reminder information.



The task has documentation impact information.

### 43.5. Installation of “backupfile” in MS Word

Using the SURE browser, it is very easy to download MCP printer backup files from the mainframe to your PC. After the backup file is copied to the PC, it is loaded in Word because, in most cases, the user wants to see the contents of that backup file. We have chosen to load the backup file in Word, because Word correctly handles the page-skips in the backup file.

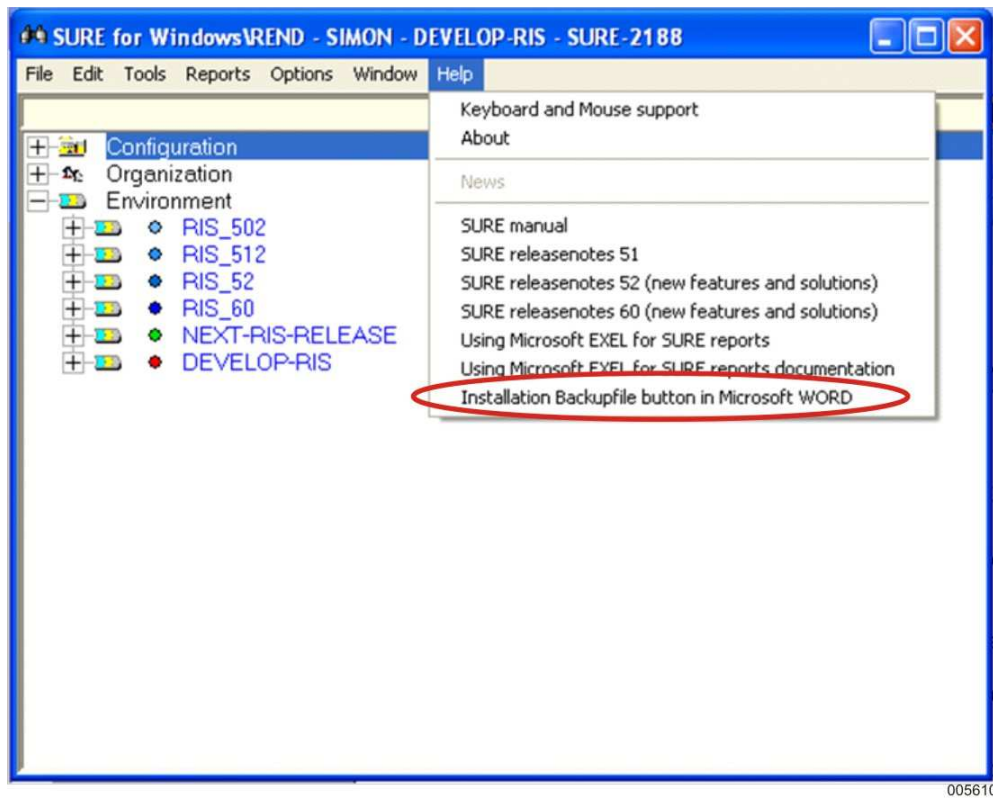
The problem that now arises is that the default font and font size of Word are not appropriate for the downloaded backup file: MCP printer backup files are created with a non-proportional font and up to 132 characters on a line, and Word’s default font and font size are mostly not set accordingly. This problem is fixed with a button&macro: BackupFile.

The “BackupFile” option (in Word) sets the font to Courier New (not proportional), and changes the font size to ensure that each line can have 132 characters and each page can have 132 lines. A print of the file gives correct output: no lines are wrapped or truncated, and there are no unexpected page-skips.

The layout of the report can be checked online through the Print Preview option.



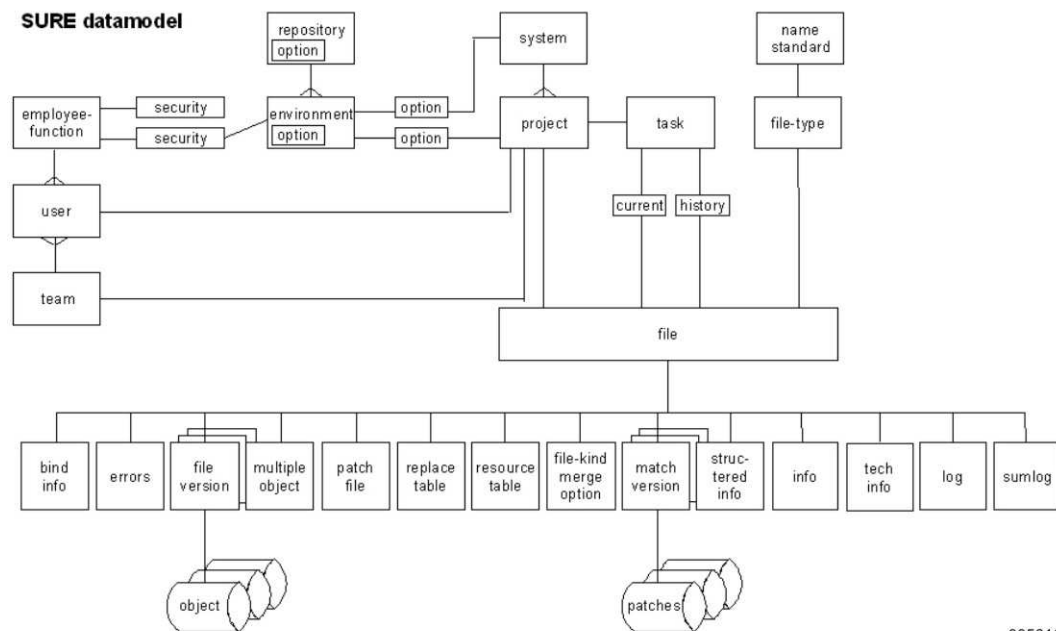
The instructions to install the “Backupfile” option in Word are available in the Help menu.





## Section 44

# SURE Datamodel



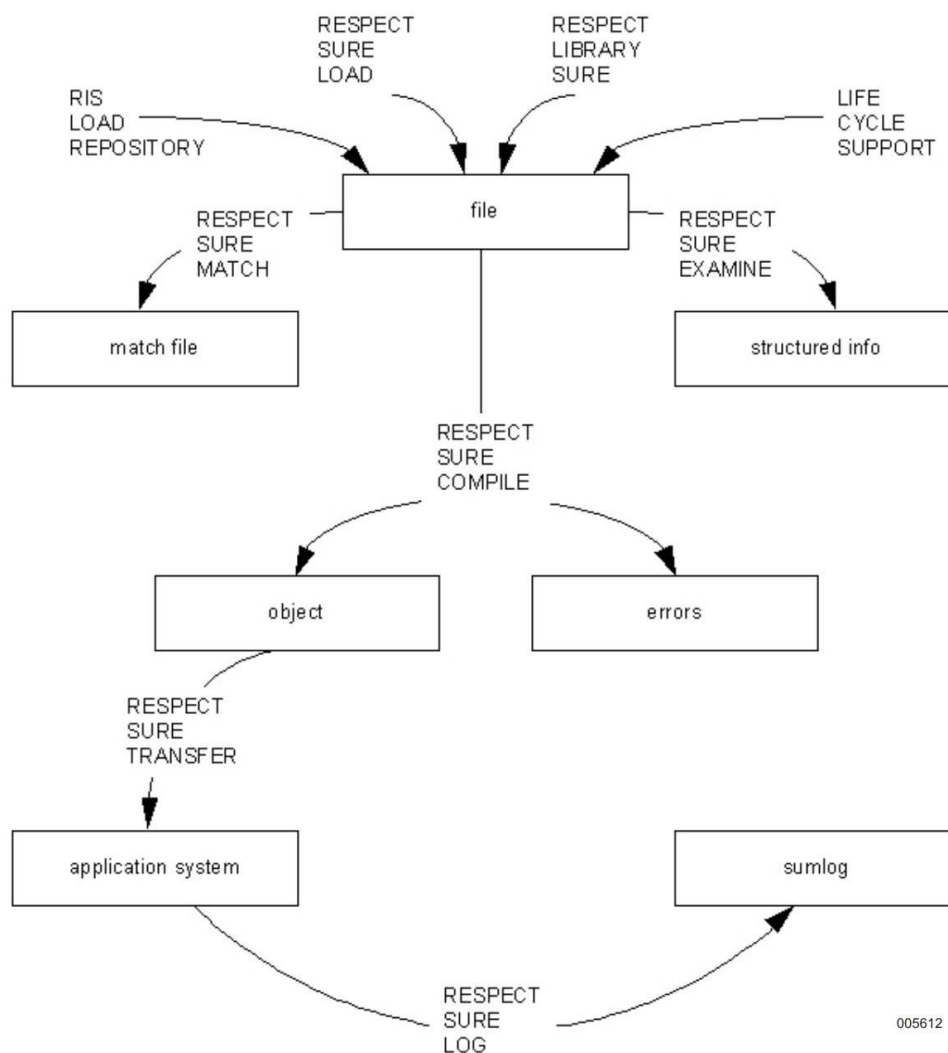
005611



## Section 45

# SURE Process Model

**SURE process model**



005612

repository	Environment/Global Options
environment	<Environment>/Properties
Name Standards	<at file-type , global options>
file-type	Configuration/File Type
Employee-Function	Organization/Employee Function
User	Organization/User
Team	Organization/Team
Security	<per employee function , user>
System	<Environment>/System
Project	<Environment>/System/Project
Task	new
File	new

### Bind-Info

#### *Relations*

Group owner:class(asset)	Action
FILE <file>:DRIVER(<driver or bound code>)	1
FILE <file>:START-JOB(<bind-job-name>)	1
FILE <driver or bound code>:START-JOB(<bind-job-name>)	1
FILE-CONTROL <bind-job-name>:JOB-STATUS(JOB-BUSY)	2
FILE-CONTROL <bind-job-name>:JOB-STATUS(JOB-READY)	3

- 1 Created by RELATE function in SURE (Terminal Emulation)
- 2 Created by RESPECT/SURE/COMPILE  
Deleted by RESPECT/SURE/FINISH
- 3 Created by RESPECT/SURE/FINISH

**Errors***Relations*

<b>Groupowner:class(asset)</b>	<b>Action</b>
FILE-CONTROL <file>:COMPILE-STATUS(SYNTAX)	1

*Information*

<b>Group</b>	<b>Owner</b>	<b>Class</b>	<b>Action</b>
FILE-CONTROL	<file>	SYNTAX	1

1 Created by RESPECT/SURE/COMPILE if compiler indicated syntax errors.

**FileVersion***Relations*

<b>Groupowner:class(asset)</b>	<b>Action</b>
FILE <file>:STOR(<NULL>)	1

*DSTOR*

<b>Owner</b>	<b>Class</b>	<b>FileVersion</b>	<b>Action</b>
<file>	TOTAL	<FileVersion>	1

1 Relation contains the File Version in REL-VALUE  
 Created by RIS/MENU (SURE)  
 Created by RESPECT/LIBRARY/SURE  
 Created by RESPECT/SURE/LOAD  
 Deleted by RESPECT/SURE/MATCH

### Multiple Objects

#### *Relations*

Group owner: class(asset)	Action
FILE-CONTROL <file>:MULTI-OBJECT(<object>)	1
FILE-CONTROL <object>:OBJECT-USERCODE(<usercode>)	1
FILE-CONTROL <object>:OBJECT-PACK(<pack>)	1
FILE-CONTROL <object>:HOST(<host>)	1
FILE-CONTROL <object>:SYSTEM(<system>)	1
FILE-CONTROL <object>:SECURITY(<security>)	1

- 1 Maintained by MULTIOBJECT in UI (Terminal Emulation)  
 Maintained by Command/Miscellaneous/Multiple Object UI (Windows)  
 Used by RESPECT/SURE/COMPILE  
 Used by RESPECT/SURE/TRANSFER

### Patch Files

#### *Relations*

Group owner: class(asset)	Action
FILE <file>:PATCH-FILE(<file>)	1

- 1 Maintained by PATCH in UI (Terminal Emulation)  
 Maintained by Command/Miscellaneous/Patch files UI (Windows)  
 Created by RESPECT/SURE/RELATE/PATCH  
 Used by RESPECT/SURE/COMPILE

### Replace Tables

Definition	Environment properties, system or project properties
Contents	<replace target>@ <replace with>@
Used	RESPECT/SURE/COMPILE if the replace table is visible

### Filekind Merge Options

Definition	Environment properties, system or project properties, file-type properties file-properties, filekind (if filekind is defined as file-type)
Contents	\$ <any compiler control cards> \$ SET MERGE is defined automatically by SURE



