# TEAMCONNECTION INTEROPERABILITY WITH MVS AND SCLM

Document Number TR 29.2231

Mike Petersen
Jeri Petersen

TeamConnection Integration
IBM Software Solutions
Research Triangle Park, N.C.

(c) Copyright International Business Machines Corporation 1997.
All rights reserved.

# TRADEMARKS AND SERVICE MARKS

The following terms are trademarks or service marks of the IBM Corporation in the United States or other countries:

IBM

ISPF

MVS

OS/2

SCLM

TeamConnection

The following terms are trademarks or service marks of other companies:

| | |
|---|---|
| Object Store | Object Design, Inc. |

# ABSTRACT

This report describes techniques for interoperability between TeamConnection, MVS, and SCLM (Software Configuration and Library Manager) for application development. These techniques will help the application developer to:

run TeamConnection commands on a workstation from MVS

run TeamConnection commands on a workstation from an SCLM BUILD

run SCLM commands on MVS from a TeamConnection BUILD

You might find TR 29.2196 "USING REXX COMMAND FILES WITH TEAMCONNECTION MVS BUILD SCRIPTS" to be a useful companion document. This document contains details regarding use of REXX command files on MVS for running compilers, linkage editors, and other products from a TeamConnection BUILD. The techniques described in this report for using SCLM with TeamConnection BUILD are not covered in TR 29.2196.

The techniques in this report demonstrate:

how to change TeamConnection parts and run a TeamConnection BUILD from MVS:

- use TeamConnection CHECKOUT to lock a part in the TeamConnection data base
- edit a copy of the part in MVS
- use TeamConnection CHECKIN to update the part in the TeamConnection data base
- use TeamConnection BUILD to build the application

how to coordinate a BUILD between TeamConnection and SCLM:

- construct an architecture definition member in SCLM that will result in a TeamConnection BUILD from the SCLM BUILD
- construct a part in a TeamConnection build tree that will result in an SCLM BUILD from the TeamConnection BUILD

how to setup a TeamConnection command line interface from MVS:

- run TeamConnection commands from SCLM
- run TeamConnection commands from MVS

## ITIRC KEYWORDS

Build agent

Build processor

Build server

Configuration Management

Library Management

MVS

REXX

SCLM

TeamConnection


## Acknowledgements

The author would like to acknowledge the following people who helped with developing the techniques presented in this report:

Gary Greiner

Marv Knight

Marsha O'Brien

Matt Seabold

Beth Shelly

Alain Vandermeersh

Xavier Veroughstraete

# ABOUT THE AUTHORS

Mike Petersen is an IBM software developer for TeamConnection Integration in Raleigh, North Carolina. Mike was an SCLM software developer for six years. He has worked in systems, application, and other product programming areas. He also has had assignments with world trade, IBM faculty loan program, technical education, and technical support for marketing.

Mike received a BS in mathematics with a physics minor in 1972 and an MS in computer science in 1974 from Purdue University. In 1979 he received an MBA from Marist College.

Jeri Petersen is an IBM software developer for TeamConnection Integration in Raleigh, North Carolina. Jeri was in VisualAge Generator development for nine years. She has also worked in application development for IBM, as well as for other companies.

Jeri received a BS in mathematics with a physics minor in 1972 from Purdue University, an MS in computer science in 1974 from Rutgers University, and an MBA in 1979 from Marist College.

# CONTENTS

**TRADEMARKS AND SERVICE MARKS** . . . . . iii

**ABSTRACT** . . . . . v

**ABOUT THE AUTHORS** . . . . . vii

**Figures** . . . . . xiii

**BEFORE WE BEGIN** . . . . . 1
Disclaimer - Notice to Users of the Source Code Examples . . . . . 1

**OVERVIEW** . . . . . 3

**MVS to TeamConnection** . . . . . 5
MVS to TeamConnection - CHECKOUT . . . . . 6
MVS to TeamConnection - CHECKIN . . . . . 9
MVS to TeamConnection - BUILD . . . . . 11
Putting it all Together . . . . . 13
Setup Requirements . . . . . 13
Step 1 - Workstation preliminary activities . . . . . 13
Step 2 - MVS preliminary activities . . . . . 14
Step 3 - TeamConnection CHECKOUT issued from MVS . . . . . 16
Step 4 - Change the part in MVS . . . . . 17
Step 5 - TeamConnection CHECKIN issued from MVS . . . . . 17
Step 6 - TeamConnection BUILD issued from MVS . . . . . 19
Step 7 - Test the OS/2 Application . . . . . 19
Hints and Tips for Debugging . . . . . 19

**SCLM to TeamConnection** . . . . . 21
SCLM Workstation BUILD - How it Works . . . . . 21
SCLM Workstation BUILD - TeamConnection Commands . . . . . 22
SCLM to TeamConnection - CHECKOUT . . . . . 23
SCLM to TeamConnection - CHECKIN . . . . . 24
SCLM to TeamConnection - BUILD . . . . . 24
Putting it all Together . . . . . 24
Setup Requirements . . . . . 24
Notes . . . . . 25
Step 1 - Workstation preliminary activities . . . . . 25

Step 2 - MVS preliminary activities . . . 25
Step 3 - TeamConnection CHECKOUT issued from SCLM BUILD . . . 26
Step 4 - Change the part in SCLM . . . 30
Step 5 - TeamConnection CHECKIN issued from SCLM BUILD . . . 32
Step 6 - TeamConnection BUILD issued from SCLM BUILD . . . 33
Step 7 - Test the OS/2 Application . . . 36

**TeamConnection to SCLM** . . . 39
TeamConnection to SCLM - BUILD . . . 39
Setup Requirements . . . 39
Step 1 - Workstation preliminary activities . . . 39
Step 2 - MVS preliminary activities . . . 40
Step 3 - Create TeamConnection builder and parts for an SCLM BUILD . . . 41
Step 4 - Connect TeamConnection parts in a build tree . . . 43
Step 5 - Create an SCLM controlled member. . . . 45
Step 6 - Create an SCLM architecture definition member . . . 46
Step 7 - Run an SCLM BUILD from a TeamConnection BUILD . . . 47
Step 8 - Examine SCLM build map . . . 48

**Coordinated TeamConnection and SCLM BUILD** . . . 49
TeamConnection BUILD from SCLM . . . 49
SCLM BUILD from TeamConnection . . . 51
BUILD from SCLM or TeamConnection . . . 55
Marking SCLM members out of date . . . 55
Marking TeamConnection parts out of date . . . 56
Doing the builds from SCLM . . . 56
Doing the builds from TeamConnection . . . 56
Summary . . . 56

**SCLM Architecture Definition Parameters for TeamConnection Commands** . . . 61
Additional Parameters Recognized by the Extended FLMLTWST . . . 61
Parameters to Avoid File Transfers . . . 63
General Purpose ACTION in the ACTINFO File . . . 64
Using the ACTION value in FLMTRNSL . . . 65
Using the Extended FLMLTWST with ISPF . . . 66

**Appendix A. TeamConnection Parts and Scripts** . . . 69
Demonstration Materials . . . 70
Simple C++ Programs . . . 71
Simple C++ Program Include . . . 71
SCLM Build Command Parameters . . . 71
C++ Build Script . . . 72
Linkage Edit Build Script . . . 75

**Appendix B. TeamConnection Family Setup** . . . . . 81
Administration . . . . . 82
Preliminary Activities . . . . . 82
Create a New Family . . . . . 83
Start the TeamConnection Client . . . . . 85
Create Components . . . . . 87
Create a Release . . . . . 87
Create Builders . . . . . 88
Create Parsers . . . . . 90
Optional - Save the Family Database . . . . . 90
Open a Feature and Create a Work Area . . . . . 91
Create Parts . . . . . 92
Create Build Tree . . . . . 93
Start the OS/2 Build Agent and Build Processor . . . . . 95
Build the OS/2 Application . . . . . 96
Test the OS/2 Application . . . . . 97
Integrate the Work Area to the Release . . . . . 97
Create a Work Area for Application Development . . . . . 98
Optional - Save the Family Database . . . . . 98

**Appendix C. ISPF C/S Workstation Installation Using TCP/IP** . . . . . 99
Workstation preparations . . . . . 99
MVS . . . . . 99

**Appendix D. SCLM Project Definition Setup** . . . . . 103
Create USERINFO and ACTINFO files . . . . . 103
Create SCLM Language Definitions and Project Definition Data Sets . . . . . 104
Create the SCLM Project Definition Member . . . . . 104
Assemble and Link the Project Definition . . . . . 105
Allocate the Data Sets Controlled by the SCLM Project Definition . . . . . 106
Create the VSAM Cluster for the SCLM Accounting File . . . . . 106

**Appendix E. TeamConnection Build Scripts and Parts for SCLM** . . . . . 127

**Appendix F. SCLM FLMLTWST Extensions for Parameters** . . . . . 135
Additional Keyword Parameters . . . . . 136
Avoiding Transfer of Files . . . . . 137
Parameter for a CPARM Value . . . . . 138

# FIGURES


1. Syntax - MVS REXX programs to run TeamConnection commands . . . . . 5
2. tcchkout.rex TeamConnection CHECKOUT from MVS . . . . . 7
3. tcchkin.rex TeamConnection CHECKIN from MVS . . . . . 10
4. tcbuild.rex TeamConnection BUILD from MVS . . . . . 12
5. TSO builder information for mvssclm family . . . . . 41
6. Build tree with part sclmbild.prm . . . . . 44
7. Referencing TeamConnection build trees from SCLM architecture definitions . . . . . 50
8. Referencing SCLM architecture definitions from TeamConnection build trees . . . . . 53
9. Example of building from SCLM or TeamConnection . . . . . 57
10. ACTINFO - general purpose ACTION . . . . . 64
11. FLMLANGL and FLMTRNSL for General Purpose ACTION . . . . . 65
12. testpart.c C++ Program . . . . . 71
13. testcopy.h C++ Include . . . . . 71
14. sclmbild.prm SCLM BUILD command parameters . . . . . 71
15. fhbocomp.cmd C++ Build Script . . . . . 72
16. fhbolink.cmd Linkage Edit Build Script . . . . . 76
17. OS2 builder information for mvssclm family . . . . . 89
18. Build Tree for testpart.exe . . . . . 94
19. vsamallc.jcl SCLM VSAM account file allocation . . . . . 108
20. userinfo.txt SCLM TeamConnection USERINFO file . . . . . 109
21. actinfo.txt SCLM TeamConnection ACTINFO file . . . . . 110
22. dummy.lng SCLM DUMMY Language Definition . . . . . 114
23. checkout.lng SCLM TeamConnection CHECKOUT Language Definition . . . . . 115
24. checkin.lng SCLM TeamConnection CHECKIN Language Definition . . . . . 119
25. tcbuild.lng SCLM TeamConnection TCBUILD Language Definition . . . . . 122
26. sclmtc.prj SCLM Project Definition . . . . . 125
27. CLIST to concatenate a library containing FLMLTWST to SYSEXEC . . . . . 126
28. sclmcmd.jcl - build script to run MVS REXX programs . . . . . 127
29. sclmcmd.rex - MVS REXX program to issue SCLM BUILD . . . . . 128
30. exflmcmd.rex - MVS REXX program to issue SCLM flmcmd . . . . . 132
31. Sample JCL Using LE/370 for RUNPGMT . . . . . 133
32. FLMLTWST - adding keyword parameter defaults . . . . . 136
33. FLMLTWST - avoiding transfer of input files . . . . . 137
34. FLMLTWST - avoiding transfer of output files . . . . . 137
35. FLMLTWST - CPARM value replaced by parameter value . . . . . 138

# BEFORE WE BEGIN

This report is written for:

MVS users who need a TeamConnection command line interface for tools

SCLM users who need to run TeamConnection commands (specifically BUILD)

TeamConnection users who need to run SCLM commands (specifically BUILD)

The MVS section is independent of the SCLM section although what happens "behind the scenes" is the same. You do not need to know SCLM to use the MVS section. (SCLM users typically have some MVS knowledge.) The MVS and SCLM users do not need extensive knowledge of TeamConnection to use the techniques. Similarly, TeamConnection users do not need extensive knowledge of MVS and SCLM to use the techniques.

In order to run the examples in this report you will need to:

have access to an MVS system with ISPF 4.2 (or newer) and the TeamConnection MVS build processor

have access to a workstation with TeamConnection Version 2.0 (or newer)

connect your MVS system to your workstation using the ISPF Client/Server function through TCP/IP or APPC

create a simple TeamConnection family with components, releases, work areas, parts, builders, and build trees (Appendix B, “TeamConnection Family Setup” on page 81 contains the instructions for this)

create REXX programs from examples in the appendices (a few changes to part, file, and data set names may be required)

create a simple SCLM project definition with language definitions, architecture definitions, and members for the SCLM samples (Appendix D, “SCLM Project Definition Setup” on page 103 contains the instructions for this)

## DISCLAIMER - NOTICE TO USERS OF THE SOURCE CODE EXAMPLES

THIS TECHNICAL REPORT PROVIDES THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH YOU.

SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

# OVERVIEW

The general idea for the use of TeamConnection from MVS and SCLM is to use the file transfer, workstation command, and client/server features of ISPF 4.2. The MVS user can use REXX command files to transfer data to and from the workstation as well as run TeamConnection commands on the workstation.

The SCLM user can use the workstation build techniques of ISPF 4.2 with TeamConnection commands to also transfer data to and from the workstation as well as run TeamConnection commands on the workstation.

The general idea for the use of SCLM from TeamConnection is to have a part in the build tree whose builder puts a build event on the job queue for the MVS build processor. This build event directs the build processor to run IRXJCL which then runs a REXX command file that contains an SCLM command.

A coordinated TeamConnection and SCLM BUILD can be accomplished: 1) from TeamConnection by using a part in the TeamConnection build tree with a builder that results in running an SCLM BUILD command, or 2) from SCLM by using a source member in an SCLM controlled library with a language definition that invokes a TeamConnection BUILD command.

Each major section explains the technique and then contains a "Putting it All Together" that walks you through the steps to try out the technique.

The following technical reports might be helpful in understanding these techniques:

TR 29.2196 USING REXX COMMAND FILES WITH TEAMCONNECTION MVS BUILD SCRIPTS

TR 29.2147 SCLM GUIDE TO TEAMCONNECTION TERMINOLOGY

TR 29.2034 Using ISPF's Software Configuration and Library Manager (SCLM) to Manage Workstation and Cooperative Applications

# MVS TO TEAMCONNECTION

The connection between MVS and the TeamConnection client or family server machine is through the ISPF 4.2 Client/Server function. The ISPF Client/Server software must be installed on the workstation in order for ISPF to communicate with the workstation. See Appendix C, “ISPF C/S Workstation Installation Using TCP/IP” on page 99 for ISPF Client/Server software installation instructions.

The MVS interface to a TeamConnection command line uses the ISPF SELECT service and WSCMDV (Workstation Command with Variables) to run a TeamConnection command. This service can be invoked from a REXX program. REXX programs running from TSO or the ISPF TSO command line (usually option 6 of the "ISPF Primary Option Window") can use this service to run any TeamConnection command. The following commands are examples of what can be done:

CHECKOUT a part from the TeamConnection data base

CHECKIN a part to the TeamConnection data base

BUILD a part in the TeamConnection data base

The high level data set qualifier **TSOUSER** used in following examples can be replaced by your TSO userid or some other value for your MVS system.

```
The REXX programs in the following examples use the same syntax.

To run a REXX program that is member 'member' of the MVS partitioned
data set 'dataset' from native TSO, at the "Ready" prompt type:
EX 'dataset(member)' 'partname workarea'

To run this program from ISPF, on a command line type:
TSO EX 'dataset(member)' 'partname workarea'

where 'dataset(member)' is the MVS data set containing the REXX program
      'partname'        is the name of the TeamConnection part
      'workarea'        is the name of the TeamConnection work area
```

**Figure 1. Syntax - MVS REXX programs to run TeamConnection commands**

## MVS TO TEAMCONNECTION - CHECKOUT

Running the MVS REXX program in Figure 2 on page 7 results in a part being checked out from a TeamConnection data base and a copy placed in a member of an MVS partitioned data set.

The TeamConnection family must be running and the release, work area, and part must exist. The part version will be checked out from the work area view.

The numbers in parenthesis are not part of the REXX program. They are used as references to statements in the program for the explanation that follows the program.

```
/  REXX PROGRAM TO PERFORM TEAMCONNECTION CHECKOUT      /
TRACE INTERMEDIATES
ADDRESS ISPEXEC

parse arg partname wkareanam                                              (1)

/                                    /
/  BUILD NAME OF PART FOR MVS HOST SYSTEM                /
/                                    /
namelen = LENGTH(partname)                                                (2)
IF namelen > 8 then
   namelen = 8
periodloc = POS('.',partname)
IF periodloc < namelen & ,
     periodloc <>   then
   namelen = periodloc - 1
hostpart = SUBSTR(partname,1,namelen)
hostpart = translate(hostpart)
/                                    /
/  CONSTRUCT TEAMCONNECTION COMMAND                      /
/                                    /
   USER_CMD = 'teamc part -checkout ' partname                            (3)
   USER_CMD = USER_CMD||' -release release1 -workarea ' wkareanam
   USER_CMD = USER_CMD||' -family    mvssclm'
   USER_CMD = USER_CMD||' -relative e:\mytemp'
   WSDIR = 'e:\ispfcs\'                                                   (4)
/                                    /
/  RUN TEAMCONNECTION COMMAND                            /
/  MODE CAN BE MAX, MIN, VIS, OR INVIS                   /
/                                    /
   MODE = 'MAX'
   'SELECT WSCMDV(USER_CMD) WSDIR(WSDIR) 'MODE' MODAL'                                (5)
   SAY 'RC = ' RC
/                                    /
/  TRANSFER WORKSTATION FILE TO MAINFRAME FILE           /
/  HVAR = MAINFRAME FILE TO RECEIVE WORKSTATION FILE     /
/  WVAR = WORKSTATION FILE TO SEND TO MAINFRAME          /
/                                    /
IF RC =   THEN                                                            (6)
  DO
     HVAR = "'TSOUSER.PRIVATE.SOURCE('hostpart')'"
     WVAR = 'e:\mytemp\'partname
     'FILEXFER HOST(HVAR) WS(WVAR) TO(HOST),
                TEXT'
     SAY 'RC = ' RC
  END
EXIT
```

**Figure 2. tcchkout.rex TeamConnection CHECKOUT from MVS**

The parse statement at (1) assigns the first parameter to the name of the part to be checked out and the second parameter to the name of the work area to be used for the TeamConnection CHECKOUT command.

The statements starting at (2) assign a truncation of the part name parameter to the variable "hostpart". This variable will be used for a member name in a partitioned data set. The truncation will start at the ninth character or the first period (whichever occurs first in the parameter). Depending upon your naming conventions, a different technique might be required here to convert from a TeamConnection part name to a partitioned data set member name.