**Delta Files***DSTOR*

Owner	Class	FileVersion	Action
<file>	MATCH	<FileVersion>	1

1 Created by RESPECT/SURE/MATCH

**Structured Info**

Relations created by RESPECT/SURE/EXAMINE

Groupowner:class	
FILE-CONTROL	<file>:CARD
FILE-CONTROL	<file>:COPY-FILE
FILE-CONTROL	<file>:DATABASE
FILE-CONTROL	<file>:DATASET
FILE-CONTROL	<file>:DISK
FILE-CONTROL	<file>:EXECUTE
FILE-CONTROL	<file>:EXTERNAL
FILE-CONTROL	<file>:FORMAT
FILE-CONTROL	<file>:FORMAT-FILE
FILE-CONTROL	<file>:LIBRARY
FILE-CONTROL	<file>:OBJECT
FILE-CONTROL	<file>:PORT
FILE-CONTROL	<file>:PRINTER
FILE-CONTROL	<file>:PUBLIC
FILE-CONTROL	<file>:REMOTE
FILE-CONTROL	<file>:SET
FILE-CONTROL	<file>:SYSTEM
FILE-CONTROL	<file>:TAPE
FILE-CONTROL	<file>:TRANSACTIONBASE
FILE-CONTROL	<file>:UPDATE
FILE-CONTROL	<file>:ADDS-DICTIONARY
FILE-CONTROL	<file>:ADDS-PROGRAM
FILE-CONTROL	<file>:ADDS-FILE
FILE-CONTROL	<file>:ADDS-GROUP

Relations created by RIS/MENU (SURE)

Group Owner:Class	
FILE	<file>:AUTHOR
FILE	<file>:FILE-TYPE
FILE	<file>:FILEKIND
FILE	<file>:FUNCTION
FILE	<project>:PROJECT
FILE-CONTROL	<file>:ASSIGNED
FILE-CONTROL	<file>:MAINTAINED
FILE-CONTROL	<file>:OBJECT-HOST
FILE-CONTROL	<file>:OBJECT-PACK
FILE-CONTROL	<file>:OBJECT-USERCODE
FILE-CONTROL	<file>:REQUESTED
FILE-CONTROL	<file>:SOURCE-USERCODE
FILE-CONTROL	<file>:SOURCE-PACK
FILE-CONTROL	<file>:SOURCE-HOST
FILE-CONTROL	<file>:SOURCE-COPY

### Info

*Information*

Group	Owner	Class	Action
FILE	<file>	INFORMATION	1

- 1 Maintained by UI (Terminal Information) UPDINFO  
Maintained by UI (Windows) through Info/Information

### Tech-Info

*Information*

Group	Owner	Class	Action
FILE-CONTROL	<file>	TECHNICAL-INFO	1

- 1 Maintained by UI (Terminal Information) UPDTECH  
Maintained by UI (Windows) through Info/Technical

**Log***Information*

Group	Owner	Class	Action
FILE-CONTROL	<file>	LOG-USERCODE	1

- 1 Created by RIS/MENU (SURE)  
Created by RESPECT/SURE/COMPILE  
Created by RESPECT/SURE/TRANSFER  
Visible by UI (Terminal Information) LOG button  
Visible by UI (Windows) through Info/Log

**SUMLOG***Information*

Group	Owner	Class	Action
FILE-CONTROL	<file>	RUN-INFORMATION	1

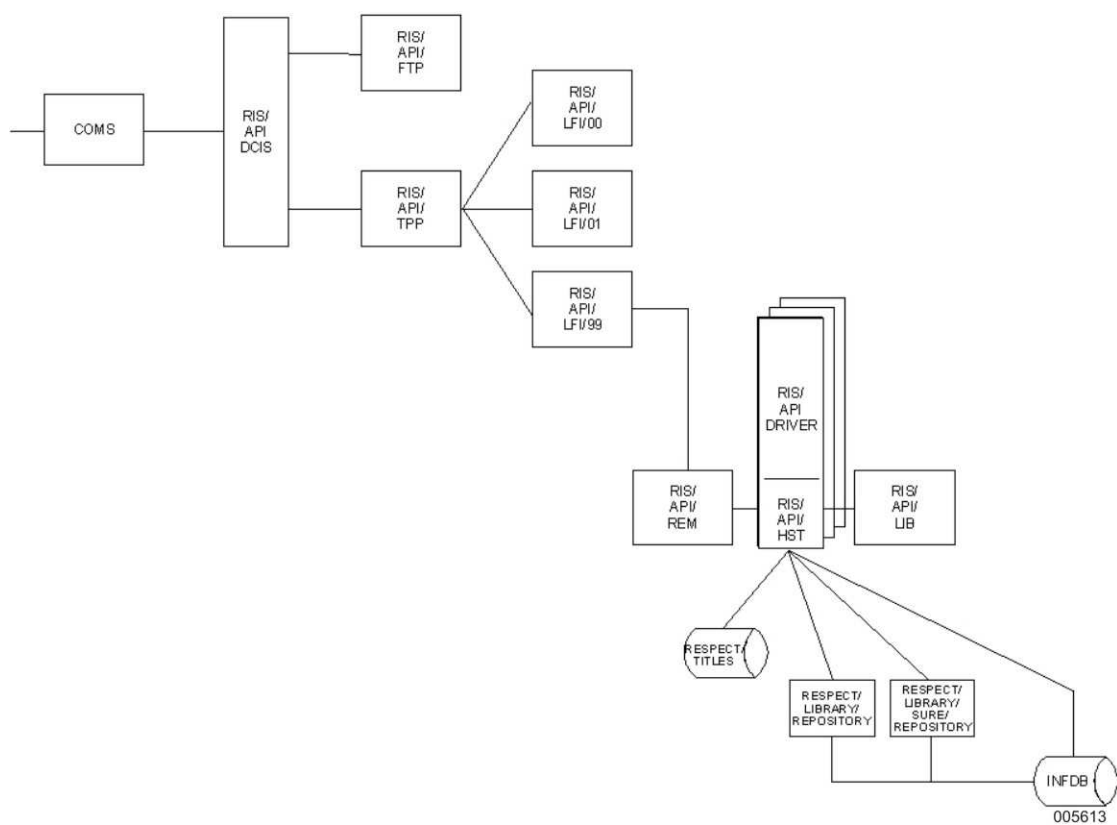
- 1 Created by RESPECT/SURE/LOG  
Visible by UI (Terminal Information) OPERATOR button  
Visible by UI (Windows) through Info/Statistics



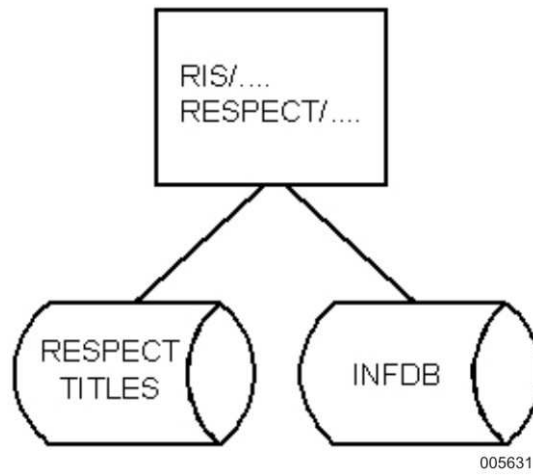
## Section 46

# Program Structure

### 46.1. Online Programs (Windows)



## 46.2. Batch Programs (Windows)



## Section 47

# Program Index

This section describes all the programs delivered with a SURE release.

Each program is described as follows:

- A brief explanation of the purpose of the program.
- Input parameters, files, task values, task string, and accept messages.
- Output parameters, files, task values, and listings.
- An example on how the program is started?
- A description about when is the program used?
- Security aspects of the program (plus the reason).

All comma's and equal sign used in parameters are ignored by the programs, except where a comma or equal sign mentioned in the parameter railroad diagram in this section.

### Example

The parameter in

```
RUN RESPECT/REPOSITORY("SELECT, CLASS = REQUEST, ASSET = SIMON")
```

is treated as:

```
RUN RESPECT/REPOSITORY("SELECT CLASS REQUEST ASSET SIMON")
```

### RESPECT/CONVERT

This program is used to convert a file that must be file transferred to a PC.

Input            target            :    999            = help info about this program.

### Example

```
RUN RESPECT/CONVERT;TARGET = 999
```

WHERE

This program can be used by the customer in self-written jobs.

### **RESPECT/LIBRARY**

All SURE programs use this library for initialization purposes. The library also does the screen handling for the online program on non-intelligent terminals (TD830/ET type). The interpretation, presentation, and screen validations are done through this library.

This library freezes permanently and provides stubs for every used repository. Closing the repository database requires thawing this library.

Input	Task value	:	software key to open a temporary test period.
	accept	:	key install date.
			length of temporary test period.

Security

MP +PU (this library contains SYSTEMSTATUS commands).

MP +TASKING (The library must always run under the repository usercode)

### **RESPECT/LIBRARY/SUPERVISOR**

This library is used to compare the creation date of an object in the run environment with the compilation date in SURE.

This library is used optionally, by Metalogic Supervisor.

### **RESPECT/LIBRARY/SURE**

This library is used to load files automatically into the repository.

This library is used by

- All generating modules in the RIS software to load the generated sources.
- The local compilation process in the SURE Explorer.

This library freezes permanently and provides stubs for every used repository. Closing the repository database requires thawing this library.

Security

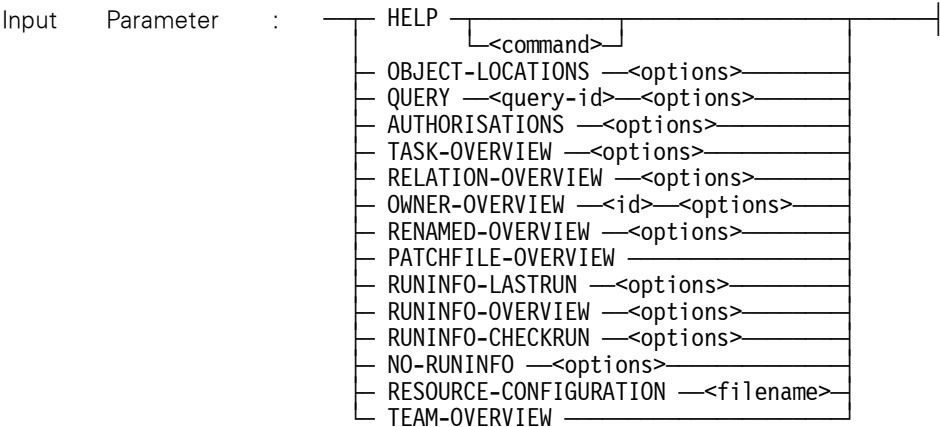
MP +PU (this library creates files in other directories).

MP +SECADMIN



RESPECT/PRINT

This program creates various overviews.



Task string      :      <Environment>.

Example

```
RUN RESPECT/PRINT( "<param>" );TASKSTRING = "DEVELOPMENT"
```

OBJECT-LOCATIONS	This function creates an overview of all the default object-locations that are defined in the repository, and the files that deviate from their default object-location.
QUERY	<p>This function creates a list of file names that meet the query expression.</p> <p>Refer to Section 13, "Query Facility," for more information on query expression.</p>
RELATION-OVERVIEW	This function prints all the relations of a specific class.
AUTHORISATIONS	This function prints the SURE authorization scheme.
TASK-OVERVIEW	This function lists all active tasks in an environment, including the task-overlaps end master-tasks. It also gives an overview of changed files that are not connected to a task.
OWNER-OVERVIEW	This function lists all attributes of the entered <id>.
RENAMED-OVERVIEW	This function lists all renamed files and items.
PATCHFILE-OVERVIEW	This function lists all files with one or more patchfiles.
RUNINFO-LASTRUN	This function lists all programs that did not run since a date.
RUNINFO-OVERVIEW	This function lists all run information. The overview also contains graphics about the system resources that were used by the selected programs. This overview may be helpful to discover programs that use many of the systems' resources.
RUNINFO-CHECKRUN	This function shows how many times programs have run (for each month).

NO-RUNINFO	This function shows programs without run information.
RESOURCE-CONFIGURATION	This function creates an overview with the definitions of all resource-locations, and how the resource definitions are used in a specific file.
TEAM-OVERVIEW	This gives an overview of the team with their users and projects, and the users with their teams and projects.