The statements starting at (3) put the TeamConnection CHECKOUT command together by concatenating pieces of the command. The release, family, and relative directory are hard coded. This REXX program can be modified to make these values parameters like "partname" and "wkareanam". The relative directory must exist.

The statement at (4) assigns to the variable WSDIR the workstation directory for running the workstation command. The default is e:\ispfcs (the directory in which the ISPF Client/Server workstation component was installed in Appendix C, “ISPF C/S Workstation Installation Using TCP/IP” on page 99). The TeamConnection commands should run from any directory if you followed the instructions in Appendix B, “TeamConnection Family Setup” on page 81.

The statement at (5) uses the ISPF SELECT service to run WSCMDV (Workstation Command with Variables). The value of USER_CMD will be run as a command from the workstation directory indicated by the value of WSDIR. This is a TeamConnection CHECKOUT command.

The statements at (6) test for a good return code from the workstation command (via the ISPF SELECT service). If the return code is 0, then:

HVAR is assigned the name of the MVS partitioned data set and member to receive a copy of the TeamConnection part that was checked out (allocate this data set as FB with an LRECL of 80). TSOUSER.PRIVATE.SOURCE is a suggested data set name. You can replace the qualifier TSOUSER with your TSO user id. This REXX program can be modified to make the partitioned data set name a parameter like "partname" and "wkareanam".

WVAR is assigned the location of the checked out part on the workstation (the directory is the value of "relative" in the TeamConnection CHECKOUT command in the statements at (3)).

The "FILEXFER" command transfers a copy of the checked out part that is in a workstation file (WVAR) from the workstation to the member of the partitioned data set (HVAR) on MVS. Members that already exist are overwritten.

## MVS TO TEAMCONNECTION - CHECKIN

Running the MVS REXX program in Figure 3 on page 10 results in a copy of a member of an MVS partitioned data set being transferred to a workstation file and the contents of that file being checked in to the TeamConnection data base.

The TeamConnection family must be running and the release, work area, and part must exist.

The numbers in parenthesis are not part of the REXX program. They are used as references to statements in the program for the explanation that follows the program.

```
/  REXX PROGRAM TO PERFORM TEAMCONNECTION CHECKIN        /
TRACE INTERMEDIATES
ADDRESS ISPEXEC
parse arg partname wkareanam                                                    (1)
/                                    /
/  BUILD NAME OF PART FOR MVS HOST SYSTEM                 /
/                                    /
namelen = LENGTH(partname)                                                      (2)
IF namelen > 8 then
   namelen = 8
periodloc = POS('.',partname)
IF periodloc < namelen & ,
     periodloc <>   then
   namelen = periodloc - 1
hostpart = SUBSTR(partname,1,namelen)
hostpart = translate(hostpart)
/                                    /
/  TRANSFER MAINFRAME FILE TO WORKSTATION FILE            /
/  HVAR = MAINFRAME FILE TO SEND TO WORKSTATION           /
/  WVAR = WORKSTATION FILE TO RECEIVE MAINFRAME FILE      /
/                                    /
   HVAR = "'TSOUSER.PRIVATE.SOURCE('hostpart')'"                                (3)
   WVAR = 'e:\mytemp\'partname
   'FILEXFER HOST(HVAR) WS(WVAR) TO(WS),
                TEXT'
   SAY 'RC = ' RC
/                                    /
/  IF TRANSFER WAS SUCCESSFUL, CHECK IN TO                /
/  TEAMCONNECTION                                         /
/                                    /
IF RC =   THEN
  DO
     /                                  /
      /  CONSTRUCT TEAMCONNECTION COMMAND                /
     /                                  /
      USER_CMD = 'teamc part -checkin ' partname                                (4)
      USER_CMD = USER_CMD||' -release release1 -workarea ' wkareanam
      USER_CMD = USER_CMD||' -family mvssclm'
      USER_CMD = USER_CMD||' -relative e:\mytemp'
      WSDIR = 'e:\ispfcs\'                                                      (5)
     /                                  /
      /  RUN TEAMCONNECTION COMMAND                      /
      /  MODE MAY BE MAX, MIN, VIS, OR INVIS             /
     /                                  /
      MODE = 'MAX'
      'SELECT WSCMDV(USER_CMD) WSDIR(WSDIR) 'MODE' MODAL'                       (6)
      SAY 'RC = ' RC
  END
EXIT
```

**Figure 3. tcchkin.rex TeamConnection CHECKIN from MVS**

The parse statement at (1) assigns the first parameter to the name of the part to be checked in and the second parameter to the name of the work area to be used for the TeamConnection CHECKIN command.

The statements starting at (2) assign a truncation of the part name parameter to the variable "hostpart". This variable will be used for a member name in a partitioned data set. The truncation will start at the ninth character or the first period (whichever occurs first in the parameter). Depending upon your naming conventions, a different technique might be required here to convert from a TeamConnection part name to a partitioned data set member name.

The statements at (3) assign the name of the MVS partitioned data set and member to variable HVAR and the workstation location for the transferred member to variable WVAR. The "FILEXFER" command transfers a copy of the MVS partitioned data set member (HVAR) from MVS to the workstation (WVAR). This REXX program can be modified to make the partitioned data set name a parameter like "partname" and "wkareanam".

The statements starting at (4) put the TeamConnection CHECKIN command together by concatenating pieces of the command. The release, family, and relative directory are hard coded. This REXX program can be modified to make these values parameters like "partname" and "wkareanam". The relative directory must exist.

The statement at (5) assigns to the variable WSDIR the workstation directory for running the workstation command. The default is e:\ispfcs (the directory in which the ISPF Client/Server workstation component was installed in Appendix C, “ISPF C/S Workstation Installation Using TCP/IP” on page 99). The TeamConnection commands should run from any directory if you followed the instructions in Appendix B, “TeamConnection Family Setup” on page 81.

The statement at (6) uses the ISPF SELECT service to run WSCMDV (Workstation Command with Variables). The value of USER_CMD will be run as a command from the workstation directory indicated by the value of WSDIR. This is a TeamConnection CHECKIN command.

## MVS TO TEAMCONNECTION - BUILD

Running the MVS REXX program in Figure 4 on page 12 results in a TeamConnection build of a part in the TeamConnection data base.

The TeamConnection family must be running and the release, work area, and part must exist.

The numbers in parenthesis are not part of the REXX program. They are used as references to statements in the program for the explanation that follows the program.

```
/ REXX PROGRAM TO PERFORM TEAMCONNECTION BUILD           /
TRACE INTERMEDIATES
ADDRESS ISPEXEC

parse arg partname wkareanam                                          (1)

/                                   /
/ CONSTRUCT TEAMCONNECTION COMMAND                      /
/                                                   /
/                                   /
   USER_CMD = 'teamc part -build ' partname ' -type TCPart'           (2)
   USER_CMD = USER_CMD||' -release release1 -workarea' wkareanam
   USER_CMD = USER_CMD||' -family mvssclm'
   USER_CMD = USER_CMD||' -pool pool1 -normal'
   WSDIR = 'e:\ispfcs\'                                               (3)

/                                   /
/ RUN TEAMCONNECTION COMMAND                            /
/ MODE MAY BE MAX, MIN, VIS, OR INVIS                 /
/                                                   /
/                                   /
   MODE = 'MAX'
   'SELECT WSCMDV(USER_CMD) WSDIR(WSDIR) 'MODE' MODAL'                (4)
   SAY 'RC = ' RC
EXIT
```

**Figure 4. tcbuild.rex TeamConnection BUILD from MVS**

The parse statement at (1) assigns the first parameter to the name of the part to be built and the second parameter to the name of the work area to be used for the TeamConnection BUILD command.

The statements starting at (2) put the TeamConnection BUILD command together by concatenating pieces of the command. The release and family are hard coded. This REXX program can be modified to make these values parameters like "partname" and "wkareanam".

The statement at (3) assigns to the variable WSDIR the workstation directory for running the workstation command. The default is e:\ispfcs (the directory in which the ISPF Client/Server workstation component was installed in Appendix C, “ISPF C/S Workstation Installation Using TCP/IP” on page 99). The TeamConnection commands should run from any directory if you followed the instructions in Appendix B, “TeamConnection Family Setup” on page 81.

The statement at (4) uses the ISPF SELECT service to run WSCMDV (Workstation Command with Variables). The value of USER_CMD will be run as a command from the workstation directory indicated by the value of WSDIR. This is a TeamConnection BUILD command which

places a build event on the TeamConnection job queue. The TeamConnection build agent and build processor must be started in order for the build event to be removed from the job queue by the build agent and sent on to the build processor.

## PUTTING IT ALL TOGETHER

The following example uses MVS REXX programs and the TeamConnection parts from Appendix A, "TeamConnection Parts and Scripts" on page 69. This example shows how from MVS you can:

1. CHECKOUT a TeamConnection part (Figure 2 on page 7)
2. change the part in MVS
3. CHECKIN the part to TeamConnection (Figure 3 on page 10)
4. BUILD a part in TeamConnection (Figure 4 on page 12)

### Setup Requirements

The TeamConnection family mvssclm and parts must be installed on the workstation (see Appendix B, "TeamConnection Family Setup" on page 81).

The ISPF Client/Server must be installed and running on the workstation (see Appendix C, "ISPF C/S Workstation Installation Using TCP/IP" on page 99).

### Step 1 - Workstation preliminary activities

| Action | Description |
|---|---|
| Start the ObjectStore server. | |
| Start the family mvssclm. | |
| From an OS/2 window set the TeamConnection family information:<br>**TC_FAMILY=mvssclm**<br>**TC_USER=SuprUser**<br>**TC_BECOME=SuprUser**<br>**TC_DBPATH=e:\family\mvssclm**<br>Start the build agent from this OS/2 window by issuing this command:<br>**teamagnt -f mvssclm -e os2 -p pool1 -s bldsock**<br>Do not close the window. | The build agent starts in the OS/2 window.<br>You must also have PATH set to e:\family\mvssclm if that is where teamagnt.exe and teamproc.exe are located. Otherwise you could get a SYS1041 error from OS/2. |

| Action | Description |
| --- | --- |
| From an OS/2 window start the build processor by issuing this command:<br>**teamproc -s bldsock**<br>Do not close the window. | The build processor starts in the OS/2 window. |
| Start the ISPF C/S on the workstation by entering **ispfcs** from the "e:\ispfcs" directory. | The "ISPF C/S" window will appear on the workstation.<br>Do not close the "ISPF C/S" window. |

## Step 2 - MVS preliminary activities

| Action | Description |
| --- | --- |
| LOGON to TSO. | |
| Bring up ISPF. | ISPF may come up automatically or may need to be brought up by entering "ISPF" of "PDF". |
| Select **Settings** from the "ISPF Primary Options Menu". This is usually option 0. | The "ISPF Settings" panel appears. |

| Action | Description |
| --- | --- |
| Change **Options**, **Print Graphics**, **General**, and **Terminal Characteristics** as needed on the "ISPF Settings" panel (these values retain their previous settings and change infrequently).<br><br>Suggested "Options"<br>/ Command line at bottom<br>/ Panel display CUA mode<br>Long message in pop-up<br>/ Tab to action bar choices<br>Tab to point-and-shoot fields<br>/ Restore TEST/TRACE options<br>Session Manager mode<br>/ Jump from leader dots<br>Edit PRINTDS Command<br>/ Always show split line<br><br>Suggested "Print Graphics"<br>Family printer type 2<br>Device name . . . .<br>Aspect ratio . . . 0<br><br>Suggested "General"<br>Input field pad . . N<br>Command delimiter . ;<br><br>Suggested "Terminal Characteristics"<br>Screen format 1 (Data)<br>Terminal Type 3 (3278) | The suggested "Options" are not required. |
| Select **Workstation** from the "ISPF Settings" panel.<br><br>Tab to the Workstation heading and press the **Enter** key. | The "Workstation" pulldown appears. |
| Select **Workstation connection...** from the "Workstation" pulldown. | The "Initiate Workstation Connection" panel appears. |

| Action | Description |
|---|---|
| Change **Workstation connection, GUI Network Protocol, GUI Window Frame, Default window background color, GUI Title,** and **TCP/IP Address** as needed on the "Initiate Workstation Connection" panel (TCP/IP is assumed for this example).<br><br>Suggested settings:<br>**Workstation connection** 2 (Without GUI display)<br>**GUI Network Protocol** 1 (TCP/IP)<br>**GUI Window Frame** 1 (Standard (STD))<br>**Default window background color** 1 Dialog (DLG)<br>**GUI Title** TSOUSER<br>**TCP/IP Address** 9.37.195.43 | The "TCP/IP Address" is the address of the workstation machine. |
| Press the **Enter** key to initiate a workstation session. | A "Connection" information window appears. |
| Select **Yes** on the "Connection" information window to accept a connection to the workstation. | |
| Go back to the "ISPF Primary Options Menu" by pressing the **F3** function key. | |
| Allocate the partitioned data set **'TSOUSER.PRIVATE.SOURCE'** (or some other name) as FB with LRECL=80 or select an existing partitioned data set to use with the REXX programs for receiving files from the workstation.<br><br>You can use ISPF Utilities (usually option 3) and then select the data set option (usually option 2). Suggested space unit is 3200, primary quantity is 10, secondary quantity is 50, and directory blocks is 10. | Parts checked out from, and checked into, TeamConnection require an MVS partitioned data set to hold copies of the parts.<br><br>You can use an existing data set or allocate a new one. The REXX programs will need to be updated by replacing 'TSOUSER.PRIVATE.SOURCE' with the correct data set name. |
| Allocate the partitioned data set **'TSOUSER.TEAMC.REXX'** (or some other name) as FB with LRECL=80. You can use an existing partitioned data set.<br><br>Create the following members:<br>**TCCHKOUT** with the contents of Figure 2 on page 7<br>**TCCHKIN** with the contents of Figure 3 on page 10<br>**TCBUILD** with the contents of Figure 4 on page 12 | See Appendix A, “TeamConnection Parts and Scripts” on page 69 for information about soft copies of the REXX programs. |

## Step 3 - TeamConnection CHECKOUT issued from MVS

| Action | Description |
|---|---|
| Make any needed changes to the REXX program in Figure 2 on page 7 (part of the materials referenced in Appendix A, “TeamConnection Parts and Scripts” on page 69). | The MVS data set 'TSOUSER.PRIVATE.SOURCE' will need to be changed to one of your MVS data sets.<br>The other values should not change if you did not change them when you set up the TeamConnection family. |
| Run the REXX program in Figure 2 on page 7 from a TSO or ISPF TSO command line to CHECKOUT TeamConnection part testpart.c (Figure 12 on page 71) and place it in an MVS data set:<br>**ex 'TSOUSER.TEAMC.REXX(TCCHKOUT)' 'testpart.c mvstcwa'** | The member 'TESTPART' in 'TSOUSER.PRIVATE.SOURCE' should contain the same data as testpart.c. |

## Step 4 - Change the part in MVS

| Action | Description |
|---|---|
| Change 'TSOUSER.PRIVATE.SOURCE(TESTPART)' using ISPF EDIT. Add the line:<br>**printf("I was changed on MVS!\n");**<br>after the 'printf("I am testpart.c\n");' line so that you can see the change by extracting and running testpart.exe after the TeamConnection BUILD. | |

## Step 5 - TeamConnection CHECKIN issued from MVS

| Action | Description |
|---|---|
| Make any needed changes to the REXX program in Figure 3 on page 10 (part of the materials referenced in Appendix A, “TeamConnection Parts and Scripts” on page 69). | The MVS data set 'TSOUSER.PRIVATE.SOURCE' will need to be changed to one of your MVS data sets.<br>The other values should not change if you did not change them when you set up the TeamConnection family. |

| Action | Description |
| --- | --- |
| Run the REXX program in Figure 3 on page 10 from a TSO or ISPF TSO command line to transfer 'TSOUSER.PRIVATE.SOURCE(TESTPART)' to a workstation file and CHECKIN that file as TeamConnection part testpart.c:<br>**ex 'TSOUSER.TEAMC.REXX(TCCHKIN)' 'testpart.c mvstcwa'** | The TeamConnection part testpart.c should contain the same data as the member 'TESTPART' in 'TSOUSER.PRIVATE.SOURCE'. This data should contain the changes made from the ISPF EDIT. |

## Step 6 - TeamConnection BUILD issued from MVS

| Action | Description |
|---|---|
| Make any needed changes to the REXX program in Figure 4 on page 12 (part of the materials referenced in Appendix A, “TeamConnection Parts and Scripts” on page 69). | There should be no changes if you did not change the family, release, work area, or pool name when you set up the TeamConnection family. |
| Run the REXX program to BUILD the application in TeamConnection:<br>**ex 'TSOUSER.TEAMC.REXX(TCBUILD)' 'testpart.exe mvstcwa'** | A window will appear on the workstation showing the build progress messages. |

## Step 7 - Test the OS/2 Application

Extract the executable part **testpart.exe** and run it for the test.

| Action | Description |
|---|---|
| From the "TeamConnection - Tasks" window select **Actions**, then **Parts**, and then **Extract**. | The "Extract Parts" window appears. |
| Set the following fields:<br>**Path names** testpart.exe<br>**Release** release1<br>**Work area** mvstcwa<br>**Destination directory** e:\mytemp<br>Press the **OK** push button. | |
| Go to an OS/2 window.<br>Run the command **e:\mytemp\testpart**.<br>Minimize or close the OS/2 window. | The messages from testpart.exe should appear:<br>I am testpart.c<br>I was changed on MVS!<br>I was built by the Team Connection integrated Build Tool!!! |

## Hints and Tips for Debugging

if you get RC=20 from the ISPF SELECT statement in TCCHKOUT, TCCHKIN, or TCBUILD

- check that the TCP/IP information you specified in the workstation connection information is correct

if you get other return codes (eg 301, 101) from the ISPF SELECT statement in TCCHKOUT, TCCHKIN, or TCBUILD

– look at e:\family\mvssclm\audit.log to determine cause of the problem

# SCLM TO TEAMCONNECTION

The connection between SCLM and the TeamConnection client or family server machine is through the ISPF 4.2 Client/Server function. The ISPF Client/Server software must be installed on the workstation in order for SCLM to communicate with the workstation. See Appendix C, “ISPF C/S Workstation Installation Using TCP/IP” on page 99 for ISPF Client/Server software installation instructions.

## SCLM WORKSTATION BUILD - HOW IT WORKS

The SCLM workstation build function uses the ISPF SELECT function to run the REXX program FLMLTWST. This REXX program uses techniques similar to those described in “MVS to TeamConnection” on page 5 to transfer files to the workstation, run a command, and transfer files back to the mainframe. FLMLTWST uses the USERINFO (see Figure 20 on page 109) and ACTINFO (see Figure 21 on page 110) files to put together the workstation commands and response files as needed. The files to be sent to the workstation and the files to be sent back to the mainframe are determined from information in the SCLM build map.

The SCLM BUILD function creates a new build map that is used by SCLM in much the same fashion as TeamConnection uses a job queue of build events. SCLM creates the build map by looking at the language definitions of the members being built. SCLM allocates the appropriate data sets and passes control to the tools (called translators) identified by the language definition COMPILE keywords for members that have changed, or have been affected by changes, since the last build. These translators are typically precompilers, compilers, linkage editors, and other mainframe tools. For a workstation build the mainframe tool is the ISPF SELECT function.

An SCLM workstation build function that uses FLMLTWST relies on 3 files:

ACTINFO - ACTION parameter information (see Figure 21 on page 110 for references to the numbered statements)

- (1) SCLM file type mapping to workstation file name extension
- (2) ASCII/EBCDIC translation for a particular SCLM file type
- (3) subdirectory for transferring files to or from the workstation (each command can specify a different subdirectory)
- (4) one workstation command (such as a TeamConnection CHECKOUT, CHECKIN, or BUILD command) for each ACTION keyword parameter value
- response file contents (not used in this example)

USERINFO - information specific to the workstation

- DATA_DIR directory for transferring files to or from the workstation (this will be concatenated with the subdirectory value from the ACTINFO file)
- WSDIR directory for running the workstation command (must be the same as DATA_DIR for TeamConnection commands)
- RESPONSE_FILE response file name for commands that use a response file (the response file contents from the ACTINFO file will be stored in this file for use by the workstation command)
- mode WSCMDV MODE default
- del_cmd delete command

language definition - one for each workstation command, see Figure 23 on page 115, Figure 24 on page 119, and Figure 25 on page 122. (The SCLM project definition load module is used as a table of information by SCLM at run time. Language definitions are part of the project definition load module.)

- FLMLTWST parameters
- data set name of USERINFO file
- data set name of ACTINFO file

## SCLM WORKSTATION BUILD - TEAMCONNECTION COMMANDS

The SCLM build function can be used to run any TeamConnection command. This section describes how the SCLM build function can be used to do this.

The SCLM workstation build function is designed to transfer MVS input members to workstation files, run workstation commands that produce output files, and transfer these workstation output files back to MVS members. SCLM does not "know" if the input files are really needed by the workstation command or if the output files are really produced by the workstation command. There is no restriction upon these workstation commands to only run compilers, linkage editors, and other traditional build tools.

By placing TeamConnection commands in the ACTINFO file (the MVS file that maps FLMLTWST ACTION parameter values to workstation commands) you can use the SCLM BUILD function to run any TeamConnection command on a workstation. The language definitions CHECKOUT (Figure 23 on page 115), CHECKIN (Figure 24 on page 119), and TCBUILD (Figure 25 on page 122) all use the ISPF SELECT function to run FLMLTWST for an SCLM BUILD. These language definitions, the ACTINFO file, and the USERINFO file are used by FLMLTWST to run TeamConnection CHECKOUT, CHECKIN, and BUILD commands on a workstation.

You need to create a member in an SCLM controlled library that has the appropriate language definition for the desired TeamConnection command. An SCLM BUILD of this member, or of

an architecture definition member that references this member, will result in running the TeamConnection command.

FLMLTWST was written to transfer files to and from the workstation based solely upon the contents of the build map. The build map identifies inputs (source members and related dependencies) and outputs. This means that the input members (both source and dependencies) identified in the build map will be transferred to the workstation even if they are not used as inputs to the TeamConnection command. Similarly output members identified in the build map will be transferred back to MVS even if they were not created by running the TeamConnection command. In a sense FLMLTWST can be used as a "Trojan horse" to get members into, run commands on, and get files out of a workstation.

You can modify FLMLTWST so that it only transfers files to the workstation when they are needed (See “SCLM Architecture Definition Parameters for TeamConnection Commands” on page 61 and Appendix F, “SCLM FLMLTWST Extensions for Parameters” on page 135 for how this can be done.) You can also run a REXX program (rather than a TeamConnection command) on the workstation which uses the files transferred to the workstation as input and then run TeamConnection commands based upon information in these files. The possibilities go beyond running a simple TeamConnection command.

The examples shown in this section cover the basics of running simple TeamConnection commands from an SCLM BUILD. The source members are sent to the workstation for each SCLM BUILD but are not always needed by the TeamConnection command. The examples run SCLM BUILD with architecture definition members (the SCLM counter-part of TeamConnection build trees) rather than source members because the author has discovered from years of SCLM experience that this is a good habit. Architecture definitions are also useful for parameters so that the TeamConnection commands, action flags, option lists, attribute flags, and arguments do not have to be hard coded (see “SCLM Architecture Definition Parameters for TeamConnection Commands” on page 61).

## SCLM TO TEAMCONNECTION - CHECKOUT

An SCLM BUILD of a member with a language of CHECKOUT (Figure 23 on page 115) results in a part being checked out from a TeamConnection data base and a copy placed in a member of an SCLM controlled library.

The TeamConnection family must be running and the release, work area, and part must exist. The part version will be checked out from the work area view.

See Figure 23 on page 115 for a description of how SCLM uses the CHECKOUT language definition.

## SCLM TO TEAMCONNECTION - CHECKIN

An SCLM BUILD of a member with a language of CHECKIN (Figure 24 on page 119) results in the contents of a member of an SCLM controlled library being transferred to a workstation file which is then checked in to a TeamConnection data base.

The TeamConnection family must be running and the release, work area, and part must exist.

See Figure 24 on page 119 for a description of how SCLM uses the CHECKIN language definition.

## SCLM TO TEAMCONNECTION - BUILD

An SCLM BUILD of a member with a language of TCBUILD (Figure 25 on page 122) results in a TeamConnection build of a TeamConnection part.

The TeamConnection family must be running and the release, work area, and part must exist. The part version will be built from the work area view.

See Figure 25 on page 122 for a description of how SCLM uses the TCBUILD language definition.

## PUTTING IT ALL TOGETHER

The following example uses SCLM BUILD and the TeamConnection parts from Appendix A, "TeamConnection Parts and Scripts" on page 69. This example shows how from SCLM you can:

1. CHECKOUT a TeamConnection part
2. change the part in SCLM
3. CHECKIN the part to TeamConnection
4. BUILD a part in TeamConnection

### Setup Requirements

The TeamConnection family mvssclm and parts must be installed on the workstation (see Appendix B, "TeamConnection Family Setup" on page 81).

The ISPF Client/Server must be installed on the workstation (see Appendix C, "ISPF C/S Workstation Installation Using TCP/IP" on page 99).

The SCLM project definition SCLMTC must be installed on the MVS system (see Appendix D, "SCLM Project Definition Setup" on page 103).

## Notes

In this example, the name of the TeamConnection part to be checked out, checked in, and built as well as the TeamConnection part commands CHECKOUT, CHECKIN, and BUILD are hard-coded in the ACTINFO file (Figure 21 on page 110). See “SCLM Architecture Definition Parameters for TeamConnection Commands” on page 61 for how to use parameters.

The SCLM type and member name to receive a copy of the checked out TeamConnection part is specified in an architecture definition output statement created in “Step 3 - TeamConnection CHECKOUT issued from SCLM BUILD” on page 26.

## Step 1 - Workstation preliminary activities

| Action | Description |
|---|---|
| Be sure these environment variables are set in config.sys:<br>**TC_FAMILY** mvssclm<br>**TC_USER** SuprUser<br>**TC_BECOME** SuprUser<br>**TC_DBPATH** e:\family\mvssclm | |
| Start the ObjectStore server. | |
| Start the family mvssclm. | |
| From an OS/2 window issue the command<br>**teamagnt -f mvssclm -e os2 -p pool1 -s bldsock** | The build agent starts in the OS/2 window.<br>Do not close the window. |
| From an OS/2 window issue the command<br>**teamproc -s bldsock** | The build processor starts in the OS/2 window.<br>Do not close the window. |
| Start the ISPF Client/Server on the workstation by entering **ispfcs** from the "e:\ispfcs" directory. | The "ISPF Client/Server" window appears on the workstation.<br>Do not close the window. |

## Step 2 - MVS preliminary activities

| Action | Description |
|---|---|
| LOGON to TSO. | |

| Action | Description |
|---|---|
| If you do not have FLMLTWST in a data set concatenated to SYSEXEC then run a CLIST that will concatenate a data set containing FLMLTWST with the libraries in SYSEXEC. Your TSO session can be set up to perform this concatenation automatically. | See Figure 27 on page 126 for an example of a CLIST that will perform the concatenation using the system library 'SYS1.ISP.SISPSAMP' which contains FLMLTWST. Some systems use a different library name. You can determine the library name by checking to see which "ISP" data set is allocated to ddname SAMPLIB.<br>This must be done before bringing up ISPF. |
| Bring up ISPF. | |
| Connect to the workstation. | See “Step 2 - MVS preliminary activities” on page 14 for detailed instructions. |

## Step 3 - TeamConnection CHECKOUT issued from SCLM BUILD

In this step you run an SCLM BUILD of an architecture definition member which references an input member assigned the language CHECKOUT and an output member for receiving a copy of a TeamConnection part. The CHECKOUT language runs FLMLTWST which issues a TeamConnection CHECKOUT command and stores a copy of the checked out part in a member of an SCLM controlled library.

There are three parts to this step:

create member 'TSOUSER.DEVELOP.SOURCE(CHECKOUT)' using SCLM EDIT and assign the language CHECKOUT (this language runs the TeamConnection CHECKOUT command)

create member 'TSOUSER.DEVELOP.ARCHDEF(CHECKOUT)' using SCLM EDIT and assign the language ARCHDEF (this member identifies the input member 'TSOUSER.DEVELOP.SOURCE(CHECKOUT)' and the output member 'TSOUSER.DEVELOP.SRC(TESTPART)' for the SCLM BUILD)

run SCLM BUILD of member 'TSOUSER.DEVELOP.ARCHDEF(CHECKOUT)' (the CHECKOUT language of the input member 'TSOUSER.DEVELOP.SOURCE(CHECKOUT)' runs the TeamConnection CHECKOUT command for TeamConnection part testpart.c and stores a copy of testpart.c in 'TSOUSER.DEVELOP.SRC(TESTPART)')

NOTE: If the member 'TSOUSER.DEVELOP.SOURCE(CHECKOUT)' exists from a previous demonstration you must either delete the member using the SCLM library utility and follow the

instructions below or use SCLM EDIT and SAVE the member so that it will be updated for the SCLM BUILD (remember to enter "SAVE" on the command line in the SCLM EDIT session).

Create 'TSOUSER.DEVELOP.SOURCE(CHECKOUT)': This member does not take part in the TeamConnection CHECKOUT but is transferred to the workstation as part of the SCLM BUILD due to the way FLMLTWST is written. For this reason it will contain one line of dummy information. (See “SCLM Architecture Definition Parameters for TeamConnection Commands” on page 61 for how to avoid the file transfer.)

| Action | Description |
|---|---|
| Select **SCLM** from the "ISPF Primary Option Menu".<br>This is usually option 10. | The "SCLM Main Menu" appears. |
| On the **SCLM Main Menu** enter:<br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 2<br>Edit is usually option 2. | The "SCLM Edit - Entry Panel" appears.<br>The 'TSOUSER.DEVELOP.xxx' data sets were allocated in Appendix D, “SCLM Project Definition Setup” on page 103. |
| On the **SCLM Edit - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** SOURCE<br>**Member** CHECKOUT<br>Press the **Enter** key. | An SCLM EDIT session appears.<br>The 'TSOUSER.DEVELOP.SOURCE' data set was allocated in Appendix D, “SCLM Project Definition Setup” on page 103. |
| Enter the following line of dummy information:<br>**THIS IS DUMMY INFORMATION.** | |
| Press the **F3** function key to save the member. | The "SCLM Edit Profile" box appears. |
| Enter **CHECKOUT** for the "Language".<br>Press the **Enter** key. | The language definition CHECKOUT (Figure 23 on page 115) is assigned to this member.<br>The "SCLM Edit - Entry Panel" appears. |

Create 'TSOUSER.DEVELOP.ARCHDEF(CHECKOUT)': An SCLM architecture definition member needs to be created for the SCLM BUILD. You can create architecture definition member 'TSOUSER.DEVELOP.ARCHDEF(CHECKOUT)' by using SCLM EDIT and assigning the language ARCHDEF. This member identifies the source member to be built ('TSOUSER.DEVELOP.SOURCE(CHECKOUT)') and the output member ('TSOUSER.DEVELOP.SRC(TESTPART)') to receive a copy of the TeamConnection part.

NOTE: If the output member exists from a previous demonstration you can delete the member using the SCLM library utility. This must be done if the output member exists but was changed to an editable member during a previous demonstration because SCLM will not replace an editable member with an output from a BUILD.

If the architecture definition member exists from a previous demonstration, it does not need to be recreated or edited and saved. It can be used "as is" and you can go directly to the SCLM BUILD of this member in this step.

| Action | Description |
|---|---|
| On the **SCLM Edit - Entry Panel** enter:<br><br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** ARCHDEF<br>**Member** CHECKOUT<br><br>Press the **Enter** key. | An SCLM EDIT session appears. |
| Enter the following:<br><br>**SINC CHECKOUT SOURCE**<br>**OUT1 TESTPART SRC** | The SINC architecture definition statement (SINC means Source INClude) identifies the input member (CHECKOUT) to be built and the low level data set qualifier (SOURCE). Member CHECKOUT has the language CHECKOUT.<br><br>The OUT1 architecture definition statement identifies the output member (TESTPART) to be created and the low level data set qualifier (SRC).<br><br>The language definition CHECKOUT (Figure 23 on page 115) allocates the SCLM output data set 'TSOUSER.DEVELOP.SRC' that contains member TESTPART. The FLMALLOC macro with KEYREF=OUT1 in this language definition performs the allocation.<br><br>Note: The ACTINFO file maps the SCLM type "SRC" to the TeamConnection extension "C". |
| Press the **F3** function key to save the member. | The "SCLM Edit Profile" box appears. |

| Action | Description |
|---|---|
| Enter **ARCHDEF** for the "Language".<br>Press the **Enter** key. | The language definition ARCHDEF is assigned to this member. The project definition (Figure 26 on page 125) uses the standard architecture definition language FLM@ARCD.<br>The "SCLM Edit - Entry Panel" appears. |
| Press the **F3** function key. | The "SCLM Main Menu" appears. |

Build 'TSOUSER.DEVELOP.ARCHDEF(CHECKOUT)': This section describes how to build member 'TSOUSER.DEVELOP.ARCHDEF(CHECKOUT)'. During the SCLM BUILD, the TeamConnection CHECKOUT command runs and the part testpart.c is checked out as per the TeamConnection CHECKOUT command in the ACTINFO file (Figure 21 on page 110). A copy of this part will be transferred to MVS and stored in 'TSOUSER.DEVELOP.SRC(TESTPART)'.

| Action | Description |
|---|---|
| On the **SCLM Main Menu** enter:<br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 4<br>Build is usually option 4. | The "SCLM Build - Entry Panel" appears. |
| On the **SCLM Build - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** ARCHDEF<br>**Member** CHECKOUT<br>**Workstation Build** /<br>**Mode** 1<br>**Scope** 2<br>**Process** 1<br>**Messages** 1 3<br>**Report** 1 3<br>**Listings** 1 3<br>Press the **Enter** key. | A workstation window appears while the CHECKOUT command runs.<br>When the build ends you will find member TESTPART in 'TSOUSER.DEVELOP.SRC'.<br>TeamConnection part testpart.c is checked out and locked on the workstation. |

## Step 4 - Change the part in SCLM

The member TESTPART in 'TSOUSER.DEVELOP.SRC' is an output from an SCLM BUILD and as such cannot be edited without changing it to an editable member. This section shows the steps to:

1. change the member 'TSOUSER.DEVELOP.SRC(TESTPART)' to an editable member
   a. copy 'TSOUSER.DEVELOP.SRC(TESTPART)' to another data set
   b. use SCLM library utilities to delete 'TSOUSER.DEVELOP.SRC(TESTPART)'
   c. use SCLM to create 'TSOUSER.DEVELOP.SRC(TESTPART)' with the language CHECKIN and the contents of TESTPART as stored in the other data set
2. change the member 'TSOUSER.DEVELOP.SRC(TESTPART)' using SCLM EDIT
3. assign the CHECKIN language

In the next step this member will be an input for an SCLM BUILD that will transfer the member to the workstation and run the TeamConnection CHECKIN command.

| Action | Description |
|---|---|
| Copy **'TSOUSER.DEVELOP.SRC(TESTPART)'** to a non-SCLM controlled data set for later use. | The member contents needs to be saved while the SCLM controlled member is deleted and then created. |
| Select **SCLM** from the "ISPF Primary Option Menu".<br>This is usually option 10. | The "SCLM Main Menu" appears. |
| On the **SCLM Main Menu** enter:<br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 3<br>Utilities is usually option 3. | The "SCLM Utilities Menu" appears.<br>The "SCLM Utilities" is used to delete member TESTPART. |
| Select **Library** from the "SCLM Utilities Menu".<br>This is usually option 1. | The "SCLM Library Utility - Entry Panel" appears. |
| On the **SCLM Library Utility - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** SRC<br>**Member** TESTPART<br>**Option ===>** D<br>Press the **Enter** key. | An SCLM "Confirm Delete" window appears. |

| Action | Description |
| --- | --- |
| Press the **Enter** key. | The member 'TSOUSER.DEVELOP.SRC(TESTPART)' is deleted.<br>The "SCLM Library Utility - Entry Panel" appears. |
| Press the **F3** function key. | The "SCLM Utilities Menu" appears. |
| Press the **F3** function key. | The "SCLM Main Menu" appears. |
| On the **SCLM Main Menu** enter:<br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 2<br>Edit is usually option 2. | The "SCLM Edit - Entry Panel" appears. |
| On the **SCLM Edit - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** SRC<br>**Member** TESTPART<br>Press the **Enter** key. | An SCLM EDIT session appears.<br>This will create the member TESTPART. |
| Copy in the contents of member **TESTPART** from the data set used to save it prior to deletion.<br>Add the following line:<br>**printf("I was also changed on SCLM!\n");**<br>before the line: 'printf("I was built by the TeamConnection\n");'. | To copy the member:<br>enter "COPY" on the EDIT command line<br>press the "Enter" key<br>enter the data set information on the panel that appears |
| Press the **F3** function key to save the member. | The "SCLM Edit Profile" box appears. |
| Enter **CHECKIN** for the "Language" in the "SCLM Edit Profile" box.<br>Press the **Enter** key. | The language definition CHECKIN (Figure 24 on page 119) is assigned to this member.<br>The "SCLM Edit - Entry Panel" appears. |

## Step 5 - TeamConnection CHECKIN issued from SCLM BUILD

Member 'TSOUSER.DEVELOP.SRC(TESTPART)' was assigned the language CHECKIN in the previous step.