### **RESPECT/REPLACER**

This program replaces a file, using a replace table.

Input	file	:	REPLACE/TABLE	This file contains strings that must be replaced by other strings. See "Replace" in Section 19.
	Task string	:	The name of the file that has to be replaced.	
Output	file	:	The replaced file.	

### **Example**

```
RUN RESPECT/REPLACER; TASKSTRING = "PROG/AA"
```

Where

This program is used by the customer in self-written jobs.

## RESPECT/REPOSITORY

This program is used to check or maintain the total repository, depending on the input parameter.

Input	Parameters	:	<div> <div>HELP</div> <div> <div>&lt;Command&gt;</div> <div>RIS</div> </div> </div>
			<div> <div>CHECK-REPOSITORY-LEVEL</div> <div>SET-REPOSITORY-LEVEL &lt;nr&gt;, &lt;nr&gt;</div> <div>CREATE-RESPECT-TITLES &lt;DB-name&gt;, &lt;obj-names&gt;</div> <div>UPDATE-RESPECT-TITLES &lt;DB-name&gt;, &lt;obj-names&gt;</div> <div>OVERRIDE-INFDB</div> <div>INITIALIZE-REPOSITORY</div> <div>SET-DEFAULT-DIRECTORY</div> <div>FIRSTUSER</div> <div>CREATE-ENVIRONMENT FROM &lt;env&gt; TO &lt;env&gt;</div> <div>DELETE-ENVIRONMENT &lt;env&gt;</div> <div>GENERATE-WFL</div> <div>DUMP-REPOSITORY</div> <div>LOAD-REPOSITORY</div> <div>DUMP-MESSAGES --&lt;Options&gt;</div> <div>LOAD-MESSAGES --&lt;Options&gt;</div> <div>DUMP-FORMATS --&lt;Options&gt;</div> <div>LOAD-FORMATS --&lt;Options&gt;</div> <div>LOAD-PATCH-REASONS</div> <div>CHECK-PCNAMES --&lt;translation mode&gt;</div> <div>RESET-PC-FILETYPES</div> <div>CHANGE-PROJECT-OF-DIRECTORY &lt;options&gt;</div> <div>CHECK-ALL-STATUS</div> <div>CHECK-SITE-LIBRARY</div> <div>CLEAR-FIND</div> <div>MULTI-OBJECT --&lt;input-file&gt;</div> <div>CHECK-DIRECTORY --&lt;Options&gt;</div> <div>CHECK-PCNAMES --&lt;Options&gt;</div> <div>SELECT --&lt;Options&gt;</div> <div>CHECK-COMPILED --&lt;Options&gt;</div> <div>CHECK-SYSTEM-ENVIRONMENTS &lt;system&gt;</div> <div>SELECT-FILES &lt;Options&gt;-FUNCTION=MARK-COPYFILES</div> <div>DUMP-INFO-CLASS &lt;class&gt;</div> <div>LOAD-INFO-CLASS &lt;class&gt;</div> <div>SYNCHRONIZE-PATCHFILES &lt;options&gt;</div> </div>

Task string : <environment>.

### Example

```
RUN RESPECT/REPOSITORY("<param>");TASKSTRING = "DEVELOPMENT";
```

Where

This program can be used by the customer in self-written jobs.

This program is used in the installation.

Security

MP +PU (the program can put files in the global directory).

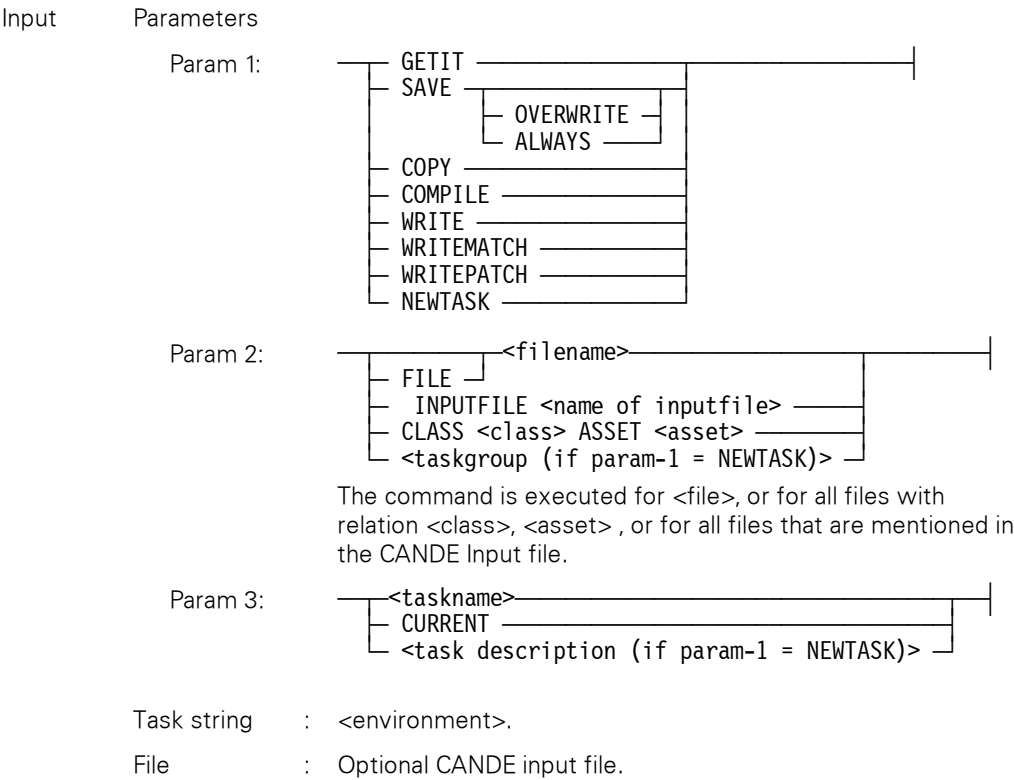
CHECK-REPOSITORY-LEVEL	<p>This function checks the version of the repository during the installation of a new SURE release. If this program gets a DMS version error, then an extra DASDL compilation is done to create an INFDB description file with infra-designs dataset timestamps and site-specific physical attributes.</p> <p>If the program runs correctly, then the current version of the repository is returned using the task value. Dependent on this current version and the new SURE version on tape, a number of conversion programs are started.</p>
SET-REPOSITORY-LEVEL	<p>This function sets the version of the repository.</p>
CREATE-RESPECT-TITLES	<p>This function creates a file RESPECT/TITLES.</p>
OVERRIDE-INFDB	<p>This function performs an AX OVERRIDE to override all the dataset timestamps of the INFDB database.</p>
UPDATE-RESPECT-TITLES	<p>This function updates existing RESPECT/TITLES.</p>
INITIALIZE-REPOSITORY	<p>This function can only be used if the repository is brand new (or totally initialized through DMUTILITY INITIALIZE ALL). This function sets the basic options for a repository.</p>
SET-DEFAULT-DIRECTORY	<p>This function changes the default work pack from IDRD to the current family.</p>
FIRSTUSER	<p>This function authorizes the current usercode for all online functions. The program cannot run multiple times.</p>
CHECK-CONSISTENCY	<p>This function checks the consistency of the repository. It optionally removes unused file names and user names</p> <p>Task value = 0: check and remove.</p> <p>Relations with OWNER = 0.</p> <p>Relations with VERSION = 0 and GROUP = 0.</p> <p>Item or file names that consist of spaces only.</p> <p>Last info-blocks that consist of spaces only.</p> <p>Invalid TYPE CODE relations.</p> <p>Task value = 1:</p> <p>All of a run with task value = 0, plus:</p> <p>Check the owner, class, and asset of all relations, and print and remove the relation if the owner, class, or asset does not exist.</p> <p>Print and remove all file names and usernames that are not used.</p> <p>Task value = 2:</p> <p>All of a run with task value = 0, plus:</p> <p>Check the owner, class, and asset of all relations and Print the relation if the owner, class, or asset does not exist.</p> <p>Print all file names and usernames that are not used.</p>
CHECK-ENVIRONMENT	<p>This function compares the files of two environments in the repository, and it checks and updates the STATUS for all files and RIS-entities. The TASKSTRING identifies the environment that is compared with its next environment.</p>

CHECK-EUROFIELD CHECK-RULE-CONNECTION CHECK-FORMATLIST CHECK-TRAN-AUTH	These functions are used for debugging purposes.
CLEAR-WINDOWS	This function is used to clear all RisForWindows information of an item.
CREATE-ENVIRONMENT	This function is used to create a new environment in the repository.
DELETE-ENVIRONMENT	This function is used to delete an environment from the repository.
GENERATE-WFL	This function is used to generate supporting jobs for the SURE software.
DUMP-REPOSITORY	This function is used to dump the TRANSFER-REQUEST queue from the repository to disk files. The files are placed in the RESPECT/DUMP/SOURCE/= directory. The dump control file is called RESPECT/DUMP/TRANSFER.
LOAD-REPOSITORY	This function is used to load a previously dumped directory into the repository. The directory is RESPECT/DUMP/SOURCE/=. The load-control-file is called RESPECT/DUMP/TRANSFER.
DUMP-MESSAGES	This function dumps messages from the repository into a file.
LOAD-MESSAGES	This function loads messages from a file into the repository.
DUMP-FORMATS	This functions dumps formats from the repository into a file.
LOAD-FORMATS	This function loads formats from a file into the repository.
LOAD-PATCH-REASONS	This is a conversion function for old SURE users: If a delta-file is not linked to a task, then a task is linked to that delta-file. The task-history is used to obtain the correct task name.
CHECK-PC-NAMES	All names of PC files (in SURE) are checked and corrected if necessary (to uppercase, lowercase, or first character up and the rest lowercase).
RESET-PC-FILETYPES	The file type of a PC files is set to the file extension.
CHANGE-PROJECT-OF-DIR	All files in the directory get the given project.
CHECK-ALL-STATUS	The status of all files is checked.
CHECK-SITE-LIBRARY	This function displays the active site-library functions.
SELECT	This function selects a group of files (through a class or asset) and displays them, prints them, writes them in a file, or adds a relation to them.
CLEAR-FIND	This function clears all SURE find results.
CHECK-DIRECTORY	This function refreshes the directory structure. The project of each file within a directory is linked to the directory name.
CHECK-PCNAMES	This function resets the case of all PC files in a directory to a site standard.

MULTI-OBJECT	Defines multi-object definitions in batch mode using an input file.
CHECK-COMPILED	Checks that programs are compiled with the latest version of each copy-book.
CHECK-SYSTEM-ENVIRONMENTS	<p>Checks the content of environments:</p> <ul style="list-style-type: none"><li>• Sources that belong to a system that has excluded environment are removed from those environments by the batch function.</li><li>• Sources that are wrongly not available in an environment are recovered from the next higher environment.</li></ul>
SELECT-FILES <class/asset> FUNCTION=MARK-COPYFILES	<p>Selects all sources with the class/asset relation and adds that same class/asset relation to all copy files of those selected sources. So, the result is a complete set of sources and copy files that are all marked with the same class/asset relation. You can use this relation in a next step in the SURE compile batch. For example, you can dump those files through RESPECT/SURE/DUMP("&lt;class&gt;","&lt;asset&gt;".)</p>
DUMP-INFO-CLASS LOAD-INFO-CLASS	Dumps or loads all DINFO records with a specific class into a data file.
SYNCHRONIZE-PATCHFILES	If a source is maintained using patch files, then its patch file relations must be equal in all environments. This batch function can be used in the case that the patch-file relations are not equal in all environments to repair the patch-file relations.

RESPECT/SURE/BATCH

This program performs certain SURE functions through a BATCH interface.



Example

```
RUN RESPECT/SURE/BATCH( "GETIT" , "PROG/AA" , "TASK25" );
TASKSTRING = "DEVELOPMENT"
RUN RESPECT/SURE/BATCH( "NEWTASK" , "GRP1" , "<task description>" );
```

Where

This program can be used by the customer in self-written jobs.

This program scans a pack directory and creates a cleanup job to remove all unused files.