This section describes how to create the architecture member 'TSOUSER.DEVELOP.ARCHDEF(CHECKIN)' using SCLM EDIT. This member is assigned the language ARCHDEF and identifies the member to be built ('TSOUSER.DEVELOP.SRC(TESTPART)').

An SCLM BUILD of 'TSOUSER.DEVELOP.ARCHDEF(CHECKIN)' results in a transfer of the contents of 'TSOUSER.DEVELOP.SRC(TESTPART)' to the workstation followed by the TeamConnection part CHECKIN command.

The part testpart.c as changed in SCLM is checked in to TeamConnection.

NOTE: If the architecture definition member exists from a previous demonstration, it does not need to be recreated or edited and saved. It can be used "as is" and you can go directly to the SCLM BUILD of this member in this step.

| Action | Description |
|---|---|
| On the **SCLM Edit - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** ARCHDEF<br>**Member** CHECKIN<br>Press the **Enter** key. | An SCLM EDIT session appears. |
| Enter the following:<br>**SINC TESTPART SRC** | Note: There is no output part created for this BUILD.<br>The ACTINFO file maps the SCLM type "SRC" to the TeamConnection extension "C". |
| Press the **F3** function key to save the member. | The "SCLM Edit Profile" box appears. |
| In the "SCLM Edit Profile" box enter **ARCHDEF** for the "Language".<br>Press the **Enter** key. | The language definition ARCHDEF is assigned to this member. The project definition (Figure 26 on page 125) uses the standard architecture definition language FLM@ARCD.<br>The "SCLM Edit - Entry Panel" appears. |

| Action | Description |
|---|---|
| Press the **F3** function key. | The "SCLM Main Menu" will appear. |
| On the **SCLM Main Menu** enter:<br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 4<br>Build is usually option 4. | The "SCLM Build - Entry Panel" appears. |
| On the **SCLM Build - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** ARCHDEF<br>**Member** CHECKIN<br>**Workstation Build** /<br>**Mode** 1<br>**Scope** 2<br>**Process** 1<br>**Messages** 1 3<br>**Report** 1 3<br>**Listings** 1 3<br>Press the **Enter** key. | A workstation window will appear while the TeamConnection CHECKIN command runs.<br>When the build ends you will find TeamConnection part testpart.c will be checked in and unlocked on the workstation.<br>You could select 'Action -> Parts -> View -> View contents' on TeamConnection for part testpart.c to see the changes that you made. |
| Return to the **SCLM Main Menu**. | |

## Step 6 - TeamConnection BUILD issued from SCLM BUILD

In this step you run an SCLM BUILD of an architecture definition member which references an input member assigned the language TCBUILD. The TCBUILD language runs FLMLTWST which issues a TeamConnection BUILD command for part testpart.exe.

There are three parts to this step:

create member 'TSOUSER.DEVELOP.SOURCE(TCBUILD)' using SCLM EDIT and assign the language TCBUILD (this language runs the TeamConnection BUILD command)

create member 'TSOUSER.DEVELOP.ARCHDEF(TCBUILD)' using SCLM EDIT and assign the language ARCHDEF (this member identifies the input member 'TSOUSER.DEVELOP.SOURCE(TCBUILD)' for the SCLM BUILD)

run SCLM BUILD of member 'TSOUSER.DEVELOP.ARCHDEF(TCBUILD)' (the TCBUILD language of the input member 'TSOUSER.DEVELOP.SOURCE(TCBUILD)' runs a TeamConnection BUILD command for TeamConnection part testpart.exe)

Even though the member 'TSOUSER.DEVELOP.SOURCE(TCBUILD)' does not take part in the TeamConnection BUILD it is transferred to the workstation due to the way FLMLTWST is written. For this reason it contains one line of dummy information because there are no "empty" members in SCLM controlled libraries. (See “SCLM Architecture Definition Parameters for TeamConnection Commands” on page 61 for how to avoid a transfer of a file.)

NOTE: If the member exists from a previous demonstration, you must either delete the member using the SCLM library utility and follow the instructions below or use SCLM EDIT and SAVE the member so that it will be updated for the SCLM BUILD (remember to enter "SAVE" on the command line in the SCLM EDIT session).

The TeamConnection CHECKIN from the previous step results in TeamConnection part testpart.c needing to be built. If testpart.c had not changed since the last successful TeamConnection BUILD of testpart.exe then testpart.c would need to be "touched" if you wanted to see TeamConnection BUILD an out of date part.

## Create 'TSOUSER.DEVELOP.SOURCE(TCBUILD)'

| Action | Description |
|---|---|
| On the **SCLM Main Menu** enter:<br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 2<br>Edit is usually option 2. | The "SCLM Edit - Entry Panel" appears. |
| On the **SCLM Edit - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** SOURCE<br>**Member** TCBUILD<br>Press the **Enter** key. | An SCLM EDIT session appears. |
| Enter the following line of dummy information:<br>**THIS IS DUMMY INFORMATION.** | |
| Press the **F3** function key to save the member. | The "SCLM Edit Profile" box appears. |
| In the "SCLM Edit Profile" box enter **TCBUILD** for the "Language". | The language definition TCBUILD (see Figure 25 on page 122) is assigned to this member. |
| Press the **Enter** key. | The "SCLM Edit - Entry Panel" appears. |

Create 'TSOUSER.DEVELOP.ARCHDEF(TCBUILD)': This section shows how to create member 'TSOUSER.DEVELOP.ARCHDEF(TCBUILD)' using SCLM EDIT and assigns the language ARCHDEF. This member identifies the source member to be built ('TSOUSER.DEVELOP.SOURCE(TCBUILD)').

The language definition TCBUILD tells SCLM to run FLMLTWST which transfers the contents of 'TSOUSER.DEVELOP.SOURCE(TCBUILD)' to the workstation and runs TeamConnection BUILD of part testpart.exe as per the ACTINFO file.

NOTE: If the architecture definition member exists from a previous demonstration it does not need to be recreated or edited and saved. It can be used "as is" and you can go directly to the SCLM BUILD of this member in this step.

| Action | Description |
|---|---|
| On the **SCLM Edit - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** ARCHDEF<br>**Member** TCBUILD<br>Press the **Enter** key. | An SCLM EDIT session appears. |
| Enter the following:<br>**SINC TCBUILD SOURCE** | |
| Press the **F3** function key to save the member. | The "SCLM Edit Profile" box appears. |
| In the "SCLM Edit Profile" box enter **ARCHDEF** for the "Language".<br>Press the **Enter** key. | The language definition ARCHDEF is assigned to this member. The project definition (Figure 26 on page 125) uses the standard architecture definition language FLM@ARCD.<br>The "SCLM Edit - Entry Panel" appears. |
| Press the **F3** function key. | The "SCLM Main Menu" appears. |

Build 'TSOUSER.DEVELOP.ARCHDEF(TCBUILD)': This section describes how to build member 'TSOUSER.DEVELOP.ARCHDEF(TCBUILD)'. The TeamConnection BUILD command is invoked and the part testpart.exe is built as per the TeamConnection BUILD command in the ACTINFO file (Figure 21 on page 110).

| Action | Description |
|---|---|
| On the **SCLM Main Menu** enter:<br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 4<br>Build is usually option 4. | The "SCLM Build - Entry Panel" appears. |
| On the **SCLM Build - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** ARCHDEF<br>**Member** TCBUILD<br>**Workstation Build** /<br>**Mode** 1<br>**Scope** 2<br>**Process** 1<br>**Messages** 1 3<br>**Report** 1 3<br>**Listings** 1 3<br>Press the **Enter** key. | A workstation window appears with the TeamConnection build progress messages while the TeamConnection BUILD command runs.<br>TeamConnection part testpart.exe will be up to date on the workstation after a successful TeamConnection BUILD. It can be extracted and run to demonstrate that the changes made in SCLM are present in the part. |

## Step 7 - Test the OS/2 Application

Extract the executable part **testpart.exe** and run it for the test.

| Action | Description |
|---|---|
| Select **Actions**, then **Parts**, and then **Extract** from the "TeamConnection - Tasks" window. | The "Extract Parts" window appears. |
| Set the following fields:<br>**Path names** testpart.exe<br>**Release** release1<br>**Work area** mvstcwa<br>**Destination directory** e:\mytemp<br>Press the **OK** push button. | |

| Action | Description |
| --- | --- |
| Go to an OS/2 window.<br>Run the command **e:\mytemp\testpart**.<br>Minimize or close the OS/2 window. | The messages from testpart.exe should appear:<br>I am testpart.c<br>I was changed on MVS!<br>I was also changed on SCLM!<br>I was built by the Team Connection integrated Build Tool!!!<br>The line "I was changed on MVS!" will appear if you ran the MVS example (“MVS to TeamConnection” on page 5). |

# TEAMCONNECTION TO SCLM

The connection between TeamConnection and SCLM is through the TeamConnection BUILD function. The concept is to place a part in a build tree with a builder that runs IRXJCL on MVS which then runs a REXX program containing an SCLM command. TR 29.2196 USING REXX COMMAND FILES WITH TEAMCONNECTION MVS BUILD SCRIPTS describes this concept in detail for MVS tools such as precompilers, compilers, and linkage editors. Running an SCLM command requires a slightly more complex REXX program because it needs to use the ISPSTART function in order to run FLMCMD (the service interface to SCLM).

## TEAMCONNECTION TO SCLM - BUILD

This example uses the TeamConnection family described in Appendix B, “TeamConnection Family Setup” on page 81 and the SCLM project definition described in Appendix D, “SCLM Project Definition Setup” on page 103. The example that follows shows how to construct a build tree with a part that contains parameters for FLMCMD. (see Figure 14 on page 71). An MVS REXX program (see Figure 29 on page 128) creates an FLMCMD parameter string from these parameters and runs the FLMCMD for the SCLM BUILD.

### Setup Requirements

The TeamConnection family mvssclm and parts must be installed on the workstation (see Appendix B, “TeamConnection Family Setup” on page 81).

The SCLM project definition SCLMTC and parts must be installed on the MVS system (see Appendix D, “SCLM Project Definition Setup” on page 103).

### Step 1 - Workstation preliminary activities

| Action | Description |
|---|---|
| Be sure these environment variables are set in config.sys:<br>**TC_FAMILY** mvssclm<br>**TC_USER** SuprUser<br>**TC_BECOME** SuprUser<br>**TC_DBPATH** e:\family\mvssclm | |
| Start the ObjectStore server. | |
| Start the family mvssclm. | |

| Action | Description |
|---|---|
| From an OS/2 window issue the command<br>**teamagnt -f mvssclm -e tso -p pool1 -s tsosock -k IBM-85** | The MVS build agent starts in the OS/2 window.<br>The MVS build processor will be started on MVS by submitting a job that runs TEAMPROC from TSO. |

## Step 2 - MVS preliminary activities

| Action | Description |
|---|---|
| LOGON to TSO. | |
| Submit the job that runs the TeamConnection **RUNPGMT** program (it might be in the data set 'TSOUSER.TEAMC.JCLLOAD(RUNPGMT)').<br>See Figure 31 on page 133 for an example of the JCL. | This job should be active on the MVS job queue and waiting for a file from the MVS build agent. |
| Allocate the partitioned data set **'TSOUSER.TEAMC.REXX'** (or some other name) as FB with LRECL=80. You can use an existing partitioned data set.<br>Create the following members:<br>**SCLMCMD** with the contents of Figure 29 on page 128<br>**EXFLMCMD** with the contents of Figure 30 on page 132 | See Appendix A, “TeamConnection Parts and Scripts” on page 69 for information about soft copies of the REXX programs. |

## Step 3 - Create TeamConnection builder and parts for an SCLM BUILD

This section creates the tso builder and parts for the TeamConnection build tree that will result in running an SCLM BUILD command on MVS. A TeamConnection part contains parameters for the FLMCMD service that runs the SCLM BUILD. (As an alternative the part containing these parameters may be stored on MVS as a member of a partitioned data set. See TR 29.2196 USING REXX COMMAND FILES WITH TEAMCONNECTION MVS BUILD SCRIPTS for a discussion of the technique.)

NOTE: If the input part sclmbild.prm exists from a previous demonstration and has not changed since the last time it was built you must "touch" the input part sclmbild.prm in order for a build event to be placed on the job queue. After touching the input part you can go to “Step 4 - Connect TeamConnection parts in a build tree” on page 43.

| Action | Description |
|---|---|
| Modify e:\tcdemo\mvssclm\scripts\sclmcmd.jcl to have SYSEXEC DD point to the data set created in “Step 2 - MVS preliminary activities” on page 40. | Changes to the MVS build script must be made before creating the builder in order to avoid modifying the builder properties later.<br>Changes to a build script are only picked up by the builder when you follow the changes with a builder "Modify properties". |
| From the "TeamConnection - Tasks" window, select **Actions**, then **Builders**, and then **Create** on client work-station. | The "Create Builders" window appears. |
| Enter the builder information shown in Figure 5, pressing the **Apply** push button after each entry. | |
| Press the **Cancel** push button to return to the Tasks list. | The "TeamConnection - Tasks" window appears. |

```
Builder          SCLMjcl
Release          release1
Script           SCLM.jcl
Environment      tso
Comp. oper.      ==
RC value
File type        text
Source file      e:\tcdemo\mvssclm\scripts\sclmcmd.jcl
Parameters       'SCLMCMD'
Timeout          1
```

**Figure 5. TSO builder information for mvssclm family**

| Action | Description |
|---|---|
| From the "TeamConnection - Tasks" window, select **Actions**, then **Parts**, and then **Create**. | The "Create Parts" window appears. |
| Set the following fields for each part:<br>**Part names** see Table 1 on page 42<br>**Release** release1<br>**Work area** mvstcwa<br>**Component** projname<br>**File type** see Table 1 on page 42<br>**Source** see Table 1 on page 42<br>**Source file** see Table 1 on page 42<br>**Builder** see Table 1 on page 42<br>**Parser** see Table 1 on page 42<br>Enter the parts and information shown in Table 1 on page 42, pressing the **Apply** push button after each entry. | |
| Press the **Cancel** push button to return to the Tasks list. | The "TeamConnection - Tasks" window appears. |

Table 1. projname Create Part Information

| Part name | File type | Source | Source file | Builder | Parser |
|---|---|---|---|---|---|
| sclmbild.prm | Text | Copy from | e:\tcdemo\mvssclm \parts\sclmbild.prm | | |
| sclmbild.dmy | None | | | SCLMjcl | |

## Step 4 - Connect TeamConnection parts in a build tree

FLMCMD parameters to run the SCLM BUILD command are located in part sclmbild.prm (Figure 14 on page 71) that is connected as an input to part sclmbild.dmy. Part sclmbild.dmy has the builder SCLMjcl (Figure 17 on page 89) with the MVS build script (Figure 28 on page 127).

Part sclmbild.dmy could be connected as an input to a collector part in a larger build tree for a more interesting example of a coordinated build between TeamConnection and SCLM.

NOTE: If the parts are connected as in Figure 6 on page 44 from a previous demonstration you can go to “Step 5 - Create an SCLM controlled member.” on page 45.

| Action | Description |
|---|---|
| Select **Objects**, then **Parts**, and then **Parts**. | The "Part Filter" window appears. |
| Set the following fields:<br>**Release** release1<br>**Work area** mvstcwa<br>In the "History" section, select **Show all Parts**<br>Press the **OK** push button. | The "TeamConnection - Parts" window appears. |
| For each icon in Table 2 on page 44:<br>click on the **icon** in the "TeamConnection - Parts" window<br>click on mouse button 2<br>click on **Connect** from the pop up<br>Set the following fields and radio buttons in the "Connect Parts" window:<br>**Parent** see Table 2 on page 44<br>**Relation to parent** see Table 2 on page 44<br>Press the **OK** push button. | The fields "Path names", "Type", "Release", and "Work area" should already have the correct values. |
| Close the "TeamConnection - Parts" window. | The "TeamConnection - Tasks" window appears. |
| Select **Objects**, then **Parts**, and then **BuildView**. | The "BuildView Filter" window appears. |

| Action | Description |
|---|---|
| Set the following fields:<br>**Release** release1<br>**Work area** mvstcwa<br>In the "History" section, select **Show all BuildView**<br>Press the **OK** push button. | The "TeamConnection - BuildView" window appears. |
| Expand the **sclmbild.dmy** icon to show the entire tree. | Examine the tree for proper structure (see Figure 6 on page 44). |
| Close the "TeamConnection - BuildView" window. | The "TeamConnection - Tasks" window appears. |

Table 2. release1 Build Tree Information

| Icon | Parent | Relation |
|---|---|---|
| sclmbild.prm | sclmbild.dmy | input |

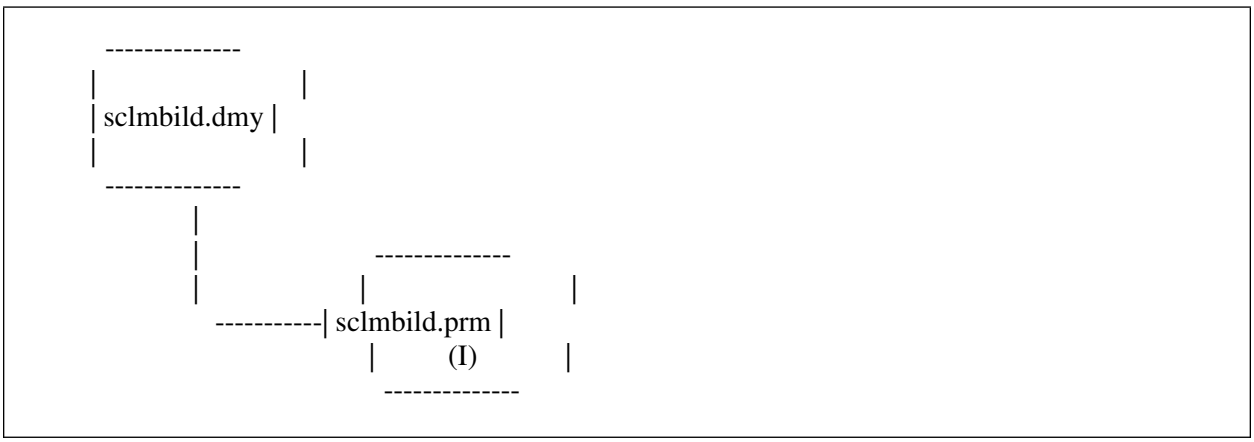



**Figure 6. Build tree with part sclmbild.prm**

## Step 5 - Create an SCLM controlled member.

Member 'TSOUSER.DEVELOP.SOURCE(PROGRAM1)' will be created using SCLM EDIT and assigned the language DUMMY. This member will represent a source program. A more complicated example could use a COBOL program and a COBOL language definition. The simple language definition DUMMY is used for demonstration purposes.