### Example

Where

## Security

## RESPECT/SURE/CLOSEQUEUE

### Example

Where

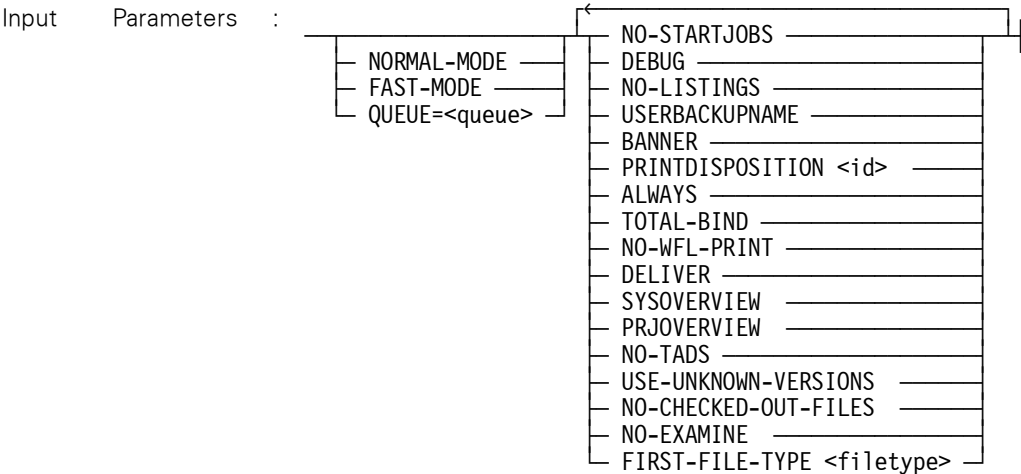
## Security

8207 4121-000



RESPECT/SURE/COMPILE

This program handles all compilations for an environment.



The default mode = NORMAL-MODE

- Task string : <environment>.
- Task value : Amount of simultaneous compilations, default 1.
- File : SURE/COMPILE/<filekind> Compiler card file.
- SURE/COMPILE/TABLE/<sys> Translate table
- sources to be compiled
- Compilers
- XGEN/CONFIGS XGEN configuration file.
- Output file : Compiled objects.
- Overview : An overview of the compiled files and the integrity status.

Example

```
RUN RESPECT/SURE/COMPILE;TASKSTRING = "DEVELOPMENT";
```

Where

This program is used in the SURE evening batch.

Security

MP +PU.

MP +SECADMIN (the program adds predefined securities to the compiled objects).

NORMAL-MODE	(default) Select files that are placed in the normal compile queue. The newest version of each file or copy file in this environment is used for the compilation, unless the file is linked to a specific compile queue. (Previous version is then used.)
FAST-MODE	Select files that are placed in the compile fast queue. The newest version of each file or copy file in this environment is used for the compilation, unless the file is linked to a specific compile queue. (Previous version is then used.)
QUEUE=<queue>	Select files that are placed in the specified compile queue. The newest version of each file or copy file is used for the compilation. A compile-queue is automatically available for each defined baseline.
NO-STARTJOBS	No start-jobs are started (see later in this chapter).
NO-LISTINGS	No compilation listings are created.
USERBACKUPNAME	The compilation backup files are named as follows: COMPILOUT/<source>/LIST on <my family>. The backup file of an XREF listing is called COMPILOUT/<source>/XREF on <my family>.
PRINTDISPOSITION <id>	Identify the print disposition of the compilation listings.
BANNER	Print a banner (with the source name) in front of each compilation listing.
DEBUG	Extra DEBUG info is printed on the compilation overview.
ALWAYS	Used only for compilations in a development environment. If this option is used, then RESPECT/SURE/COMPILE does not go to end-of-task when all compilations are done, but it checks every minute if new files are placed in the compile-queue. If the compile queue is not empty, then RESPECT/SURE/COMPILE resumes compiling. The integrity mechanism is ignored in this mode. DRIVER relations are ignored in this mode. A start-job is started for each START-JOB relation. Compiled objects are copied to their object-locations by RESPECT/SURE/COMPILE. It is not necessary to run the transfer program.
NO-TADS	Enforce that all programs are compiled without TADS.
TOTAL-BIND	Start-jobs can be started for a compiled source, or for the driver of that compiled source. The later situation results in a total bind. By default, a start-job is started for the compiled-source itself, if a START-JOB relation is linked to that source. If this parameter is used, and a start-job relation is linked to the driver, then the driver's start-job is started. This parameter is useful in combination with the parameter ALWAYS. Refer to Section 24, "Binding of Programs," for more information.

PRJOVERVIEW	A compilation overview is made for each application project. This overview reports (for each project) the files that are compiled, blocked by integrity, or added to the transfer queue. The tasks that were the reason of the compilations are also reported. This can be a handy overview for the application project manager.
SYSOVERVIEW	A compilation overview is made for each application system. This overview reports (for each system) the files that are compiled, blocked by integrity, or added to the transfer queue. The tasks that were the reason of the compilations are also reported. This can be a handy overview for the application system manager.
DELIVER	If this parameter is used then a data file is created for each integrity chain (with objects) that is released for the run-time environment. Overlapping integrity chains are treated as one big chain. The name of the data file is TRF/PGM/<task>, and all file names that are part of the chain are written in that data file. The data file can be input for site-specific procedures.
USE-UNKNOWN-VERSIONS	If a copy file is already resident on disk, then this copy file is used only for the compilations if the RELEASEID of that copy file is correct (because it is the correct version). Invalid versions of copy files are changed to the "UNKNOWN/" directory. This option also uses unknown versions of copy files.
NO-CHECKED-OUT-FILES	Programs are normally copied from the repository to disk before the compilation is started. If a source is in maintenance by a developer, then the checked out version is used (only in the development environment). If this option is set, then the file is always copied from SURE and the checked out version cannot be used.
NO-EXAMINE	By default, the files and copy files are changed to the EXAMINESOURCE/= directory after the compilations. This is not done, if this option is used.
FIRST-FILE-TYPE <filetype>	This option enforces that the files with <filetype> are compiled first. This option makes it possible to use the result of the first compilations (or the result of directly started start-jobs of these first compilations) in start-jobs that are triggered by other compilations
<taskvalue>	The task value determines the amount of compilations that are executed simultaneously. This amount can be adapted during the run of RESPECT/SURE/COMPILE using the ODT command <mixno> HI <amount>. The default task value = 1.

RESPECT/SURE/COPY

This program copies a file from the repository to disk.

Input	Parameter	: —<file name>	
			AS —<copy title>
	Task string	:	<environment>.
	Task value	:	1 = Do not download the include files of the source.

Example

```
RUN RESPECT/SURE/COPY( "PROG/AA AS PROG/BB" );TASKSTRING="DEVELOPMENT"
```

Where

This program is used by the customer in self-written jobs.

RESPECT/SURE/DUMP

This program copies a group of files from the repository to disk. The group FILE , FILE-CONTROL, or PROJECT is used for the relation evaluation.

Input	Parameters	:	Two parameters: <class> and <asset>
	<class> :		—<class>— S —
	<asset> :		—<asset>— { — = —<yyyymmdd> — } — < >
			All file with relation <class>/<asset> are selected. The <timestamp> can be used to limit the selection.
	Task string	:	<environment>.
	Task value	:	999 = Dumped files do not get a release-id.

Example

```
RUN RESPECT/SURE/DUMP( "CHANGED" , "{>19980609}" );TASKSTRING = "PROD"
```

Where

This program is used by the customer in self-written jobs.

**RESPECT/SURE/ERRORLIST**

This program prints all the ERROR files from the SURE batch compilation.

Input        Directory        : All files in directory (<my-usercode>)ERRORS/= on <my-pack>  
                                 are printed.

Output      Overview        : All error files are printed.

**Example**

```
RUN RESPECT/SURE/ERRORLIST
```

Where

This program is used by the customer in self-written jobs, or in the SURE evening batch.

Security

MP +PU (the program does SYSTEMSTATUS commands).

**RESPECT/SURE/EXAMINE**

This program scans sources to create a data dictionary at the file level.

Input	Parameters	:	<div><div>&lt;file&gt;</div><div><div>MERGE</div><div>TOTAL</div></div><div>SAVE</div><div>COPYONLY</div><div>ALGOL-INTERNAL</div></div>				
	Task string	:	<environment>.				
	Task value	:	20	= Examine only for copy file statements.			

**Example**

```
RUN RESPECT/SURE/EXAMINE(" ");TASKSTRING = "PROD"
```

Where

This program runs in the SURE evening batch. The program can be started from the file-property-screen through the EXAMINE function.

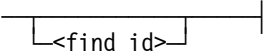
Security

MP +PU.

<file>	This parameter can be used to examine a single file instead of all files. If a <filename> is not passed then the MERGE option is true.
MERGE or TOTAL	A temporary merge file is created using the OMIT options in the source, the SURE compilation translate tables, and the copy file statements. This temporary merge file is examined. The result of the merge option is that the examine process has the same input as the compilation process, and the examine-result (= relationships in the repository) is a correct reflection of the object environment.
COPYONLY	Only the copy file statements are examined. Old copy file relationships are not removed.
SAVE	The temporary merge file is not removed after the examine process. MERGE in this situation is always true.
ALGOL-INTERNAL	If this option is set, then internal data file names of ALGOL programs are always examined. If this option is NOT set, then internal file names of data files in ALGOL programs are examined only if an external file name is found.

RESPECT/SURE/FIND

This program handles the FIND requests in the repository.

Input	Parameters	:	
	Task string	:	<environment>.
Output	Overview	:	FIND listing. All lines in all selected sources that contain one of the find targets are printed.

Example

```
RUN RESPECT/SURE/FIND( " " );TASKSTRING = "DEVELOPMENT"
```

Where

This program runs in the SURE evening batch. The program is started from the SURE-browser through the FIND function.

Security

MP +PU.

**RESPECT/SURE/FINISH**

This program informs the compile process about the result of a start job. These start jobs are often used in the bind process.

Input	Parameters	:	—<source name>—
			Refer to Section 24, "Binding of Programs," for more information.
	Task string	:	— <environment>—   <queue>
	Task value	:	0 = Report a successful run of the start-job. 1 = Report that unknown errors were found by the start-job. 2 = Report that syntax errors were found by the start-job.

**Example**

```
RUN RESPECT/SURE/FINISH( "DR/BIND1" );TASKSTRING = "PROD"
```

Where

This program runs in the SURE evening batch.

Security

MP +PU.

MP +SECADMIN (the program adds predefined securities to another object).

Parameter details:

<source-name>	The result of the start-job is linked to this source.
<queue>	If the compile-queue is not the normal queue, then the queue is passed to RESPECT/SURE/FINISH through the task string.

RESPECT/SURE/LOAD

This program loads a directory of (new) files in the repository.

Input	Parameters	:	Three parameters: <directory>, <attributes> and <object>
	<directory>	:	— (<usercode>)<directory> ON <family> —
	<attributes>	:	— PROJECT <proj> FILE-TYPE <type> —  <div><div>FUNCTION=&lt;func&gt;</div><div>OBJECT-USERCODE=&lt;user&gt;</div><div>OBJECT-PACK=&lt;pack&gt;</div><div>OBJECT-HOST=&lt;host&gt;</div><div>GENERATED</div><div>PURGE</div><div>INTEGRITY&lt;class&gt;&lt;asset&gt;</div><div>TASK   &lt;task&gt;     CURRENT     SAME-AS &lt;file&gt;</div></div>
	<object>	:	—  source-prefix=<object prefix> —  See "RESPECT/SURE/LOAD" in Section 27.
	Task string	:	<environment>.
	Task value	:	1 = Load only new files 2 = Reload only existing files & keep the attributes 3 = Reload only existing files & modify the attributes 4 = Load all files & keep attributes of existing files 5 = Load all files & modify attributes of existing files

Example

```
RUN RESPECT/SURE/LOAD(" (SG)SRC ON PK1", "PROJECT=AA,FILE-TYPE=BATCH", "" );
TASKSTRING = "DEVELOPMENT";VALUE=1
```

Where

This program can be used by the customer in self-written jobs.

Security

MP +PU (the program uses SYSTEMSTATUS commands).