NOTE: If the member exists from a previous demonstration you must either delete the member using the SCLM library utility and follow the instructions below or use SCLM EDIT and SAVE the member so that it will be updated for the SCLM BUILD (remember to enter "SAVE" on the command line in the SCLM EDIT session).

| Action | Description |
|---|---|
| Select **SCLM** from the "ISPF Primary Option Menu".<br><br>This is usually option 10. | The "SCLM Main Menu" appears. |
| On the **SCLM Main Menu** enter:<br><br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 2<br><br>Edit is usually option 2. | The "SCLM Edit - Entry Panel" appears. |
| On the **SCLM Edit - Entry Panel** enter:<br><br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** SOURCE<br>**Member** PROGRAM1<br><br>Press the **Enter** key. | An SCLM EDIT session appears. |
| Enter the following line of source code for PROGRAM1:<br><br>**THIS IS SOURCE CODE.** | |
| Press the **F3** function key to save the member. | The "SCLM Edit Profile" box appears. |
| In the "SCLM Edit Profile" box enter **DUMMY** for the "Language".<br><br>Press the **Enter** key.<br><br>Press the **F3** function key. | The simple language definition DUMMY (see Figure 22 on page 114) that is convenient for demonstration purposes is assigned to this member.<br><br>The "SCLM Edit - Entry Panel" appears.<br><br>The "SCLM Main Menu" appears. |

## Step 6 - Create an SCLM architecture definition member

Member 'TSOUSER.DEVELOP.ARCHDEF(APPL1)' will be created using SCLM EDIT and assigned the language ARCHDEF. This member identifies the source member to be built (PROGRAM1).

NOTE: If the architecture definition member exists from a previous demonstration it does not need to be recreated or edited and saved. It can be used "as is" and you can go directly to “Step 7 - Run an SCLM BUILD from a TeamConnection BUILD” on page 47.

| Action | Description |
|---|---|
| On the **SCLM Main Menu** enter:<br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 2<br>Edit is usually option 2. | The "SCLM Edit - Entry Panel" appears. |
| On the **SCLM Edit - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** ARCHDEF<br>**Member** APPL1<br>Press the **Enter** key. | An SCLM EDIT session appears. |
| Enter the following:<br>**SINC PROGRAM1 SOURCE** | |
| Press the **F3** function key to save the member. | The "SCLM Edit Profile" box appears. |
| In the "SCLM Edit Profile" box enter **ARCHDEF** for the "Language".<br>Press the **Enter** key.<br>Press the **F3** function key. | The language definition ARCHDEF will be assigned to this member. The project definition (Figure 26 on page 125) uses the standard architecture definition language FLM@ARCD.<br>The "SCLM Edit - Entry Panel" appears.<br>The "SCLM Main Menu" appears. |

## Step 7 - Run an SCLM BUILD from a TeamConnection BUILD

This TeamConnection BUILD is the same as any TeamConnection BUILD involving an MVS build processor. In this demonstration a BUILD of sclmbild.dmy puts a build event on the job queue. The MVS build agent sends the build script and FLMCMD parameters to the MVS build processor. The MVS build processor runs IRXJCL as per the build script which will then run an MVS REXX program containing the FLMCMD service interface to the SCLM BUILD.

After a successful BUILD you can examine the build map using the "SCLM Utilities" for 'TSOUSER.SCLMTC.ARCHDEF(APPL1)'. The BUILD date and time will be updated to show the date and time of this SCLM BUILD.

| Action | Description |
|---|---|
| Select **Objects**, then **Parts**, and then **BuildView**. | The "BuildView Filter" window appears. |
| Set the following fields:<br><br>**Release** release1<br>**Work area** mvstcwa<br><br>In the "History" section, select **Show all BuildView** and press the **OK** push button. | The "TeamConnection - BuildView" window appears. |
| Click on **sclmbild.dmy**. | |
| Select **Selected**, and then **Build**. | The "Build Parts" window appears. |
| Set the following fields and radio buttons:<br><br>**Path name** sclmbild.dmy<br>**Type** TCPart<br>**release** release1<br>**Work area** mvstcwa<br>**Pool** pool1<br>**Build mode** normal<br>**Parameters**<br>**Detail file**<br><br>Press the **OK** push button. | |
| Monitor the "Build Progress" window, press the **OK** push button when the build is complete. | The "TeamConnection - BuildView" window appears. |
| Close the "TeamConnection - BuildView" window. | The "TeamConnection - Tasks" window appears. |

## Step 8 - Examine SCLM build map

The build map for member 'TSOUSER.DEVELOP.ARCHDEF(APPL1)' is examined to demonstrate that the SCLM BUILD actually took place.

| Action | Description |
|---|---|
| On the **SCLM Main Menu** enter:<br>**Project** TSOUSER<br>**Alternate** SCLMTC<br>**Group** DEVELOP<br>**Option ===>** 3<br>Utilities is usually option 3. | The "SCLM Utilities Menu" appears. |
| On the **SCLM Utilities Menu"** select **Library**.<br>Library is usually option 1. | The "SCLM Library Utility - Entry Panel" appears. |
| On the **SCLM Library Utility - Entry Panel** enter:<br>**Project** TSOUSER<br>**Group** DEVELOP<br>**Type** ARCHDEF<br>**Member** APPL1<br>**Option ===>** M<br>Press the **Enter** key. | A display of the "TSOUSER.DEVELOP.ARCHDEF(APPL1): Build Map Record" appears.<br>The "Change Date" and "Change Time" should be the same as when the member was built from TeamConnection. |
| Press the **F3** function key. | The "SCLM Library Utility - Entry Panel" appears. |
| Press the **F3** function key. | The "SCLM Utilities Menu" appears. |
| Press the **F3** function key. | The "SCLM Main Menu" appears. |

# COORDINATED TEAMCONNECTION AND SCLM BUILD

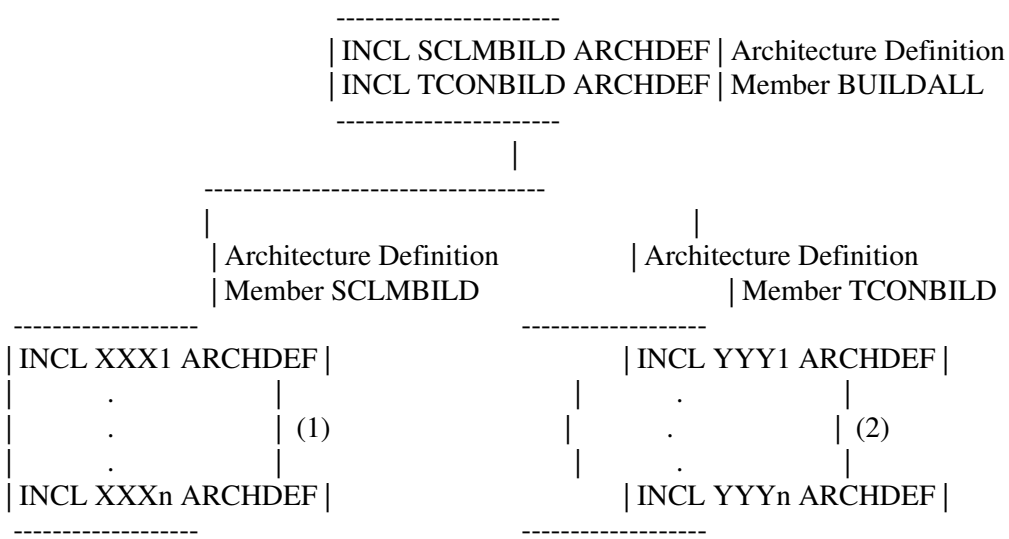
This section ties together the application development environment of TeamConnection with that of SCLM for BUILD activity. The techniques presented in this section expand upon those of the previous section.

The section “BUILD from SCLM or TeamConnection” on page 55 presents a TeamConnection build tree and SCLM architecture definition structure that reference each other. A TeamConnection BUILD of the top collector part in the build tree will yield similar results as an SCLM BUILD of the top high level architecture definition member. A BUILD from either TeamConnection or SCLM produces the same application outputs.

## TEAMCONNECTION BUILD FROM SCLM

An application might exist as a combination of members of data sets controlled by SCLM and parts stored in a TeamConnection family. The techniques described in this paper can be used to create an SCLM architecture definition that has one member for each TeamConnection build tree to be built. Each of these architecture definition members references a source member whose language definition uses the ISPF SELECT function to run an MVS REXX program which then runs a TeamConnection BUILD.

As shown in Figure 7 on page 50, one way of adding a TeamConnection BUILD to an existing SCLM architecture definition structure is by creating two additional high level architecture definition members. One member points to the architecture definition members for the TeamConnection BUILD while the other points to the existing SCLM structure. This provides the ability to build both the SCLM members and TeamConnection parts, only the SCLM members, or only the TeamConnection parts.




The architecture definition in box (1) builds SCLM controlled members whose language definitions use MVS translators such as compilers and linkage editors. It controls the existing applications under SCLM.

The architecture definition in box (2) builds SCLM controlled members whose language definitions use the ISPF SELECT service and WSCMDV to run TeamConnection build commands. It controls the applications that exist under TeamConnection. For example, a build of this member could result in a build of collector part tcbuild.col in Figure 8 on page 53.

When an SCLM BUILD command is issued for member BUILDALL both SCLM and TeamConnection builds will occur.

When an SCLM BUILD command is issued for SCLMBILD then only the members for applications controlled by SCLM will be built.

When an SCLM BUILD command is issued for TCONBILD then only the parts for applications controlled by TeamConnection will be built.

NOTE: A change in a TeamConnection part outside of SCLM will not update the SCLM accounting records. The "dummy" parts with the TeamConnection BUILD command language definitions must be "out of date" in order to be built without a "forced" SCLM build.

A "forced" SCLM build does not change the TeamConnection build to "force". The TeamConnection build executed by WSCMDV must specify "force" in order to run a "force" TeamConnection build. However, SCLM does have the provision for a user-written build exit that runs after the SCLM build. This build exit receives a parameter list that includes the SCLM build mode. The exit could examine the parameter list and run a "force" TeamConnection build if the SCLM build mode was "forced".

**Figure 7 (Part 1 of 2). Referencing TeamConnection build trees from SCLM architecture definitions**

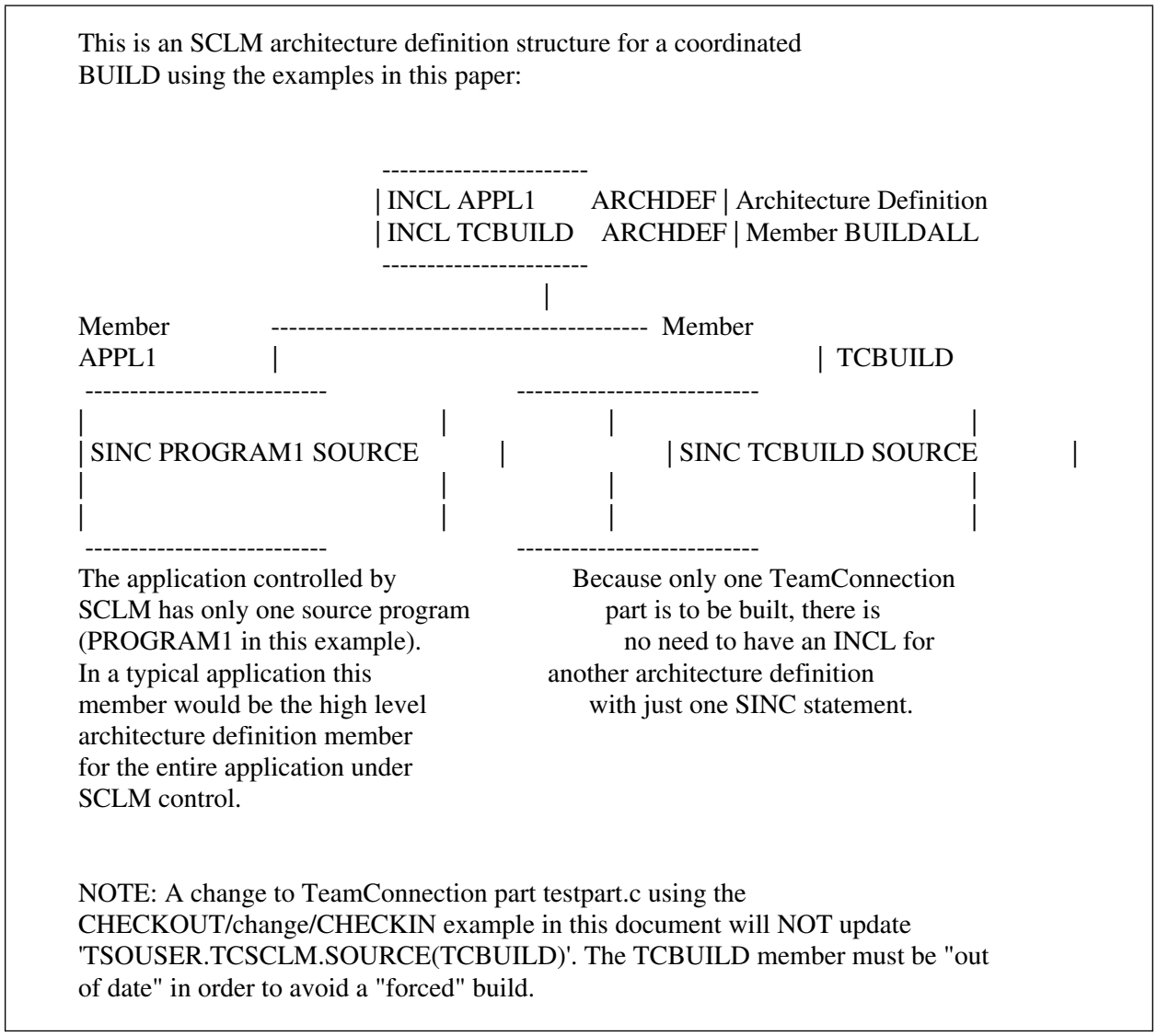


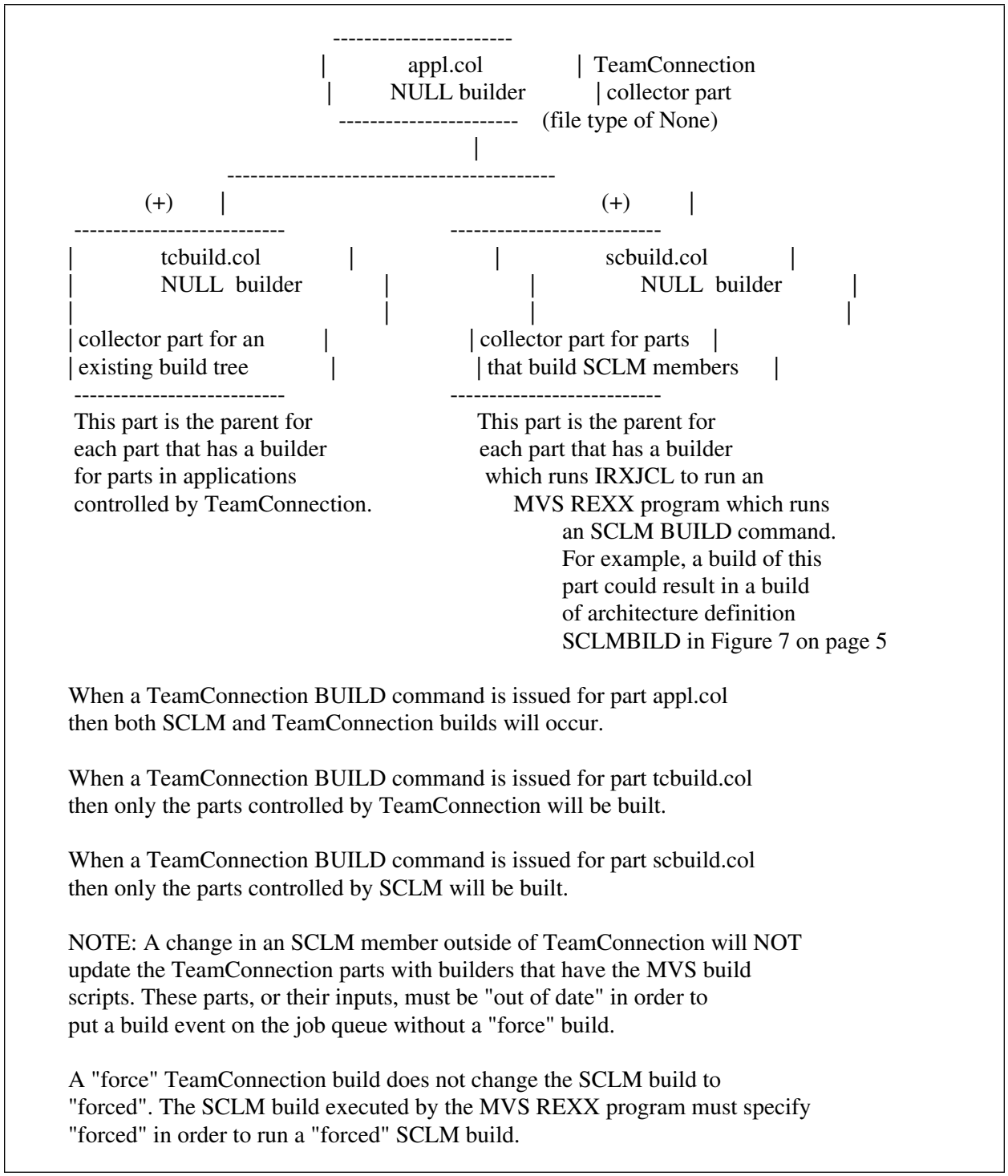

**Figure 7 (Part 2 of 2). Referencing TeamConnection build trees from SCLM architecture definitions**

## SCLM BUILD FROM TEAMCONNECTION

An application might exist as a combination of parts stored in a TeamConnection family and members of data sets controlled by SCLM. The techniques described in this paper can be used to create a TeamConnection build tree that has a part for each SCLM controlled application to be built. Each of these parts has a builder that will run an MVS REXX program to do an SCLM BUILD.

As shown in Figure 8 on page 53, one way of adding an SCLM BUILD to an existing TeamConnection build tree is by creating two additional parts. One part contains FLMCMD parameters for building a specific SCLM architecture definition member (normally the high level member for an application). The other part is the parent of the FLMCMD parameters part and has a builder that runs IRXJCL which then runs an MVS REXX program. This parent part could be connected to a collector part in an existing build tree.

This type of tree structure provides the ability to build both the TeamConnection parts and SCLM members, only the TeamConnection parts, or only the SCLM members.




When a TeamConnection BUILD command is issued for part appl.col
then both SCLM and TeamConnection builds will occur.

When a TeamConnection BUILD command is issued for part tcbuild.col
then only the parts controlled by TeamConnection will be built.