<directory>	The directory with files that have to be loaded. Only source files are loaded into the database. Object-files, data-files, and dbdatafiles that are resident in the specified directory are skipped. The syntax is as follows: USERCODE ON <pack>      Select and load all files under any usercode (not *) on family <pack>. (<uscd>)<dir> ON <pack>      Select and load all files in directory (<uscd>)<dir> ON <pack>.
-------------	--



PROJECT	Each loaded file gets these attributes. <Project> and <file-type> are required attributes.
FILE-TYPE	
FUNCTION	If <object-usercode> and <object-pack> are not entered, then they are inherited from the default object-location of the application system, using the object-inheritance-option of the file-type.
OBJECT-USERCODE	
OBJECT-PACK	
OBJECT-HOST	
GENERATED	<p>If this option is used then each loaded file gets STATUS(GENERATED); otherwise, the status is set to the environment name.</p> <p>It is not possible to transfer a generated file to a next environment, because the generation process is usually dependent on the current contents of the environment.</p>
PURGE	Removes a source from disk after it is loaded in SURE.
TASK <task>	The task is linked to each loaded file.
TASK CURRENT	The current task of the usercode where RESPECT/SURE/LOAD is started and linked to each loaded file.
TASK SAME-AS <file>	<p>Each loaded source gets the same tasks as SAME-AS-&lt;file&gt;.</p> <p>In addition, the impact relations, the integrity-relations, and the integrity-prod relations are inherited from that file.</p> <p>This makes it possible to load a generated file through a start-job during the batch-compile, with combination of the parameter FIRST-FILE-TYPE or ALWAYS-MODE for RESPECT/SURE/COMPILE.</p>
<source=object>	<p>Specifies a non-standard object name for a group of files. By default, the object of source A is titled OBJECT/A. This is done automatically by SURE. It is possible to specify a non-standard object name for a file. However, this can only be done by using an online function where the non-standard object name has to be specified for each individual file. This can be a tedious task. With this option, it is possible to load the non-standard object names for all files in the specified directory.</p> <p>N.B. This is not the recommended method. A preferred way to define non-standard object-names for a group of files is through a library that determines a site-specific standard for object-names. See "Object-names" in Section 23, "Compilation and Object files."</p>

**Examples with source name SOURCE/ABC:**

<b>&lt;source/object&gt; conversion</b>	<b>Object-name</b>
"SOURCE=OBJECT"	'OBJECT/ABC'
"SOU=X/O"	'X/ORCE/ABC'
"SOURCE="	'ABC'
"=OBJ/"	'OBJ/SOURCE/ABC'
" "	'OBJECT/SOURCE/ABC'

### RESPECT/SURE/LOG

This program loads SUMLOG information into the repository. The program produces several overviews.

1. Overview of incorrectly ended (DS-ed) programs.
2. Overview of file names with the number of printed lines by this program run.
3. List of programs with a SURE compile timestamp that differs from the creation timestamp of the object file.
4. Overview of program names that are not known in SURE.

Input	Task string	:	<environment>.	
	target	:	1 = debug mode.	
	Task value	:	value DIV 10 = 0	Create overviews a and c.
		:	value DIV 10 = 1	Create overviews a, b and c.
		:	value DIV 10 = 2	Create overviews a, c and d.
		:	value DIV 10 = 3	Create overviews a, b, c and d.
		:	value MOD 10 = 1	Initialize the compile-timestamp for files without a compile-timestamp.
		:	value = 99	Filter the SUMLOG file: read-only the BOJ, BOT, EOJ and EOT records and write them in a filtered SUMLOG file.
	file	:	SUMLOG file (file equate FILE LOG = <SUMLOG/...).	
	accept	:	Only if a non-expected SUMLOG file is passed to the program.	
Output	overview	:	Four different overviews.	

### Example

```
RUN RESPECT/SURE/LOG;TASKSTRING = "PROD";VALUE = 31;
FILE LOG = *SUMLOG//1111/020403/000003 ON DLPACK;
```

Where

This program runs in the SURE evening batch.

Security

MP +PU (the program accesses files in other directories).

RESPECT/SURE/MATCH

This program creates delta files and removes unused File versions from the repository.

Input	Parameter	:	<div><div>DEBUG</div><div>PRTALL</div><div>HECK</div><div>ANPREV</div><div>DELFILE</div><div>NO-RESEQ-MERGE</div></div> <div>FILE &lt;filename&gt;</div> <div>PRINTER &lt;dest&gt;</div>
	Task string	:	<environment>.
Output	overview	:	Debug overview (optional). Listing of each created delta file (optional).

Example

```
RUN RESPECT/SURE/MATCH( " " );TASKSTRING = "DEVELOPMENT"
```

Where

This program runs in the SURE evening batch.

Security

MP +PU (the program accesses files in other directories).

<run parameter> = empty	The default that creates delta files in the repository and deletes the old delta files.
CLEANPREV	Is used if the previous FileVersions are to be deleted. If an environment does not need objects, one may choose not to run the RESPECT/SURE/COMPILE program. Not running the compile program introduces a conflict situation within SURE: the previous FileVersion (= the last good compiled version of the file) is deleted by this compile process. If the compile program never runs, then these previous FileVersions are never deleted. Therefore, omitting the compile process must be combined with an additional run of RESPECT/SURE/MATCH providing the CLEANPREV option.
CHECK	Enforces the match program to check all versions of all files that are stored in the repository. If a FileVersion is not in use anymore (= not current in any environment, not previous in any environment, and not needed for a match in any environment) then it is removed. It should not be necessary to use this option, because the regular match process does a check of all FileVersions for the files that are changed.
DEBUG	Is used for debugging purposes. It makes an overview of available delta files.
PRTALL	This option is used for debugging. It makes an extended overview of available delta files.

CLEAN	RESPECT/SURE/MATCH automatically does the housekeeping for the entire repository on every fifth day, according to the history options that are defined for each environment and on the global level. Expired records are removed during the clean action. This option enforces an immediate clean action on the repository.
DUMP_DELFILE	This option enforces the match program to dump the FileVersions files that are deleted from the repository into a directory SUREDELETE/<filename>.
NO-RESEQ-MERGE	RESPECT/SURE/MATCH does an automatic merge of patch files into the main source if the source itself and all patch files of that source are equal in all environments. The merged file automatically re-sequenced. With this option, the re-sequence is skipped.
FILE <filename>	This parameter can be used to match a single file instead of all files.
PRINTER <dest>	This parameter can be used to route the output to a print destination. If this parameter is not used, then the output is routed to the default print destination of the usercode that started the match program.

**RESPECT/SURE/REPLACE**

This program handles the REPLACE requests in the repository.

Input	Parameter	:	<div><div></div><div>&lt;replace id&gt;</div></div>
	Task string	:	<environment>.

**Example**

```
RUN RESPECT/SURE/REPLACE( " " );TASKSTRING = "DEVELOPMENT"
```

Where

This program can run in a user-written job. The program can be started from the SURE browser through the REPLACE function.

Security

MP +PU (the program accesses files in other directories).

**RESPECT/SURE/SECURE**

This program secures all in-use sources and all files that are marked as "to secure."

Input      Task string      :    This program does the secure for all environments.  
                 Task value      :    0    = The secured sources are copied to tape.  
   1    = The secured sources are copied to a dump family.  
  
Output      job                :    SURE/SECURE/SOURCES.  
   This job is started automatically and copies the sources to a  
   tape or to a backup family.

**Example**

```
RUN RESPECT/SURE/SECURE;VALUE = 1;
```

Where  
This program runs in the SURE evening batch.

Security  
MP +PU (the program accesses files in other directories).

**RESPECT/SURE/SOURCELIST**

This program prints directories of source files from the repository or from CANDE.  
Each file is printed in a SURE layout containing the MARKID that indicates when and by whom a source line was changed.

Input      Parameters      :

@ <class> <asset>  
<Cande directory>  
<filename>

DATABASE  
DIRECTORY  
PAGE  
NOPAGE  
NOSCAN  
HP  
POR132  
PAGELENGTH  
NAMES  
TYPE <0, 1 or 2>

Task string      :

File              :

Output      overview      :

<environment>.

file D1 has to be file-equated if no selection of files is given through the input parameter.

Each selected file is printed.

Example

```
RUN RESPECT/SURE/SOURCELIST( "@ REQUEST SIMON,TYPE 2" );TASKSTRING = "PROD"
```

Where

This program can run in a user-written job.

Security

MP +PU (the program uses SYSTEMSTATUS commands).

- DATABASE Files are used from the repository.
- DIRECTORY Files are used from a CANDE directory.
- PAGE A "COBOL/" or an "ALGOL \$ PAGE" results in a page skip.
- NOPAGE No page skip between two files.
- HP In case of double-sided printing.
- POR132 Special form id with 132 chars on a line.
- PAGELength The amount of lines on a page.
- NAMES Only the selected file names are printed.
- NOSCAN No index of procedures and declarations at the end of a file.
- TYPE The printer type:
  - 0 = 60 lines, 132 chars.
  - 1 = 132 lines, 132 chars, single sided.
  - 2 = 132 lines, 132 chars, double sided.

RESPECT/SURE/STARTLOG

This program scans a directory with SUMLOG files and starts a job that starts the RESPECT/SURE/LOG program for every file in the directory.

Input	Parameters	: —<directory with sumlog files>—
		— FILTER —
		└ PREFIX<prefix> ┐
		See "RESPECT/SURE/STARTLOG" in Section 27.
	Task string	: <environment>. The task string is passed to the log-loader RESPECT/SURE/LOG
	Task value	: The task value is passed to the log-loader.

Example

```
RUN RESPECT/SURE/STARTLOG( "*SUMLOG ON LOGPACK" );TASKSTRING = "PROD"
```

Where

This program can run in a user-written job.

Security

MP +PU (the program accesses files in other directories).

- FILTER

Reads the SUMLOG files on the production host and creates filtered SUMLOG files that contain all the required records. The size of such a filtered SUMLOG file is 5 to 10% of the size of the original SUMLOG file.
- PREFIX <prefix>

The default prefix of the filtered SUMLOG files is "FILTERED/". This option overrules the default prefix.

RESPECT/SURE/TRANSFER

This program deploys the compiled objects to the defined environments after compilation.

Input

Parameters

:

NORMAL-TAPE

FAST-MODE

QUEUE<queue>

VIA-TAPE

/

TRANSFERSELECTION <filename>

/

OUTPUTFILE

SPLIT-COPY nnn

/

DUMBJOB

OBJECT-PREFIX <PREFIX>

/

The default mode is NORMAL-MODE.

Task string

:

<environment>.

Job

:

SURE/TRANSFER/<pack>.

This job is automatically started.

Output

Overview

:

A list of the files that are copied to the object-environment

Example

```
RUN RESPECT/SURE/TRANSFER( "NORMAL-MODE" );TASKSTRING = "PROD"
```

Where

This program runs in the SURE evening batch.

Security

MP +PU (the program uses SYSTEMSTATUS commands).

8207 4121-000

47-25

NORMAL-MODE (default)	Selects files that are placed in the normal transfer queue.
FAST-MODE	Selects files that are placed in the transfer fast queue.
QUEUE <queue>	Selects files that are placed in the user-defined transfer queue.
DUMPJOB	Enforces to generate a dump-job for each object family/host. A dump-job copies all objects, jobs, and data files that are defined in SURE with OBJECT-PACK = <family> and OBJECT-HOST=<host> to a backup tape.
VIA-TAPE	<p>RESPECT/SURE/TRANSFER generates a deployment job for each destination family and for each host.</p> <p>By default, the deployment is done through tape: copy the objects from the SURE compilation directory to tape, and copy the files from tape to the object-location.</p> <p>It is possible to define for each family and for each host that the objects or files can be copied directly from the SURE compilation directory to the object-location (= pack-to-pack copy through library maintenance and through BNA if the destination pack is on another host).</p> <p>If it is defined that the objects are copied from pack-to-pack or through BNA, then it can be overruled with this parameter. So this parameter enforces that the deployment is done through tape for a single run of RESPECT/SURE/TRANSFER.</p>
TRANSFERSELECTION <file>	<p>Deploys only files that are in the deployment queue and are subject to a pre-defined relationship. This relationship must be defined in a file &lt;file&gt;. The relationship must be entered as class in the first line and an asset in the second line, both terminated with an "@" sign. The transfer program will then only transfer the files with this relationship. This makes it possible to transfer in "phases" (for example, first the batch programs, and later the online programs (or the rest)). If value 2 is used and the input file is not resident, the transfer program will P-DS; if it is resident, dump jobs are not created and the transfers will proceed as in value 1. This facility will not update the repository. Therefore, the transfer program must be initiated each day with a value other than 2 or 702 in order to update the production timestamps in the repository. This implies that previously transferred files are copied to their destination once again.</p> <p>The layout of the RELTRANSFER file is as follows (using the batch relation as an example):</p> <pre>1000FILE TYPE@ 2000BATCH@</pre>
SPLIT-COPY <nnn>	<p>RESPECT/SURE/TRANSFER generates a deployment job that copies the necessary objects to their object-locations. This is done using a library/maintenance copy command, and each object is mentioned individually in this command.</p> <p>This option can be used to limit the amount of objects for each library/maintenance copy. If the amount of objects to be deployed exceeds &lt;nnn&gt;, a second library/maintenance copy is started.</p>



OBJECT-PREFIX <prefix> All objects are copied to the object-location with the given prefix.