When a TeamConnection BUILD command is issued for part scbuild.col
then only the parts controlled by SCLM will be built.

NOTE: A change in an SCLM member outside of TeamConnection will NOT
update the TeamConnection parts with builders that have the MVS build
scripts. These parts, or their inputs, must be "out of date" in order to
put a build event on the job queue without a "force" build.

A "force" TeamConnection build does not change the SCLM build to
"forced". The SCLM build executed by the MVS REXX program must specify
"forced" in order to run a "forced" SCLM build.

**Figure 8 (Part 1 of 2). Referencing SCLM architecture definitions from TeamConnection build trees**

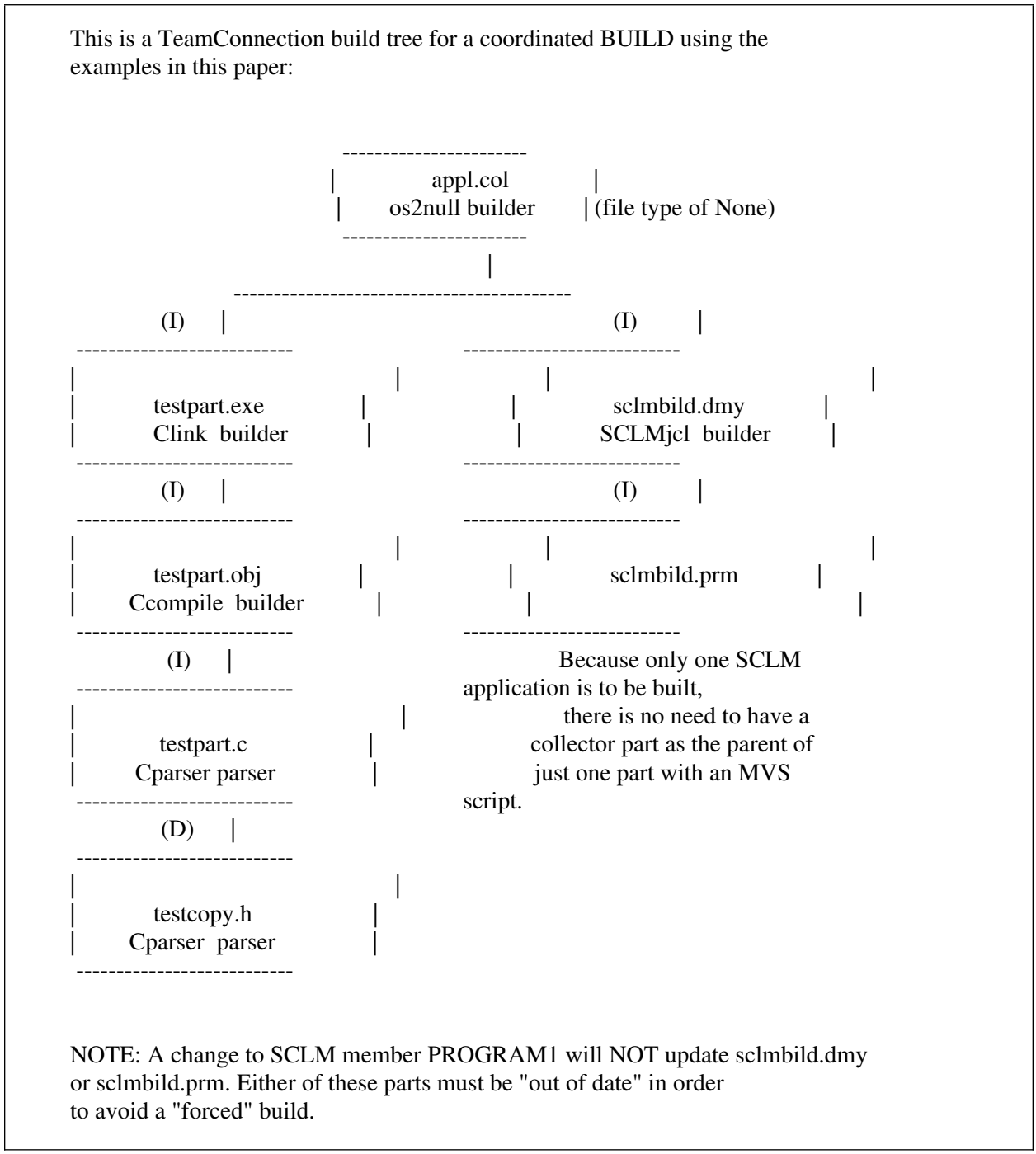



**Figure 8 (Part 2 of 2). Referencing SCLM architecture definitions from TeamConnection build trees**

## BUILD FROM SCLM OR TEAMCONNECTION

You can develop an application partly in SCLM and partly in TeamConnection with the BUILD being performed from either environment. You can build the highest level SCLM architecture definition member or the highest TeamConnection collector part in the build tree. Some coordination is required between the two platforms in order to be sure that all members and parts are examined for a build.

In order for all SCLM members to be examined from a TeamConnection BUILD, the TeamConnection parts that result in SCLM builds must be marked as out of date when the SCLM members are out of date. Similarly, in order for all TeamConnection parts to be examined from an SCLM BUILD, the SCLM members that result in TeamConnection builds must be marked as out of date when the TeamConnection parts are out of date. However, SCLM does not automatically mark SCLM members as out of date following changes to TeamConnection parts and TeamConnection does not automatically mark TeamConnection parts as out of date following changes to SCLM parts. Some coordination of updates between SCLM and TeamConnection is required in order to have a normal build started from either TeamConnection or SCLM examine each SCLM member and TeamConnection part in the build scope.

### Marking SCLM members out of date

The TeamConnection user needs to be sure that the SCLM members that correspond to the TeamConnection BUILD of the TeamConnection parts being changed are marked as being out of date so that a normal BUILD can be done from SCLM. This coordination of updates can be done manually or automatically.

The manual coordination in SCLM can be done by editing and saving the dummy source member(s) that are used for a TeamConnection BUILD or by deleting their build maps.

The automatic coordination of TeamConnection updates with SCLM can be done by using the SCLM DELETE service to delete the build maps of members whose language definition invokes a TeamConnection BUILD. This service can be invoked from a REXX program using the FLMCMD service interface just like the SCLM BUILD is invoked from TeamConnection. The TeamConnection user could use a part with parameters for the SCLM DELETE service along with an appropriate MVS REXX program to delete the build maps of the dummy source members that invoke the TeamConnection BUILD at any time prior to issuing a TeamConnection BUILD. Ideally TeamConnection would automatically issue the SCLM DELETE service for the appropriate SCLM member build maps when TeamConnection recognizes a TeamConnection part is out of date.

## Marking TeamConnection parts out of date

The SCLM user needs to be sure that the TeamConnection parts that correspond to the SCLM BUILD of the SCLM members being changed are marked as being out of date so that a BUILD may be done from TeamConnection. This coordination of updates can be done manually or automatically. In TeamConnection this can be done by "touching" each part that is an input to a part with an MVS build script that runs an SCLM BUILD.

The automatic coordination of SCLM updates with TeamConnection can be done by using the TeamConnection command "part -touch" to mark the parts that invoke an SCLM BUILD as being stale and in need of a build. This command can be invoked from MVS with a REXX program or from an SCLM BUILD of a member with a language definition that uses the ISPF SELECT service to run FLMLTWST and an appropriate command in the ACTINFO file (Figure 21 on page 110). The SCLM user could run the TeamConnection "part -touch" commands for the parts that invoke an SCLM BUILD at any time prior to issuing a TeamConnection BUILD. Ideally SCLM would automatically issue the TeamConnection "part -touch" commands for the appropriate TeamConnection parts when SCLM recognizes an SCLM member is out of date.

## Doing the builds from SCLM

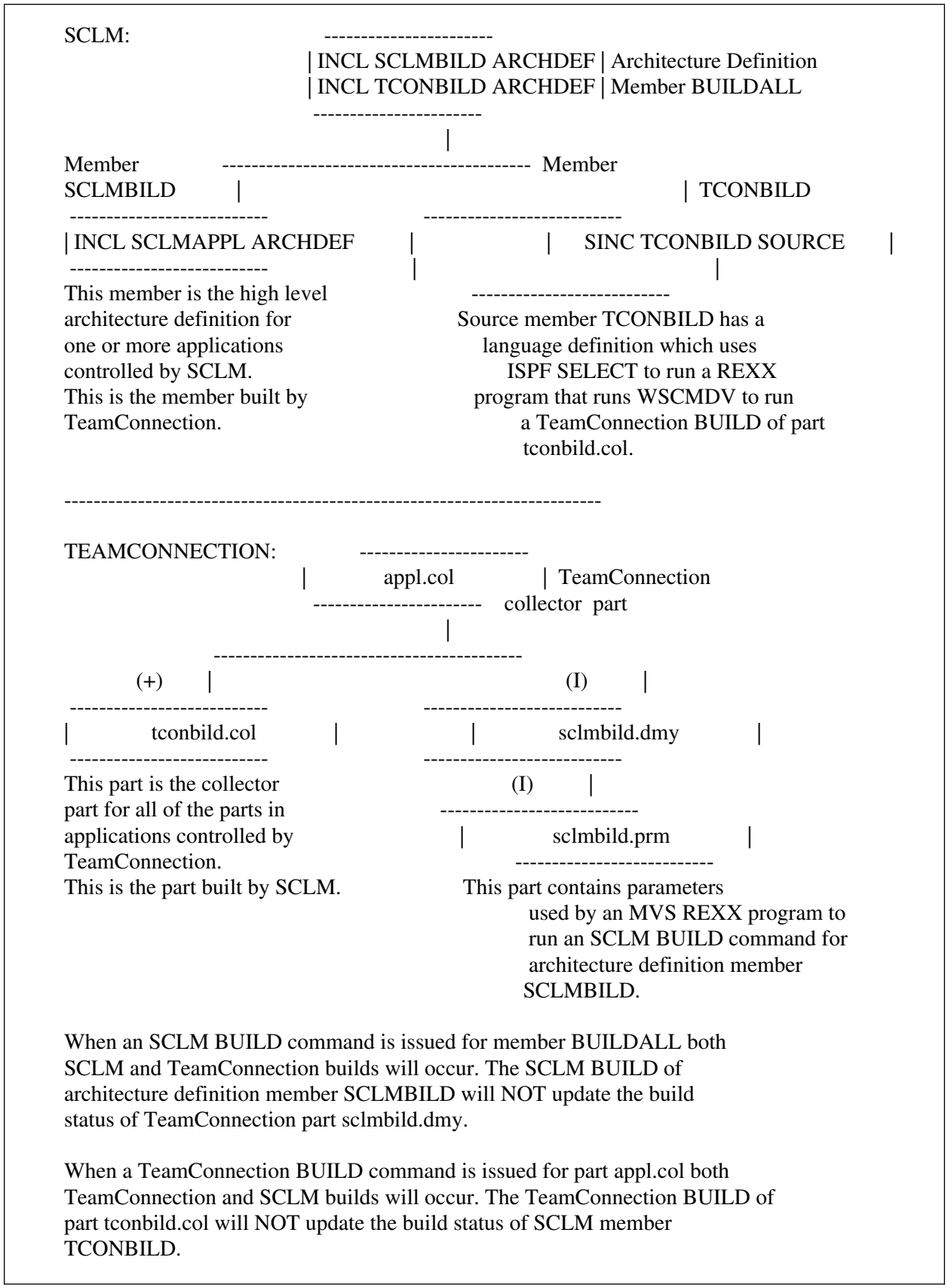
An SCLM BUILD results in the SCLM members being built and the TeamConnection parts for the applications controlled by TeamConnection being built. The TeamConnection parts which BUILD the SCLM members are not involved in this build and remain out of date. (If they were involved there would be a possibility of an infinite loop or a "lockup" because SCLM tells TeamConnection to BUILD which then tells SCLM to BUILD and so on.) A small amount of overhead results when the next TeamConnection BUILD issues an SCLM BUILD command for SCLM members that have not changed since the last SCLM BUILD.

## Doing the builds from TeamConnection

A TeamConnection BUILD results in the TeamConnection parts being built and the SCLM members for the applications controlled by SCLM being built. The SCLM members which BUILD the TeamConnection parts are not involved in this build and remain out of date. (If they were involved there would be a possibility of an infinite loop or a "lockup" because TeamConnection tells SCLM to BUILD which then tells TeamConnection to BUILD and so on.) A small amount of overhead results when the next SCLM BUILD issues a TeamConnection BUILD command for TeamConnection parts that have not changed since the last TeamConnection BUILD.

## Summary

The important point is that a BUILD command can be run in both environments without requiring a forced build. This is illustrated in the example shown in Figure 9 on page 57 which is a combination of the techniques shown in Figure 7 on page 50 and Figure 8 on page 53.

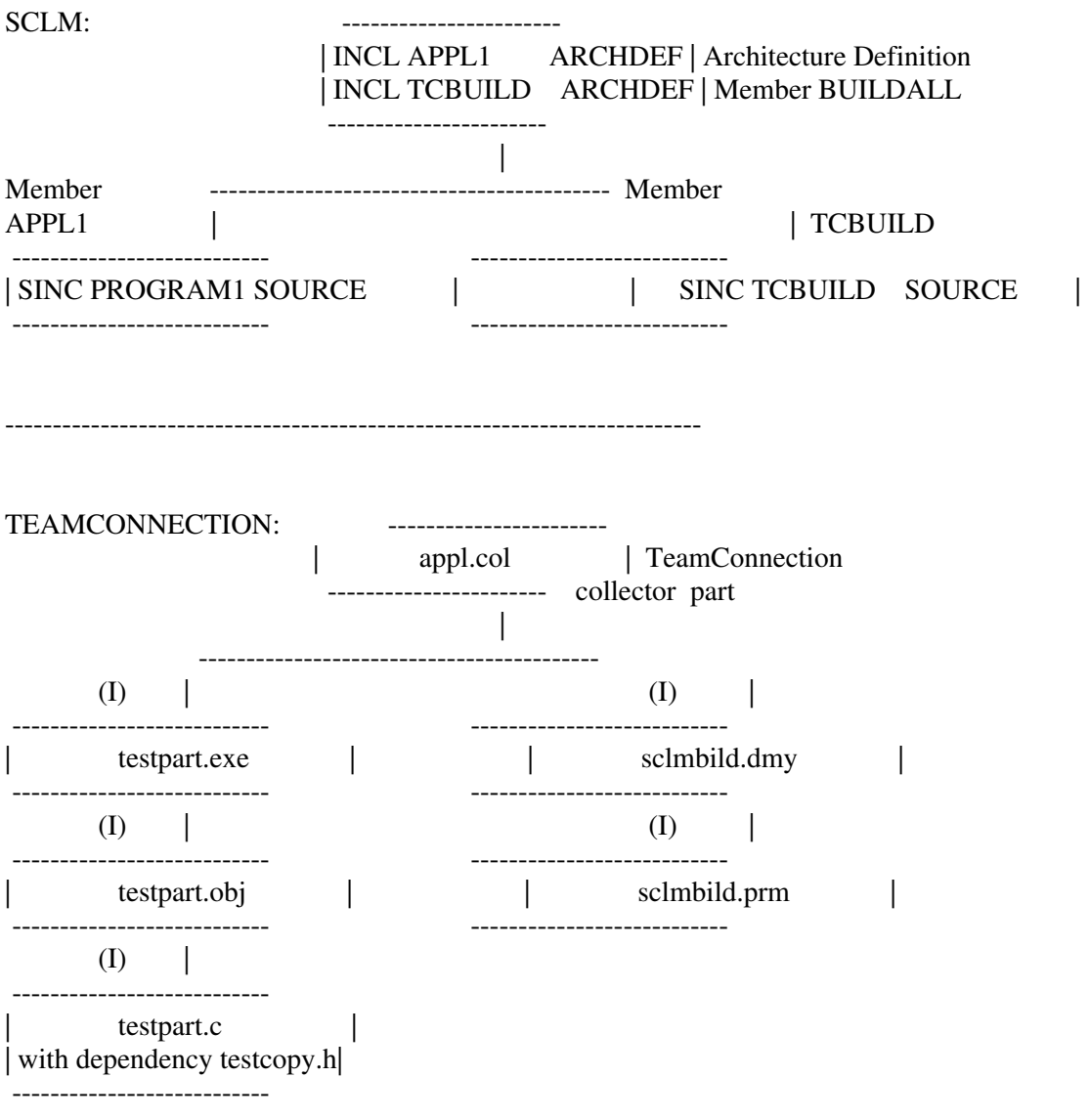



When an SCLM BUILD command is issued for member BUILDALL both SCLM and TeamConnection builds will occur. The SCLM BUILD of architecture definition member SCLMBILD will NOT update the build status of TeamConnection part sclmbild.dmy.

When a TeamConnection BUILD command is issued for part appl.col both TeamConnection and SCLM builds will occur. The TeamConnection BUILD of part tconbild.col will NOT update the build status of SCLM member TCONBILD.

**Figure 9 (Part 1 of 2). Example of building from SCLM or TeamConnection**

Using the examples in this paper these structures look like:




When an SCLM BUILD command is issued for member BUILDALL both SCLM and TeamConnection builds will occur. The SCLM BUILD is for APPL1 and TCBUILD. The BUILD of architecture member TCBUILD causes a TeamConnection BUILD for part testpart.exe. TeamConnection part sclmbild.dmy will not be built and the build status of this part will not change.

When a TeamConnection BUILD command is issued for part appl.col both SCLM and TeamConnection builds will occur. Build events for testpart.exe and sclmbild.dmy are created. The BUILD of part sclmbild.dmy causes an SCLM BUILD for architecture definition member APPL1. SCLM member TCBUILD will not be built and the build status of this member will not change.

**Figure 9 (Part 2 of 2). Example of building from SCLM or TeamConnection**

# SCLM ARCHITECTURE DEFINITION PARAMETERS FOR TEAMCONNECTION COMMANDS

## ADDITIONAL PARAMETERS RECOGNIZED BY THE EXTENDED FLMLTWST

The TeamConnection command, action flags, option lists, attribute flags, and arguments are hard coded in the ACTINFO file (Figure 21 on page 110) used in the previous examples. This is extremely restrictive for general use but is necessary due to the design of FLMLTWST as found in ISPF 4.2.

The design of FLMLTWST also causes transfer of the SCLM input members from MVS to the workstation and transfer of the output files from the workstation to MVS. These file transfers are not always required for TeamConnection commands. For example, a TeamConnection part checkout command does not require an input member to be transferred to the workstation and a TeamConnection part checkin command does not require an output file to be transferred to MVS.

The SCLM PARM and PARMx architecture definition statements can be used to add parameters to the FLMLTWST parameter string at run time. FLMLTWST can be extended to examine these parameters and to run any TeamConnection command based upon these parameter values and the use of a general purpose ACTION statement in the ACTINFO file. Parameters can also be added to avoid an unnecessary transfer of files to and from the workstation.

The file flmltwst.rex provided with the demonstration materials (see Appendix A, "TeamConnection Parts and Scripts" on page 69) is the SCLM FLMLTWST from ISPF 4.2 extended through code changes to recognize the following parameters in an SCLM architecture definition PARM or PARMx statement:

SENDINPUT

- YES - send inputs from MVS to the workstation
- NO - do not send inputs to the workstation
- DEFAULT = YES

GETOUTPUT

- YES - send outputs from the workstation to MVS
- NO - do not send outputs to MVS
- DEFAULT = YES

TCACTFLAG

- the value 'TCACTFLAG' in a 'CPARM TCACTFLAG' statement from an ACTINFO file (Figure 21 on page 110) is replaced with the value of the TCACTFLAG keyword parameter
- this parameter is intended to be a TeamConnection command action flag but can be used for any CPARM statement
- DEFAULT = -build

PARTNAME

- the value 'PARTNAME' in a 'CPARM PARTNAME' statement from an ACTINFO file (Figure 21 on page 110) is replaced with the value of the PARTNAME keyword parameter
- this parameter is intended to be a TeamConnection command action flag option list but can be used for any CPARM statement
- DEFAULT = PARTNAME

There is nothing special about the names, values, and defaults of these parameters. FLMLTWST is provided by SCLM as an example to be extended for other purposes such as this. The extensions found in file flmltwst.rex provided with the demonstration materials (see Appendix A, “TeamConnection Parts and Scripts” on page 69) can be used as examples for other extensions to provide values through parameters on the PARM and PARMx architecture definition statements. The code changes for these extensions are described in Appendix F, “SCLM FLMLTWST Extensions for Parameters” on page 135.

The character strings following the PARM and PARMx in architecture definition statements are concatenated to the OPTIONS keyword parameter value of the FLMTRNSL macro in the SCLM language definition at run time. Leading and trailing blanks are removed and a leading comma for the string from the first PARM statement and the string from each PARMx statement is added to the concatenation. The character strings could also be part of the OPTIONS keyword value coded in the language definition if they are to be applied to all SCLM BUILDs. The PARM and PARMx architecture definition statements provide a means of changing the FLMLTWST parameter string at run time based upon the member being built.

The following sections describe the techniques in more detail:

“Parameters to Avoid File Transfers” on page 63 describes the use of SENDINPUT and GETOUTPUT

“General Purpose ACTION in the ACTINFO File” on page 64 describes the use of TCACTFLAG and PARTNAME

## PARAMETERS TO AVOID FILE TRANSFERS

The transfer of SCLM input members to the workstation can be avoided by adding this PARM statement to the architecture definition that is built for running a TeamConnection command:

**PARM SENDINPUT=NO**

The transfer of workstation output files to MVS can be avoided by adding this PARM statement to the architecture definition that is built for running a TeamConnection command:

**PARM GETOUTPUT=NO**

The transfer of both SCLM input members to the workstation and workstation output files to MVS can be avoided by adding this PARM statement to the architecture definition that is built for running a TeamConnection command:

**PARM SENDINPUT=NO,GETOUTPUT=NO**

For example, building the architecture definition in “Step 3 - TeamConnection CHECKOUT issued from SCLM BUILD” on page 26 results in an unnecessary transfer of the SCLM input part to the workstation. You can avoid this input member transfer with the extended FLMLTWST program and a PARM statement. The architecture definition would look like this:

```
SINC CHECKOUT SOURCE
OUT1 TESTPART SRC
PARM SENDINPUT=NO
```

## GENERAL PURPOSE ACTION IN THE ACTINFO FILE

The ACTINFO file (Figure 21 on page 110) used in the previous examples has a specific ACTION statement for each TeamConnection command with hard-coded parameter values. The CPARM values that are used to construct the TeamConnection commands can be replaced by values of parameters found in the OPTIONS keyword parameter of the FLMTRNSL macro in a language definition or on PARM and PARMx architecture definition statements.

The following is an example of a general purpose ACTION that uses the **TCACTFLAG** and **PARTNAME** keyword parameters recognized by the extended FLMLTWST program in the demonstration materials. TeamConnection commands that do not use the following command format need a different ACTION. You can use additional keyword parameters and general purpose ACTION formats to handle other TeamConnection commands.

```
ACTION that uses the TeamConnection teamc part -build format

TCACTFLAG and PARTNAME are replaced by the keyword parameter
values during a BUILD when using the extended FLMLTWST

ACTION      TCOMMAND
COMMAND       teamc
CPARM        part
CPARM        TCACTFLAG
CPARM        PARTNAME
CPARM        -type
CPARM        TCPart
CPARM        -release
CPARM        release1
CPARM        -workarea
CPARM        mvstcwa
CPARM        -pool
CPARM        pool1
CPARM        -normal
```

**Figure 10. ACTINFO - general purpose ACTION**

The general purpose ACTION for TCOMMAND shown in Figure 10 can be used with the extended FLMLTWST by different methods:

The ACTION value in the OPTIONS keyword parameter for the FLMTRNSL macro in the language definition of the member being built can be set to **TCOMMAND**. This is described in “Using the ACTION value in FLMTRNSL” on page 65.

Advantage - the same ACTION is used by all members with the same language definition.

Disadvantage - additional language definitions are needed for using different ACTION statements in the ACTINFO file.

The ACTION value can be specified in a PARM or PARMx architecture definition statement and not in the OPTIONS keyword parameter for the FLMTRNSL macro of the member being built.

Advantage - the ACTION value used can be changed in the architecture definitions without the need for additional language definitions.

Disadvantage - the ACTION keyword parameter must be specified on all architecture definitions for parts with language definitions that use FLMLTWST and do not specify the ACTION value in the OPTIONS keyword parameter.

There is no single combination of language definitions, keyword parameters, and ACTINFO ACTIONs that meets the needs of all application development environments.

## USING THE ACTION VALUE IN FLMTRNSL

This example uses the ACTION in Figure 10 on page 64 and the extended FLMLTWST. The ACTION TCOMMAND is referenced in a language definition by specifying **ACTION=TCOMMAND** in the OPTIONS keyword parameter of the FLMTRNSL macro. Part of the language definition is in Figure 11. (See Figure 25 on page 122 for a complete example of a language definition.)

```
TCOMMAND FLMLANGL       LANG=TCCOMMAND,                                        C
               VERSION=1,CHKSYSLB=IGNORE

 EXECUTE TEAMCONNECTION TEAMC PART COMMAND WITH ACTION FLAG FOUND
 IN TCACTFLAG PARAMETER AND ACTION FLAG OPTION LIST FOUND
 IN PARTNAME PARAMETER. THE EXTENDED FLMLTWST MUST BE USED.

          FLMTRNSL     CALLNAM='TCOMMAND',                                    C
               FUNCTN=BUILD,                                                  C
               CALLMETH=ISPLNK,                                               C
               COMPILE=SELECT,                                                C
               VERSION=1,                                                     C
               GOODRC= ,                                                      C
               PORDER=1,                                                      C
               OPTIONS='CMD(FLMLTWST  ACTION=TCOMMAND,BMAPINFO=@@FLM$MP,C
               SCLMINFO=@@FLMINF,BLDINFO=@@FLMBIO,PARMS='
```

**Figure 11. FLMLANGL and FLMTRNSL for General Purpose ACTION**

In a previous example (“Step 6 - TeamConnection BUILD issued from SCLM BUILD” on page 33), a TeamConnection BUILD was run from SCLM using ACTION BUILD and a hard-coded TeamConnection command (Figure 21 on page 110) to build testpart.exe. The architecture definition in this previous example has one statement:

```
SINC TCBUILD SOURCE
```

In order to use ACTION TCOMMAND to run a TeamConnection BUILD for any part you can:

1. add a language definition for TCOMMAND (Figure 11 on page 65) to your SCLM project definition (do not forget to assemble and relink your SCLM project definition)
2. change the language of TSOUSER.DEVELOP.SOURCE(TCBUILD) from TCBUILD to TCOMMAND
3. use the PARTNAME keyword parameter in the architecture definition (TCACTFLAG defaults to -build in the extended FLMLTWST code so it does not need to be set)

   After making these changes, the architecture definition to run a TeamConnection build of part appl.col looks like this:

   ```
   SINC TCBUILD SOURCE
   PARM PARTNAME=appl.col
   ```

Another technique for using ACTION TCOMMAND to run a TeamConnection BUILD for any part you can:

1. remove ACTION=TCOMMAND from the OPTIONS value in the TCOMMAND language definition (Figure 11 on page 65) (do not forget to assemble and relink your SCLM project definition)
2. change the language of TSOUSER.DEVELOP.SOURCE(TCBUILD) from TCBUILD to TCOMMAND
3. After making these changes, the architecture definition to run a TeamConnection build of part appl.col looks like this (TCACTFLAG defaults to -build in the extended FLMLTWST code so it does not need to be set):

   ```
   SINC TCBUILD SOURCE
   PARM ACTION=TCOMMAND,PARTNAME=appl.col
   ```

## USING THE EXTENDED FLMLTWST WITH ISPF

In order to use a different FLMLTWST from the one delivered with SCLM you can:

1. put the FLMLTWST program in a private partitioned data set such as 'TSOUSER.FLMLTWST.REXX'
2. execute a CLIST prior to bringing up ISPF that will concatenate the private library to SYSEXEC (Figure 27 on page 126)

It is a good idea to design changes for an extended FLMLTWST that result in FLMLTWST running as it did before when you are not using parameters. If this is not possible, you can rename the extended FLMLTWST and add it and the original FLMLTWST to your private library.

# APPENDIX A. TEAMCONNECTION PARTS AND SCRIPTS

To use the materials, you must do the following:

Unzip tcdemo.zip using the following command:

```
pkunzip2 -d -o e:\mvssclm.zip e:\

    where the first e: is where the zip file is located
          the second e: is where you want to place the
               parts and commands for the demo
```

Create the following directory:

```
e:\mytemp
```

Determine the family name, port, and super user ID that you want to use for the demo. The values used in the demo are:

| | |
|---|---|
| **TC_FAMILY** | mvssclm |
| **TC_DBPATH** | e:\family\mvssclm |
| **TCP/IP address** | 9.37.195.43 |
| **port** | 9808 |
| **super user ID** | SuprUser in the demo script. |

Determine the TCP/IP addresses and sockets that you want to use for builds. The values used in the demo are:

| | |
|---|---|
| **TCP/IP address** | 9.37.195.43 |
| **Host name** | FamHost |
| **OS/2 Socket name** | bldsock |
| **OS/2 Socket number** | 9997 |
| **MVS Socket name** | tsosock |
| **MVS Socket number** | 32911 |

## DEMONSTRATION MATERIALS

The following materials are required for running the demonstrations.

1. C++ source parts (e:\tcdemo\mvssclm\parts).
   a. testpart.c (Figure 12 on page 71)
   b. testcopy.h (Figure 13 on page 71)
2. SCLM command parts (e:\tcdemo\mvssclm\parts).
   a. sclmbild.prm - FLMCMD parameters (Figure 14 on page 71)
3. Build scripts (e:\tcdemo\mvssclm\scripts)
   a. fhbocomp.cmd - C++ compile (Figure 15 on page 72)
   b. fhbolink.cmd - Linkage Edit (Figure 16 on page 76)
   c. sclmcmd.jcl - IRXJCL for running MVS REXX programs (Figure 28 on page 127)
4. MVS REXX programs (e:\tcdemo\mvssclm\rexx) - must be uploaded to MVS
   a. tcbuild.rex - TeamConnection BUILD command (Figure 4 on page 12)
   b. tcchkin.rex - TeamConnection CHECKIN command (Figure 3 on page 10)
   c. tcchkout.rex - TeamConnection CHECKOUT command (Figure 2 on page 7)
   d. sclmcmd.rex - construct FLMCMD parameter string (Figure 29 on page 128)
   e. exflmcmd.rex - run FLMCMD service interface to SCLM (Figure 30 on page 132)
   f. flmltwst.rex - extended FLMLTWST for use with architecture definition PARM satements
5. SCLM project definition parts (e:\tcdemo\mvssclm\sclm) - must be uploaded to MVS
   a. sclmtc.prj - project definition (Figure 26 on page 125)
   b. dummy.lng - dummy language definition (Figure 22 on page 114)
   c. checkout.lng - TeamConnection CHECKOUT language definition (Figure 23 on page 115)
   d. checkin.lng - TeamConnection CHECKIN language definition (Figure 24 on page 119)
   e. tcbuild.lng - TeamConnection BUILD language definition (Figure 25 on page 122)
   f. vsamallc.jcl - batch job to allocate VSAM account file (Figure 19 on page 108)
   g. actinfo.txt - ACTINFO file for workstation commands (Figure 21 on page 110)
   h. userinfo.txt - USERINFO file for workstation directories (Figure 20 on page 109)

## SIMPLE C++ PROGRAMS

```
/     testpart.c      /

#include <stdio.h>
#include "testcopy.h"

int main()
{
          printf("I am testpart.c\n");
          printf("I was built by the Team Connection\n");
          printf("integrated Build Tool!!!\n");
         return   ;
}
```

**Figure 12. testpart.c C++ Program**

## SIMPLE C++ PROGRAM INCLUDE

```
/  testcopy.h  /
```

**Figure 13. testcopy.h C++ Include**

## SCLM BUILD COMMAND PARAMETERS

```
BUILD
DEVELOP
ARCHDEF
APPL1
```

**Figure 14. sclmbild.prm SCLM BUILD command parameters**

## C++ BUILD SCRIPT

```
/ ------------------------------------------------------ /
/  PROGRAM:    FHBOCOMP.CMD                                        /
/  IBM C Set++ Sample Build Script TeamConnection             /
/  ICC C++ Compile Only                                          /
/                                                                 /
/ ------------------------------------------------------ /
/                  IBM TeamConnection for OS/2                   /
/                          Version 1 Release                     /
/                             5622-717                           /
/                                                                 /
/  (C) Copyright, IBM Corp., 1995. All Rights Reserved.  /
/                  Licensed Materials - Property of IBM          /
/                US Government Users Restricted Rights           /
/    - Use, duplication or disclosure restricted by           /
/           GSA ADP Schedule Contract with IBM Corp.            /
/                                                                 /
/                  IBM is a registered trademark of              /
/          International Business Machines Corporation          /
/                                                                 /
/  DISCLAIMER OF WARRANTIES:                                          /
/    The following enclosed code is sample code created     /
/    by IBM Corporation.    This sample code is not part of /
/    any standard product and is provided to you solely     /
/    for the purpose of assisting you in the development  /
/    of your applications.    The code is provide "AS IS",  /
/    without warranty of any kind. IBM shall not be           /
/    liable for any damages arising out of your use           /
/    of the sample code                                            /
/ ------------------------------------------------------ /
/  Inputs from the environment                                   /
/           TC_INPUT                                                /
/           TC_OUTPUT                                                 /
/                                                                 /
/  Inputs from the argument list                                 /
/           PARMS                                                   /
/                                                                 /
/    TC_INPUT contains inputs stored in the library that  /
/      are input to the translator build step.                /
/    TC_OUTPUT contains outputs that are to be built by     /
/      translator and that will be stored in the library     /
/    PARMS are translator options that are specified          /
/      on the TeamConnection builder definition.                 /
/                                                                 /
```

**Figure 15 (Part 1 of 2). fhbocomp.cmd C++ Build Script**

```
/  Outputs:                                                          /
/     files generated by the translator                          /
/                                                                      /
/  Returns:                                                          /
/     return code from the translator execution                  /
/ ------------------------------------------------------ /
    parse arg parms

   environ   =  'OS2ENVIRONMENT'
   input       =  VALUE('TC_INPUT',,environ)
   output      =  VALUE('TC_OUTPUT',,environ)

    translator        = 'icc'      /  specify the translator to be invoked  /
    default_parms = '/C+'      /  specify any default parms to be used  /
                                          /  for the compile                          /

    if parms = '' then parms = default_parms

   call  invokeTranslator

exit result /  exit with the rc return by the translator  /

/ ------------------------------------------------------ /
/  Routine: invokeTranslator                                        /
/                                                                      /
/  Function:                                                         /
/     This function invokes the translator defined by          /
/     the translator variable.    The default_parms define     /
/     translator options that are used for all translator  /
/     executions. The parm variable contains options           /
/     specified by the user to override the build script     /
/    default_parms.                                                  /
/                                                                      /
/  Variables used:                                                  /
/    translator,  default_parms                                    /
/     input, output, parms location, bomfile                     /
/                                                                      /
/  Returns:                                                          /
/     return code from the translator execution                  /
/ ------------------------------------------------------ /
invokeTranslator:

     /     This build script assumes only one input and     /
     /     one output are passed in.                              /

     translator parms '/Fo'||output input

return rc
```

**Figure 15 (Part 2 of 2). fhbocomp.cmd C++ Build Script**

## LINKAGE EDIT BUILD SCRIPT

```
/ ------------------------------------------------------ /
/  PROGRAM:    FHBOLINK.CMD                                          /
/  IBM ILINK Sample Build Script TeamConnection                 /
/  VisualAge C++ for OS/2 ILINK, Link Only                      /
/                                                                  /
/ ------------------------------------------------------ /
/                  IBM TeamConnection for OS/2                    /
/                           Version 1 Release                     /
/                              5622-717                           /
/                                                                  /
/  (C) Copyright, IBM Corp., 1995. All Rights Reserved.  /
/                  Licensed Materials - Property of IBM          /
/                US Government Users Restricted Rights           /
/    - Use, duplication or disclosure restricted by            /
/           GSA ADP Schedule Contract with IBM Corp.           /
/                                                                  /
/                  IBM is a registered trademark of              /
/         International Business Machines Corporation          /
/                                                                  /
/    The following enclosed code is sample code created     /
/    by IBM Corporation.    This sample code is not part of /
/    any standard product and is provided to you solely     /
/    for the purpose of assisting you in the development  /
/    of your applications.    The code is provide "AS IS",  /
/    without warranty of any kind. IBM shall not be           /
/    liable for any damages arising out of your use           /
/    of the sample code                                          /
/ ------------------------------------------------------ /
/  Inputs from the environment                                  /
/           TC_INPUT                                               /
/           TC_OUTPUT                                                /
/                                                                  /
/  Inputs from the argument list                                /
/           PARMS                                                  /
/                                                                  /
/    TC_INPUT contains inputs stored in the library that  /
/      are input to the translator build step.                /
/    TC_OUTPUT contains outputs that are to be built by     /
/      translator and that will be stored in the library     /
/    PARMS are translator options that are specified          /
/      on the TeamConnection builder definition.                /
/                                                                  /
/  Outputs:                                                       /
/    files generated by the translator                          /
/                                                                  /
```