This makes it possible to put the new object automatically on the run-time environment, but to control the moment that the new objects are actually activated (through a manual copy).

RESPECT/TASK/ADD\_TRF\_QUEUE

This program is used to add tasks with a delivery date into the batch task transfer queue.

Input	Parameter	:	<div><div><div>DELIVERY</div><div>QUEUE</div></div><div><div>TASK</div><div>CURRENT</div></div><div>PRINTER</div></div> <div><div>= yyyyymmdd</div><div>&gt; yyyyymmdd</div><div>&lt; yyyyymmdd</div><div>&lt;destination environment&gt;</div><div>&lt;destination&gt;</div></div> <div>/</div>
-------	-----------	---	--

Example

```
RUN RESPECT/TASK/ADD_TRF_QUEUE("=19980604");TASKSTRING = "PROD"
```

Where

This program can be used by the customer in self-written jobs.

DELIVERY <date> Selects all tasks with a delivery date that meets the input date, and puts that task into the batch transfer queue.

QUEUE <task> Adds a task to the batch transfer queue.

RESPECT/TASK/LIST

This program creates a task overview.

Input

Parameter

:

— LIST-TYPE

OVERVIEW

OVERTIME

ALL

SETTLEMENT

ACTIVE

SOLVED

DACIMPACT

CUSTOMER

COUNTS

TRANSFERRED

DUMPED

HISTORY

RELEASE

PROJECT

ALSO-STATISTICS

GROUP

PROBLEM-TYPE

PRIORITY

ANNOUNCED-BY

RESOLUTION-BY

EMP-FUNC

DEPARTMENT

PERIOD-FROM

PERIOD-TO

STATUS

SUB-STATUS

MODE FORMS

PRINTER

NONE

Refer to Section 37, "Task," for more examples.

Task string

:

<environment>.

Output

Overview

:

Task overview.

Example

```
RUN RESPECT/TASK/LIST("OVERVIEW PROJECT SURE");TASKSTRING="DEVELOPMENT";
```

Where

This program can be started through the "Reports" menu in the SURE browser.

**RESPECT/TASK/STATISTICS**

This program creates a file with task statistics. For each solved task, the date-started and date-solved are placed in the statistics file.

Input

Parameter

:

RELEASE

PROJECT

GROUP

PROBLEM-TYPE

PRIORITY

ANNOUNCED-BY

SOLUTION-BY

RTMENT

FROM

o

<nr>

<project>

<task group>

<task type>

<priority>

<announce>

<user-id>

<department>

<from>

<until>

These parameters can be used to limit the selection of tasks

Task string

:

<environment>.

Output

File

:

OUTF.

The record layout of this statistics file is:

TASK-NAME

DATE-RECEIVED

DATE-STARTED

DATE-READY

RELEASE-NR

SOLVED-BY

X(18)

9(08)

9(08)

9(08)

X(18)

X(18)

**Example**

```
RUN RESPECT/TASK/STATISTICS("PERIOD-FROM 19990101, PERIOD-TO 19990331");
```

Where  
From CANDE.

**RESPECT/TASK/TRANSFER**

This program transfers the tasks that are queued in the task-transfer-queue to the destination environment.

**Example**

```
RUN RESPECT/TASK/TRANSFER;TASKSTRING = "<destination environment>"
```

Where  
This program is used by the customer in self-written jobs.

### RESPECT/TOOLS

This program is used to perform some support functions. This program does not access the repository.

Input	Parameters	:	HELP	<Command>
			DCKEY	<command>
			DCKEYDIR	<command>
			CREATE-FILE	<line> ; <line> ; etc
			CHECK-FILEATTRIBUTES	<directory>
			COMPARE-DIR	<directory> <directory>
			REPLACE	<replace commands>
			MIXCMD	<mix-entry> , <ODT-command>
			WAIT-RESIDENT	<file>

### Example

```
RUN RESPECT/TOOLS(" <param> " );
```

Where

This program is used by the customer in self-written jobs.

This program is used in the installation.

Security

MP +PU.

MP +SECADMIN (the program can change the security of other files).

DCKEY

This function performs a dkeyin command.

Example:

```
RUN RESPECT/TOOLS("1234 LP-")
```

DCKEYDIR

This function performs a dkeyin command for all files of a directory. If no dkeyin command is specified, then the directory is made public io.

The command must have one of the following layouts

1: <directory>

2: <directory> SYSTEM 'odt <FILE> command'

3: <directory> LIBRARY <lib> '<command>'

Examples:

1: (SIM)PRG ON PK

2: (SIM)PRG ON PK SYSTEM 'SECURITY <FILE> PUBLIC IO'

3: (SIM)PRG ON PK LIBRARY OBJECT/RIS/INSTALL/MARKFILE  
'RELEASEID RIS-49-0001'

CREATE-FILE	<p>This function creates a file.</p> <p>The destination file must be file-equated with the internal file OUT.</p> <p>Each semicolon in the parameter enforces a new record in the file.</p> <p>If the parameter is empty, then an empty file is created.</p>
CHECK-FILEATTRIBUTES	<p>This function checks the attributes of a directory of files. An overview is created that can be used to improve block factors either and or or area sizes.</p>
COMPARE-DIR	<p>This function compares the contents of the files in two directories.</p> <p>Examples of the input:</p> <p>COMPARE-DIR (SIMON)PRG ON IDRD (SAVE)PRG ON IDRD</p> <p>COMPARE-DIR PRG SAVE</p>
REPLACE	<p>This function replaces a string in a file by another string.</p> <p>The target file must be file-equated with the internal file. WORKSOURCE.</p> <p>The replace-command must have the following layout:</p> <p>R/&lt;target string&gt;/&lt;new string&gt;/</p>
MIXCMD	<p>The &lt;ODT command&gt; is given to the task with name &lt;mix-entry&gt;.</p>
WAIT-RESIDENT	<p>The program waits until one or two files are resident or not resident.</p> <p>The complete railroad diagram.</p> <div><pre>— WAIT-RESIDENT — [ NOT ] — F1 — / / [ AND OR ] [ NOT ] — F2 — / / [ DISPLAY TERMINATE ] — &lt;seconds&gt; —</pre></div>

File-equate F1 and F2 with external file names.

DISPLAY: display each <seconds> if the program is still waiting.

TERMINATE: abort after <seconds> if the program is still waiting.

**RIS/API/CVTCODE**

This program is part of the SURE Explorer interface and changes a file to an object file.

Security

MP +COMPILER.

**RIS/API/DCIS**

This program is part of the SURE Explorer interface. It provides the COMS and TCP/IP interface to the SURE API functions. This program is defined as a direct COMS window in COMS utility.

### **RIS/API/DRIVER**

This program is part of the SURE Explorer interface. It contains all SURE API functions.

Security

MP +PU (creates files in other usercode directories).

MP +TASKING.

### **RIS/API/FTP**

This program is part of the SURE Explorer interface. It handles FTP.

Security

MP +PU (creates files in other usercode directories).

### **RIS/API/LFI/00**

This program is part of the SURE Explorer interface. It deals with off-line transactions.

### **RIS/API/LFI/01**

This program is part of the SURE Explorer interface. It deals with DCIS native transactions.

### **RIS/API/LFI/99**

This program is part of the SURE Explorer interface. It provides an interface to all SURE API functions in the RIS/API/DRIVER

Security

MP +PU (creates files in other usercode directories).

MP +TASKING.

### **RIS/API/LIB**

This program is part of the SURE Explorer interface. It provides support routines for SURE API functions.

Security

MP +PU (creates files in other usercode directories).

### **RIS/API/REM**

This program is part of the SURE Explorer interface. It routes transactions from the LFI to a free DRIVER.

Security

MP +PU (creates files in other usercode directories).

MP +TASKING.

### **RIS/API/TPP**

This program is part of the SURE Explorer interface. It handles the Transaction Processing Protocol

### **RIS/BACKGROUND/EMAIL**

This program is automatically started by SURE and runs in the background. The purpose of this program is to execute all email requests that are generated by SURE.

Input            Task value        :    10        = debug mode.

Where

    This program can be run always.

Security

    MP +PU (the program accesses files in other directories).

### **RIS/BACKGROUND/EXAMINE**

This program is automatically started by SURE and runs in the background. The purpose of this program is to examine new and changed sources for include/copy statements.

Input            Task value        :    10        = Debug mode.

Where

    This program can be run always.

Security

    MP +PU.

### **RIS/BACKGROUND/IMPACT**

This program is automatically started by SURE and runs in the background. The purpose of this program is to analyze the impact of changes to Ris-entities and sources.

Input            Task value        :    10        = Debug mode.

Where

    This program can be run always.

RIS/BACKGROUND/OBJLOC

This program is automatically started by SURE and runs in the background. The purpose of this program is to check and maintain the object-locations of files.

Input Task value : 10 = Debug mode.

Where

This program can be run always.

RIS/COMPLETE/OBJECT

This program runs directly after a successful manual compilation of a source. The program updates certain program attributes of the compiled object and optionally starts a bind job for that object. The compiled object is moved to the object-location defined in SURE. The customer can use the return parameters in self-written compilation jobs.

Input	Parameters	: —<source name> —<object name>— REFERENCE —
		This is a parameter by reference.
Output	Parameter	: BINDINPLACE A SURE in-place bind has to be started for this compiled object.
		BINDTOTAL A SURE total bind has to be started.
		BINDBATCH A SURE batch bind has to be started.
		REFRESHSERVICE The compiled object is a service library and the running one must be refreshed.
		FILE-TYPE The file-type of the source.

Example

```
STRING RCOPAR
    ,SOURCENAME
    ,FILETYPE
;
RCOPAR:=SOURCENAME;

RUN OBJECT/RIS/COMPLETE/OBJECT(RCOPAR REFERENCE);
IF LENGTH(RCOPAR) > 9 THEN
    IF TAKE(RCOPAR,9) = "FILE-TYPE" THEN
        FILETYPE:=TAIL(DROP(RCOPAR,9), " ");
```

Where

This program is started in the generated compile jobs WFL/<env.>/COMPILE and WFL/<env.>/PC-SESSION

Security

MP +PU (the program uses DCKEYIN statements).

MP +SECADMIN (this program gives other objects the MP +PU status).



**RIS/COMPLETE/WORKENVIRONMENT**

This program compares all copy files, rule files (RIS), and FCT files (RIS) and RIS-include files (RIS) of a repository-environment with their versions in the CANDE work-environment. If the CANDE version is not available or invalid, then the repository version is copied to CANDE.

Input	Parameter	:	<div><div></div><div>&lt;system&gt;</div></div>
			<system>:Check only the files of this system.

**Example**

```
RUN RIS/COMPLETE/WORKENVIRONMENT(" ");TASKSTRING = "DEVELOP"
```

Where  
This program runs in a user-written job.

Security  
MP + PU (the program accesses files in other directories).

**RIS/COPY/ENVIRONMENT**

This is a new program that copies an entire environment from an existing repository to an empty repository. The "global" environment that is applicable for all environments is copied too.

Input	Parameter	:	<div><div></div><div>DUMP</div><div>LOAD</div><div>SET-ENVIRONMENT</div></div>
	Task string	:	<environment> that you want to copy
Output	Data files	:	DUMP/GLOBAL DUMP/DMSG DUMP/DINFO DUMP/DFIL DUMP/DREL DUMP/DFMT DUMP/DSCR DUMP/DUSER DUMP/DNAM DUMP/DSTOR DUMP/DITEM
	Task value		Only applicable if the parameter = DUMP 0 or 99: Dump all datasets 3: Dump dataset DMSG into file DUMP/DMSG 4: Dump dataset DINFO into file DUMP/DINFO 5: Dump dataset DREL into file DUMP/DREL Dump dataset DSTOR into file DUMP/DSTOR Dump dataset DFMT into file DUMP/DFMT

	Dump dataset DSCR into file DUMP/DSCR
	Dump dataset DNAM into file DUMP/DNAM
6:	Dump dataset DFIL into file DUMP/DFIL
7:	Dump dataset DUSER into file DUMP/DUSER
8:	Dump dataset DITEM into file DUMP/DITEM
9:	Dump global dataset into file DUMP/GLOBAL

### Example

```
RUN RIS/COPY/ENVIRONMENT( "DUMP" );TASKSTRING = "DEVELOPMENT"
```

DUMP	Dumps the contents of the environment to data files. The task value determines the data files that are created.
LOAD	Loads the data files into the target repository. The target repository must be an initialized INFDB (but this will not be checked, because it is possible to re-load the datasets after an abort). The DUMP/= files that are visible will be loaded. If the program was started with a task string, then that environment name will be activated in the new repository.
SET-ENVIRONMENT	The environment in the task string will be activated in the new repository.

### Where

This program can run in a user-written job.

### Security

MP + PU (the program accesses files in other directories).

## RIS/INSTALL

The installation program for the SURE software.

Input	online	:	Various installation parameters.
Output	job	:	WFL/RIS/INSTALL
			This job does the installation of the software.

### Example

```
RUN RIS/INSTALL
```

### Where

This program runs in the installation procedure.

## RIS/INSTALL/MARKFILE

This library sets the RELEASEID attribute for a set of files.

### Security

MP +PU (accesses files in other usercode directories).

**RIS/INSTALL/LIBRARY**

This library supports routines for the installation procedure.

Where

    This program runs in the installation procedure.

**RIS/MENU**

This program provides the terminal emulation end user interface to the RIS and SURE software.

Input	screen	:	remote file for user interface.
	Task string	:	<environment>
Output	screen	:	remote file for user interface.
	overview	:	various overviews.

**Example**

    RUN RIS/MENU

Security

    MP +PU (uses DCKEYIN and GETSTATUS directory scan).

    MP +CONTROL (standard a control status for online programs).

    MP +TASKING (the program can change its running user code).

**RIS/PRINT/TRANSFORMS**

This library contains various printer transform procedures that are used to print correctly on a remote printer. This library is also used by the document system.

### Example of usage

Consider a remote line printer of type HP LASERJET 4 PLUS.

The name of the device is REND.

The printing system command PS DEV REND returns the following information:

```
Devices in group REND:
Station IP100_0_4_1/S40012
  Status:      Idle
  Device Type: Remote Line Printer
  AutoConnect: BYSYSTEM
  Blocksize:   2950 bytes
  Buffers:     Unspecified (Default to 2)
  Accessibility: By DESTINATION only
  VFU:        Present on device
  Request Limits: None
  Header/Trailer: Headers suppressed; trailers print
  Driver:      "PCL132S IN OBJECT/RIS/PRINT/TRANSFORMS ON DISK"
  Printerkind: LINEPRINTER
  Available Fonts: None, no PDL PRINTERKINDs are configured
```

In this case, the printer transform procedure PCL132L is used. This transform makes it possible to print 132 lines of 132 characters on a A4 portrait paper.

The printer destination that is linked to a usercode in SURE is in this example REND.

### RIS/PUTLINE

This program is used to store status information in the technical info in the repository.

Input	Parameters	: Three parameters: <info-owner>, <info-class> and <text>: — Info-owner — = —<owner>————— — Info-class — = —<class>————— — text — = —————<text>————— └ <RIS-entity> @ ─┘
		If no <RIS-entity> is given, then RIS-entity is "USER." This is text added to a usercode.
	Task string	: <environment>
	Task value	: 0 = Do not clean the existing info for this owner, class, entity, environment. 1 = Clean the existing info before the new info is added.

**Example**

```
RUN RIS/PUTLINE("MYUSER","TECHNICAL-INFO","USER@Line 1 of the info");VALUE=1;  
RUN RIS/PUTLINE("ITEM01","INFORMATION","DATAMODEL@line 2 of test info");
```

**Where**

This program is used in various SURE jobs.

**Security**

MP +CONTROL (this program runs very often and for a very short time).



## Section 48

# Command Index

This section describes the available SURE commands in the terminal emulation interface.

Command ACTIVATE is used to activate a user-defined compile queue.

— ACTIVATE —<queue>—————|

Command ACT is used to reactivate a logically deleted file.

— ACT —<filename>—————|

Command (multi) ASSIGN is used to assign a file to a user.

— ASSIGN —|  
          |<filename>|—————|

Command (multi) ASSIGNIT is a combination of REQUEST and ASSIGN.

— ASSIGNIT —|  
              |<filename>|—————|

Command BACK is used to start the Unisys backup processor from SURE.

— BACK —————|

Command BYE is used to leave SURE.

— BYE —————|

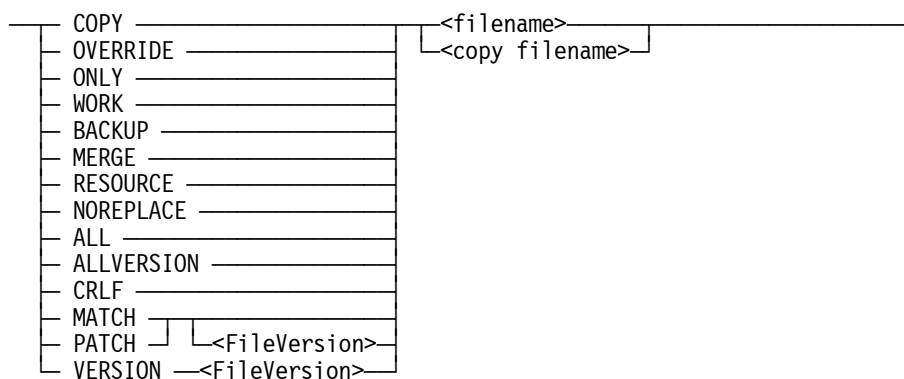
Command (multi) COMPILE is used to put one or more files in a compile queue.

— COMPILE —|  
          | NORMAL —|  
          | FAST —|  
          |<date> —|  
          |<queue> —|  
          | BLOCK —|  
          | UNBLOCK —|  
          |<filename>|—————|

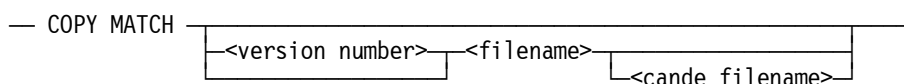
Command COMPILESTATUS shows the contents of compile queues.

— COMPILESTATUS —|  
                  |<system name> —|  
                  |<project name>|—————|

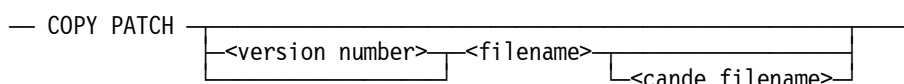
Command COPY is used to copy a file from SURE to disk.



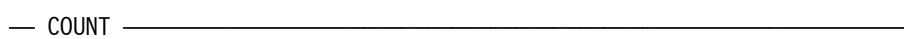
Command (multi) COPY MATCH is used to match the version of a delta file from SURE to disk.



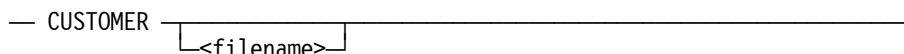
Command (multi) COPY PATCH is used to patch the version of a delta file from SURE to disk.



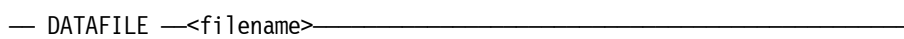
Command (multi) COUNT is used to count files that meet a selection expression.



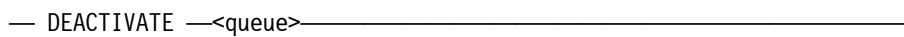
Command CUSTOMER is used to maintain the customer relations of a file.



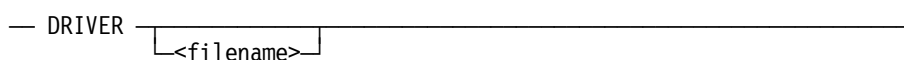
Command DATAFILE shows the data files used by a program.



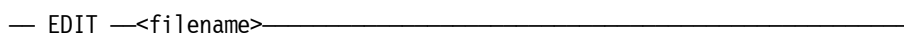
Command DEACTIVATE is used to deactivate a user-defined compile queue.



Command DRIVER is used to maintain the driver relations of a file.



Command EDIT is used to edit a file using the Unisys SYSTEM/EDITOR (U ED).





Command END can be used to leave SURE.

— END —————|

Command ENTER is used to enter a new file name.

— ENTER —————|  
           └─<filename>┐ └─<attribute>┐

Command EXAMINE is used to examine a file (in the background).

— EXAMINE —————|  
           └─ TOTAL ─┐  
           └─ MERGE ─┐  
           └─ COPYONLY ┐

Command EXT is used to update attributes of a file.

— EXT —————|  
   └─ EXTENSION ┐ └─<FileName>┐

Command FIND is used to add files in a find queue and start a find process.

— FIND —————|  
           └─ UPD ─┐  
           └─ CLEAR ─┐  
           └─ DEL ─┐ └─<find-id>┐  
                   └─ ? ─┐

Command GET is used to get a file from SURE.

— GET —————|  
           └─ OVERRIDE ─┐  
           └─ ONLY ─┐  
           └─ DOMAIN <emp func> ┐

Command GO is used to jump to another RIS function or to leave SURE (GO END).

— GO —<module>—————|

Command GUARD is used to define guard file information.

— GUARD —————|  
           └─<FileName>┐  
           └─<database name>┐

Command (multi) INQ is used to show a file or present the SELECT screen.

— INQ —————|  
           └─<filename>┐

Command INTEGRITY is used to show the reason why a file is waiting for integrity.

— INTEGRITY —<filename>—————|

## Command Index

---

Command LANGUAGE is used to change the internal language of SURE.

— LANGUAGE —<language char> —————  
                                  └──┬──┘

Command (multi) LINESCOUNT is used to count the amount of lines of a group of file.

— LINESCOUNT —————  
└──┬──┘  
  LINESCNT

Command LINK is used to link a file to the current task.

— LINK —<filename>—————

Command LIST is used to list a file or delta file.

— LIST —————→  
└──┬──┘  
  └──<sequence number>┘ └── DUMP —————  
  └── MATCH ┘  
  └── PATCH ┘ └──<version number>┘ └──<sequence number>┘  
                                  └── ALL ─────────┘ └── D ─────────┘  
└── LISTMATCH ┘  
└── LISTM ┘ └──<version number>┘ └──<sequence number>┘  
                                  └── ALL ─────────┘ └── D ─────────┘  
→<filename>—————

Command LISTING is used to look at the last compile listing of a file.

— LISTING —<filename>—————

Command (multi) LISTING-OPTION is used to define extended listing options.

— LISTING-OPTION —————  
                                  └──<filename>┘

Command LOAD is used to start the batch loading process.

— LOAD —<file directory>—<project>—<file-type>—————  
→└──<object>┘ └──<author>┘ └──<function>┘

Command LOG is used to show the files that are deleted from SURE.

— LOG —————

Command MAKEMATCH is used to create a temporary delta file for inquiry purposes.

— MAKEMATCH —<FileName>—————

Command MULTIOBJECT is used to define multiple object names for one file.

— MULTIOBJECT —————  
                                  └──<filename>┘

Command OPERATOR is used to update operator information of a file.

— OPERATOR —<FileName>—————|

Command PATCH is used to connect a manually written patch file to a source.

— PATCH —————|  
           └─<filename>—┘

Command (multi) PRINT is used to print a selection of files through the select mechanism.

— PRINT —————|

Command (multi) PRINTALL is used to print all attributes of a selection of files.

— PRINTALL —————|

Command PUR is used to delete a file physically.

— PUR —————|  
           └─ MATCH ─┘ ─<file name>—

Command PURGE is used to purge a compile queue.

— PURGE —————|  
           └─ NORMAL —┘  
           └─ COMPILE-FAST —┘  
           └─ RECOMPILE —┘  
           └─<queue>—┘

Command QUEUE is used to change the current compile queue.

— QUEUE —<compile queue>—————|

Command REC is used to recover a physically deleted file.

— REC —<filename>—————|

Command REF is used to show the user-selected references of a file.

— REF —<filename>—————|

Command (multi) RELATE is used to modify relations.

— RELATE —————|  
           └─<FileName>—┘

Command REM is used to remove a file logically.

— REM —————|  
           └─ MATCH ─┘ ─<file name>—

## Command Index

---

Command RENAME is used to rename a file or delete a file name physically.

— RENAME — 

<FileName>
------------

Command REPLACE is used to add files to the replace queue and start the replace process.

— REPLACE — 

UPD
CLEAR
DEL

<replace-id>
?

Command (multi) REQUEST is used to request a file for maintenance.

— REQUEST — 

<filename>
------------

Command RESEQ is used to resequence a file.

— R — 

+ <increment>
<base>
+
<increment>

Command RESET is used to undo a “check-out” action.

— RESET — 

REQUEST
ASSIGN
ASSIGNIT
GET
GETIT
COMPILE

 <FileName>

Command RESET COMPILE is used to delete a file from a compile queue.

— RESET — 

NORMAL
FAST
RECOMPILE
<queue>

 <file name>

Command RESOURCE is used to modify resource information of a file.

— RESOURCE — 

VERSION
TABLE
DELETE
HELP

 <FileName>

Command RUN is used to start an object from SURE.

— RUN — <filename> 

<parameters>
--------------

Command SAVE is used to save a file into SURE.

— SAVE —————>  
           └─ AGAIN ─┘  
           └─ REPLACED ─┘

Command SECURE is used to add a file or a file directory to the secure list.

— SECURE —<FileName or directory>—————>

Command STARTJOB is used to maintain the start-job relations of a file.

— STARTJOB —————>  
           └─<filename>┘

Command STATUS is used to show the status of a file.

— STATUS —<filename>—————>

Command STOP is used to terminate the online program.

— STOP —————>

Command TASK is used to enter the SURE task utility.

— TASK —————>  
   └─ PROBLEM ─┘

Command UPD is used to update the file attributes that are listed on the SURE main screen.

— UPD —————>  
   └─ UPDATE ─┘

Command UPDATEOK is used to introduce entirely new file attributes in SURE.

— UPDATEOK —<screen>—————>

Command UPDDELIVERY is used to update delivery information of a file.

— UPDDELIVERY —<filename>—————>

Command UPDINFO is used to update the information of a file.

— UPDINFO —<filename>—————>

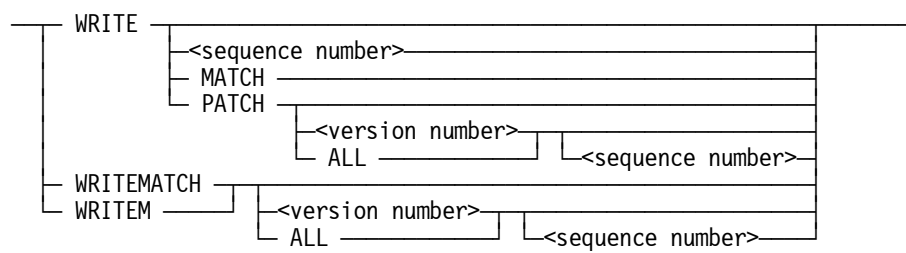
Command UPDTECH is used to update the technical information of a file.

— UPDTECH —<FileName>—————>

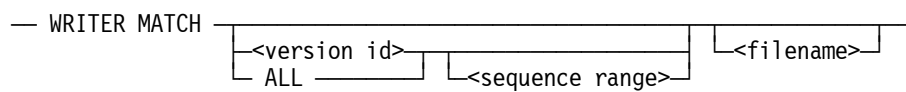
Command VIEW is used to view a file (no updates) through the Unisys SYSTEM/EDITOR.

— VIEW —<FileName>—————>

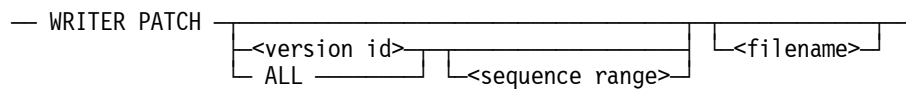
Command WRITE is used to print a file or delta file on a printer.



Command (multi) WRITE MATCH is used to print delta files.



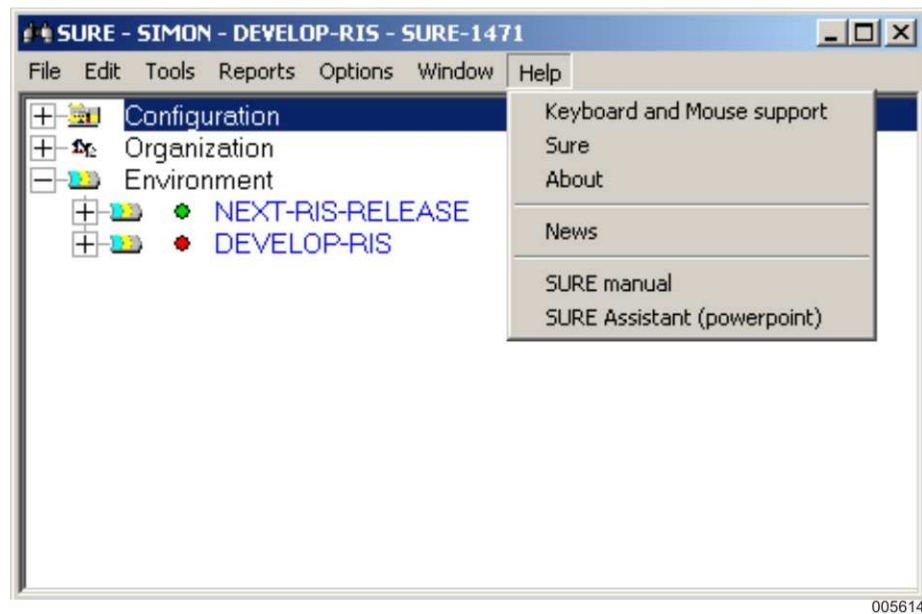
Command (multi) WRITE PATCH is used to print patch versions of delta files.



## Section 49

# Help Files

The SURE software includes many help files with additional information.



### Keyboard and Mouse Support

This help file contains:

- Data entry window support
- Keyboard Support
- Mouse Support
- Drag and Drop operations in the SURE browser
- Explanation of the icons that are used in the SURE browser

### **SURE**

The technical SURE documentation (this manual) in HLP format.

### **SURE Manual**

The technical SURE documentation (this manual) in PDF format.

### **SURE Assistant**

PowerPoint presentation describes the execution of SURE functions with examples.



## Section 50

# Software Delivery Old Style

The CUSTOMER function is used to maintain customer related information. Using the functions available on the "maintain customer definitions" screen and the batch program RESPECT/SURE/DELIVERY, you can deliver files to a customer in an organized manner.

### Screen: Maintain Customer Definitions

InterCom Terminal Emulation - [A40NTW - TA14021]

Spcfy back SURE ( Maintain Customer Info ) infra design

ITEM INFO RELATION FILE CUSTOMERS UPD-CUST-INFO DASDL-VERSIONS HELP

Function ▶ ◀  
Filename ▶ ◀  
Customer ▶ ◀ Version ▶ ◀  
Dasdl-version ▶ ◀

Function 1: Show all customers of this file  
2: Show all files with this customer  
3: Show customer-versions file, or dasdl-version customer  
4: Add a new customer  
5: Remove a customer totally  
6: Add relationship between file and customer  
7: Remove relationship between file and customer  
8: Remove all customer-relationships of the file  
9: Update customer-version file, or dasdl-version customer  
10: deliver empty copy-file to this user

22 4 Pg=4 TA14021 Form Rev LTAI

005615

### On-line Interface

The CUSTOMER function is available on the SURE main screen and a continuation screen displays the following features:

- Show all customers of the specified file. The relationship between the file and the customer is added with the aid of Function 6. This information is a subset of the information given by the function specified on the RELATION menu.
- Show all files associated with this customer. This information can be obtained by using the function INQ FILES CUSTOMER (<names>) (the INQ-SELECT function).
- Show all customer-versions of this file. This function shows:
  - the current version number of the file (relation INQ-VERSION)
  - the version number of the file at delivery time for each customer.
- If the file has not been delivered to a customer, there is no customer-version relation.
- Add a new customer.
- Remove a customer. This function removes the customer and references to a specific customer.
- Add a relation between a file and a customer. This function defines a file to be delivered to a specific customer. If the file must be delivered to all customers, the word "DEFAULT" must be entered in the customer field.
- Remove the relation between the file and the customer. All relations between the file and the customer (and also the version-relation) are removed.
- Remove relations between the file and all customers. This function can be used if the program is not in use. All customer-version relations for the file are also removed.
- Update the customer-version of this file.
- Deliver an empty file to this user. This function is used to deliver a copy file that is needed for compilation, but does nothing for this customer.

### Valid Specify Codes

CUSTOMERS	Show all customers.
ITEM	Request last misspelled item.
FILE	Request last misspelled file.
UPD-CUST-INFO	Enter customer description (customer documentation).
INFO	Show customer description.

To list or print all file-customer relations for all the files, the SELECT-screen can be completed as follows:

```
select-type      [ FILES          ]
select-criteria  [ INQ-VERSION( ) ]
```

You can also make more complex queries. For example, if you want to list all the files delivered to customer A, do the following:

```
select-type      [ FILES          ]
select-criteria  [ CUSTOMER(A) OR CUSTOMER(DEFAULT) ]
```

### Relations

Relations consists of three parts: OWNER, CLASS, and ASSET.

OWNER = A number given to a FileName.

CLASS = A number given to a FileName or an itemname.

ASSET = A number given to a FileName or an itemname.

These file and item numbers are always positive integers. However, the file-customer-version relations are built in the following way:

OWNER = The negative of the number given to the FileName.

CLASS = The number given to the customer-name (item).

ASSET = The number given to the version-number (item).

Using this construction, it is possible to show all customer-version relations for this file only. But, the customer-version relations for this file will not appear if a specific function on "RELATION" is used.

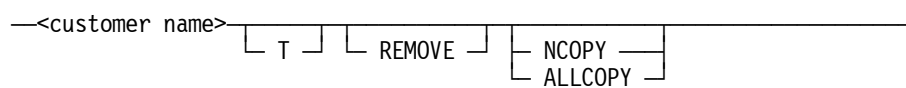
**Note:** You cannot manipulate the relations by using the RELATE function.

The current version number of the file is shown on the main menu. This version number is kept in the STOR relation of the file.

### Batch Interface

The RESPECT/SURE/DELIVERY("<customer-name> options") program selects the sources to be delivered to a specified customer. The program copies all selected sources from the database to disk and generates a job that copies the selected sources from disk to tape and removes the sources from disk. The program generates a second job that compiles all selected sources. This compile-job is also delivered to the customer.

The following railroad-diagram illustrates the use of the parameter:



<customer name>	=	Customer name for the delivery.
T	=	Test run. Only a report is given; no updates are made in the repository and no WFLs are generated.
Remove	=	Files are removed from the disk after delivery.
Nocopy	=	Copied files are not to be delivered.
Allcopy	=	By default, a changed copy file is not delivered if any of its master files were changed. When one of the master file is changed and delivered, the changed copy files are delivered. This option can be overwritten by start option "ALLCOPY." In this case, all changed copy files are delivered.

When the program is started with value = 0 (default), it will select all sources with the relation:

```
<filename>:CUSTOMER(<customer name>) or <filename>:CUSTOMER(DEFAULT)
```

When the program is started with value = 1, it asks for a <class> and an <asset>. These names can be entered by <mix>AX messages. All files with the relation:

<filename>:<class>(<asset>) are selected for delivery.

If the program is started with value = 10 or 11, test sources are delivered when the file contains the relation <filename>:TEST(<customer-name) and does not have production status. The last saved version is selected for delivery instead of the backup version. Value=10 is treated as value=0 and value=11 is treated as value=1.

A selected file is delivered when the customer-version number is less than the production-version number. In this case, the customer-version number is updated to the production-version number, and the file is copied to the disk.

The program generates a job that copies the files from disk to tape:  
SURE/DELIVERY/<customer name>.

The program generates a second job that copies the files from tape to disk and compiles the files: SURE/CUSTOMERJOB/<customer-name>.

The program produces a report containing the following information:

- The delivered files with the version and reason.
- The undelivered files with the version and reason.
- The delivered files with no customer relation.
- The undelivered files with no customer relation.
- The delivered files with DASDL reason.

When the TECH-INFO is filled in and the first sentence is "DELIVERYINFO:" this information is also printed on the delivery list.

The generated compile-job is copied to tape. This compile-job handles COBOL and ALGOL sources. The customer should copy this compile-job from tape to disk and start the job. The job then copies all files from tape to disk and starts the compilations one at a time.

### **Restrictions**

All update functions (functions 4 - 10 and UPD-INFO) are secured with the CUSTOMER authorization.

If an update is not allowed, function descriptions 4 to 10 do not appear on the screen.

The FileName must exist in the repository.

Except for function 4, the customer name must also exist.