**Figure 16 (Part 1 of 3). fhbolink.cmd Linkage Edit Build Script**

```
/  Returns:                                                                  /
/    return code from the translator execution                    /
/ ------------------------------------------------------ /

   parse arg parms

  environ   =  'OS2ENVIRONMENT'
  input       =  VALUE('TC_INPUT',,environ)
  output      =  VALUE('TC_OUTPUT',,environ)

   translator       = 'ILINK'      /  specify the translator to be invoked  /
   default_parms = '/NOFREE' /  specify any default parms to be used  /
                                          /  for the compile                                  /
   libraries         = ''              /  specify libraries to be searched          /

  call  invokeTranslator

exit result /  exit with the rc return by the translator  /
/ ------------------------------------------------------ /
/  Routine: invokeTranslator                                          /
/                                                                          /
/  Function:                                                              /
/    This function invokes the translator defined by          /
/    the translator variable.    The default_parms define     /
/    translator options that are used for all translator  /
/    executions. The parm variable contains options           /
/    specified by the user to override the build script     /
/   default_parms.                                                        /
/                                                                          /
/  Variables used:                                                       /
/   translator,  default_parms                                        /
/    input, output, parms location, bomfile                        /
/                                                                          /
/  Returns:                                                                /
/    return code from the translator execution                    /
/ ------------------------------------------------------ /
invokeTranslator:

     /    This build script assumes only multiple inputs and  /
     /    multiple inputs are passed in.                                /

     call parseForFile '.DEF' input
     call parseForFile '.MAP' output
     if mapfile <> '' then
          parms = parms '/MAP'
     translator default_parms parms objfiles ',' exefile ',' mapfile ','
                                                  libraries deffile ';'
return rc
```

**Figure 16 (Part 2 of 3). fhbolink.cmd Linkage Edit Build Script**

```
/ ------------------------------------------------------ /
/  Routine: parseForFile                                 /
/                                                        /
/  Function:                                             /
/    This function parses the string looking for the     /
/    file with the extension passed in.                  /
/    That file is removed from the input string    if    /
/    it is found.                                        /
/    If the type is '.def' (an input) then it sets the   /
/    deffile and objfiles variables.                     /
/                                                        /
/    If the type is '.map' (an output) then it sets the  /
/    mapfile and exefile variables.                      /
/                                                        /
/  Variables used:                                       /
/    inString type deffile objfiles mapfile exefile      /
/                                                        /
/  Variables changed:                                    /
/    deffile objfiles mapfile exefile                    /
/ ------------------------------------------------------ /
parseForFile:

     parse arg type inString

     file = ''

     typepos = pos(type,translate(inString))
     if typepos >   then do
       wordnum    =  words(substr(inString,1,typepos))
       file       =  word(inString,wordnum)
        /  remove the file from the string  /
        outfile    = subword(inString,1,wordnum-1) subword(inString,wordnum+1)
    end
    Else
        outfile = inString

    Select
        when type = '.DEF' then do
           deffile    = ',' file
           objfiles = outfile
       end
        when type = '.MAP' then do
           mapfile = file
           exefile = outfile
       end
       Otherwise  NOP
     end /  select  /
return
```

**Figure 16 (Part 3 of 3). fhbolink.cmd Linkage Edit Build Script**

# APPENDIX B. TEAMCONNECTION FAMILY SETUP

The family is simple so as not to detract from the techniques used for interaction with MVS and SCLM. A TeamConnection client is used on the TeamConnection family server so that only one workstation is required.

Assumptions:

OS/2 is installed on the C: drive.

TeamConnection is installed on the D: drive.

The family setup includes these main activities:

1. create a new family
2. normal administration
   - create components
   - create a release with the no_track process
   - create builders for OS/2 application development
3. initial load activities
   - create builders
   - create a parser
   - create a new work area for the initial load of the parts
   - create a part for a small C "include" source program
   - create a part for a small C source program
   - create parts for object and link
   - create a build tree
   - start TeamConnection build agents and build processors
   - use TeamConnection to BUILD an application
   - integrate the work area into the release
4. create a new work area for use with MVS and SCLM

## ADMINISTRATION

The following steps exercise the administrative TeamConnection functions that are needed to establish a new family and to do the initial creation of parts in a TeamConnection family.

### Preliminary Activities

The following activities must take place before creating the family and parts.

| Action | Description |
|---|---|
| The OS/2 build scripts must be ready. This demonstration assumes the build scripts are located in the directory e:\tcdemo\mvssclm\scripts. | |
| The C source program and "include" must be available on the TeamConnection client. This demonstration assumes the source programs are located in the directory e:\tcdemo\mvssclm\parts. | |
| The following environment variables should be set in **config.sys** in the machines that run the build agents:<br>**TC_FAMILY=mvssclm**<br>**TC_USER=SuprUser**<br>**TC_BECOME=SuprUser**<br>**TC_DBPATH=e:\family\mvssclm** | Do not forget to reboot your machine if you change these values and are relying on them to be set from "config.sys". |
| Create **e:\family** as the path for the family information. | |
| Create **e:\famsave\mvssclm** as a place to save copies of the family. | |

## Create a New Family

This section shows how to create a new family, update the hosts and services files, and start the family.

| Action | Description |
|---|---|
| Make sure the following are running:<br>TCP/IP<br>ObjectStore | |
| Open the TeamConnection group folder | The "TeamConnection Group - Icon View" window appears. |
| Double-click on the TeamConnection Family Administrator icon. | The "TeamConnection Family Administrator" window appears. |
| Select **Family**, then **New**, and then **Default**. | An "Untitled-Settings" notebook appears, with the "Family Information" page displayed. |
| Enter the following:<br>Family Information:<br>**Name** mvssclm<br>**Path** e:\family<br>**Port** 9808 or some other unused port<br>**Mailer** mailexit or null.cmd<br>Initial Superuser:<br>**Login** SuprUser<br>**Name** Super User<br>**Host** FamHost | |
| Once set, you cannot alter these values through TeamConnection administration. | Below the notebook, press the **Create** push button. |
| The "Create Family" window appears and runs the commands to create the family and the super user. This takes about 5 minutes. The return code at the end should be 0. | |
| Press the **OK** push button. | The "TeamConnection Family Administrator" window appears with an icon for mvssclm in the window. |
| From an OS/2 window, enter:<br>`echo %ETC%` | The current value of the ETC environment variable appears. This value is the name of the directory to use in the next few steps. The instructions below assume that the directory is C:\MPTN\ETC. |

| Action | Description |
|---|---|
| Change the directory and drive to C:\MPTN\ETC. | |
| Add mvssclm, bldsock, and tsosock to C:\MPTN\ETC\hosts. Also make sure that there is a carriage return after each line. The hosts listed in Table 3 on page 84 should be in the hosts file. | |
| Add mvssclm, bldsock, and tsosock to C:\MPTN\ETC\services. The services listed in Table 4 on page 84 should be in the services file. | |
| From the "TeamConnection Family Administrator" window, double-click on the mvssclm icon. | The "mvssclm- Family Servers" window appears. |
| Press the **Start** push button in the "Family Server" section of the window. | The family server is started. When the message to "Press CTRL-C to stop" appears, minimize the "mvssclm - Family Servers" window and the "TeamConnection Family Administrator" window. |

Table 3. Hosts File

| IP Address | Host Name | Family / Socket |
|---|---|---|
| 9.37.195.43 | FamHost | mvssclm |
| 9.37.195.43 | FamHost | bldsock |
| 9.67.43.25 | carmvs1.raleigh.ibm.com | tsosock |

Table 4. Services File

| Family / Socket | Port Number | Comment |
|---|---|---|
| mvssclm | 9808/tcp | #port address for mvssclm |
| bldsock | 9997/tcp | #port address for OS/2 build processor |
| tsosock | 32911/tcp | #port address for MVS build processor |

## Start the TeamConnection Client

This section shows how to start the TeamConnection client and change the **Settings**.

| Action | Description |
|---|---|
| On the TeamConnection server machine, create a working directory called E:\WORK\mvssclm | |
| Open the TeamConnection Group folder. | |
| Right mouse click on the TeamConnection client icon. | A list of options appears. |
| Click on **Settings**. | The "TeamConnection Client - Settings" notebook appears. |
| On the "Program" page, change the working directory to E:\WORK\mvssclm | |
| Close the "TeamConnection Client - Settings" notebook. | |
| Double-click on the TeamConnection client icon. | The "TeamConnection - Tasks" window appears. |
| Select **Windows** and then **Settings**. | The "Settings" notebook appears, with the "Environment" page showing. |
| Select the **GUI** tab in the "Settings" notebook.<br>Be sure that the **Verbose commands** is not checked. | "Verbose commands" checked on will result in additional information windows in which you must press the "OK" push button. |
| Select the **Environment** tab in the "Settings" notebook.<br>Enter the following on the "Environment" page:<br>**Family** mvssclm<br>**Release** release1<br>**Become user** SuprUser<br>**User ID** SuprUser<br>**Relative directory** e:\mytemp<br>Then press the **Apply** push button. | |
| Press the **Test Connection to Server** push button. | The "Settings" information window should appear and indicate that you connected to the server. There should be no error messages. |
| Press **OK** to close the information window | |

| Action | Description |
| --- | --- |
| Select the **Pool** tab in the "Settings" notebook.<br>Enter **pool1** for the **Pool** value.<br>Then press the **Apply** push button. | All build activity in the basic demonstration uses "pool1". This value is the default and is automatically entered into the "Pool" field of the build windows. |
| Select the **GUI** tab in the "Settings" notebook.<br>Make sure that the check box for **Verbose commands** is deselected.<br>Press the **Apply** push button. | Deselecting "Verbose commands" reduces the number of messages you receive. |
| Double-click on the upper left corner of the "Settings" window. | The "TeamConnection - Tasks" window appears. The user ID and mvssclm should be in the lower left corner of the tasks window. |

## Create Components

This section creates the component structure for the demonstration. The demonstration uses the overall controlling component projname. The other components can be used for modifications of the demonstration to show activities related to components.

| Action | Description |
|---|---|
| Select **Actions**, then **Components**, and then **Create**. | The "Create Components" window appears. |
| Enter the components shown in Table 5, pressing the **Apply** push button after each entry. | |
| Press the **Cancel** push button. | The "TeamConnection - Tasks" window appears. |

Table 5. Components

| Components | Parent | Process | Owner | Description |
|---|---|---|---|---|
| projname | root | prototype | SuprUser | overall controlling component |
| subsys1 | projname | prototype | SuprUser | subsystem 1 |
| subsys2 | projname | prototype | SuprUser | subsystem 2 |

## Create a Release

This section shows how to create a release. Parts, work areas, and builders cannot be created without first specifying a release.

| Action | Description |
|---|---|
| Select **Actions**, then **Releases**, and then **Create**. | The "Create Releases" window appears. |
| Enter the following information:<br>**Releases** release1<br>**Component** projname<br>**Process** no_track<br>**Owner** SuprUser<br>**Description** projname release 1<br>Also select **Automatic version pruning**.<br>Press the **OK** push button. | The "TeamConnection - Tasks" window appears. The release has been created. |

## Create Builders

This section creates the os2 builders that are used throughout the demonstration. All os2 builders should be created at this time so that you do not need to return to this activity later.

The tso builder for use in “TeamConnection to SCLM - BUILD” on page 39 is created in the example for that section.

| Action | Description |
|---|---|
| Select **Actions**, then **Builders**, and then **Create** on client workstation. | The "Create Builders" window appears. |
| Enter the builder information shown in Figure 17 on page 89, pressing the **Apply** push button after each entry. | A NULL builder is required for a collector part later in the demonstration. |
| Press the **Cancel** push button to return to the Tasks list. | The "TeamConnection - Tasks" window appears. |
| Select **Objects** and then **Builders**. | The "Builder Filter" window appears. |
| In the "History" section, select **Show all Builders**. | |
| Press the **Save to Task List** push button. | The "Edit Task List" window appears. |
| Enter **builders** in the description field and press the **Add/Change** push button. | An icon for the builders is added to the "TeamConnection - Tasks" window. |
| Press the **Cancel** push button to return to the Tasks list. | The "TeamConnection - Tasks" window appears. |
| Double-click on the **builders** icon to run the report. | The "TeamConnection - Builders" window appears. |
| Close the "TeamConnection - Builders" window. | The "TeamConnection - Tasks" window appears. |

```
Builder          Ccompile
Release          release1
Script           compC.cmd
Environment      os2
Comp. oper.      <=
RC value         4
File type        text
Source file      e:\tcdemo\mvssclm\scripts\fhbocomp.cmd
Parameters       /C+ /O+
Timeout          1

Builder          Clink
Release          release1
Script           linkC.cmd
Environment      os2
Comp. oper.      <=
RC value         4
File type        text
Source file      e:\tcdemo\mvssclm\scripts\fhbolink.cmd
Parameters
Timeout          1

Builder          os2null
Release          release1
Script           NULL
Environment      os2
Comp. oper.      ==
RC value
File type        none
Source file
Parameters
Timeout          1
```

**Figure 17. OS2 builder information for mvssclm family**

## Create Parsers

This section creates parsers. The term "create a parser" involves telling TeamConnection which command to run for parsing a part. Programs that actually get control to do the parsing must be written just like any other tool.

| Action | Description |
|---|---|
| Select **Actions**, then **Parsers**, and then **Create**. | The "Create Parser" window appears. |
| Set the following fields:<br>**Parser** see Table 6<br>**Release** release1<br>**Command** see Table 6<br>Enter the parser name and command shown in Table 6, pressing the **Apply** push button after each entry. | |
| Press the **Cancel** to return to the Tasks list. | The "TeamConnection - Tasks" window appears. |

Table 6. release1 Parser Create Information

| Parser | Release | Command |
|---|---|---|
| Cparser | release1 | fhbopars.cmd |

## Optional - Save the Family Database

Saving the family database allows you to restart from this point in the demo. Save the family by doing the following steps.

| Action | Description |
|---|---|
| From the TeamConnection Family Server, in the "mvssclm - Family Servers" window, press the **Stop** push button for the family server. Also press the **Stop** push button for the notification server if it has been started. | The family and notification servers are stopped. |
| From an OS/2 window, enter the following:<br>`cd e:\famsave\mvssclm`<br>`copy e:\family\mvssclm\mvssclm.tcd mvssclm1.tcd` | You might have to stop the build agents and/or build processors to be able to copy the family database. |
| From the TeamConnection Family Server, in the "mvssclm - Family Servers" window, press the **Start** push button for the family server. Also press the **Start** push button for the notification server if you want it to be started. | The family server and, optionally, the notification server are started.<br>If you had to stop any build agents and/or build processors, restart them now. |

## Open a Feature and Create a Work Area

This section creates a work area for the initial load of the parts. This would be a typical activity for migrating existing parts (source code, tables) into TeamConnection. A feature is not needed with the no_track release process. However, it is a good practice to open a feature when initializing a family.

| Action | Description |
|---|---|
| Select **Actions**, then **Features**, and then **Open**. | The "Open Feature" window appears. |
| Enter the following information:<br>**Component** projname<br>**Remarks** initial load of the family<br>**Name** initialize<br>**Prefix** f<br>Press the **OK** push button. | The "Open Feature" information window appears with a message that the new feature was opened successfully. |
| Press the **OK** push button again to return to the "TeamConnection - Tasks" window. | |
| Select **Actions**, then **Features**, and then **Accept**. | The "Accept Feature" window appears. |
| Enter the following information:<br>**Features** initialize<br>**Remarks** accepting initial load of the family<br>Press the **OK** push button. | The "TeamConnection - Tasks" window appears. The feature has been accepted. |
| Select **Actions**, then **Work areas**, and then **Create**. | The "Create Work Areas" window appears. |
| Enter the following information:<br>**Work areas** initwa<br>**Releases** release1<br>**Owner** SuprUser<br>Press the **OK** push button. | The "TeamConnection - Tasks" window appears. The work area has been created. |

## Create Parts

This section creates the parts for the initial load of the database as well as the output parts that will be created from builds.

| Action | Description |
|---|---|
| Select **Actions**, then **Parts**, and then **Create**. | The "Create Parts" window appears. |
| Set the following fields for each part:<br>**Part names** see Table 7<br>**Release** release1<br>**Work area** initwa<br>**Component** projname<br>**File type** see Table 7<br>**Source** see Table 7<br>**Source file** see Table 7<br>**Builder** see Table 7<br>**Parser** see Table 7<br>Enter the parts shown in Table 7, pressing the **Apply** push button after each entry. | NOTE: Builders and parsers do not need to be assigned when a part is created. They may be created and changed at any time. Assignment to a part can be done using Actions -> Parts -> Modify -> Properties. |
| Press the **Cancel** push button to return to the Tasks list. | The "TeamConnection - Tasks" window appears. |

Table 7. projname Create Part Information

| Part name | File type | Source | Source file | Builder | Parser |
|---|---|---|---|---|---|
| testcopy.h | Text | Copy from | e:\tcdemo\mvssclm \parts\testcopy.h | | Cparser |
| testpart.c | Text | Copy from | e:\tcdemo\mvssclm \parts\testpart.c | | Cparser |
| testpart.obj | Binary | No source | | Ccompile | |
| testpart.exe | Binary | No source | | Clink | |

## Create Build Tree

This section shows how to create a build tree. A build tree shows the configuration of the application. TeamConnection uses the build tree to determine what parts need to be built.

| Action | Description |
|---|---|
| Select **Objects**, then **Parts**, and then **Parts**. | The "Part Filter" window appears. |
| Set the following fields:<br>**Release** release1<br>**Work area** initwa<br>In the "History" section, select **Show all Parts**.<br>Press the **OK** push button. | The "TeamConnection - Parts" window appears. |
| For each icon in Table 8 on page 94:<br>click on the **icon** in the "TeamConnection - Parts" window<br>click on the second mouse button<br>click on **Connect** from the pop up<br>Set the following fields and radio buttons in the "Connect Parts" window:<br>**Parent** see Table 8 on page 94<br>**Relation to parent** see Table 8 on page 94<br>Press the **OK** push button. | The fields "Path names", "Type", "Release", and "Work area" should already have the correct values. |
| Close the "TeamConnection - Parts" window. | The "TeamConnection - Tasks" window appears. |
| Select **Objects**, then **Parts**, and then **BuildView**. | The "BuildView Filter" window appears. |
| Set the following fields:<br>**Release** release1<br>**Work area** initwa<br>In the "History" section, select **Show all BuildView**.<br>Press the **OK** push button. | The "TeamConnection - BuildView" window appears. |
| Expand the **testpart.exe** icon to show the entire tree. | Examine the tree for proper structure (see Figure 18 on page 94). |

| Table 8. release1 Build Tree Information | | |
|---|---|---|
| **Icon** | **Parent** | **Relation** |
| testpart.c | testpart.obj | input |
| testpart.obj | testpart.exe | input |

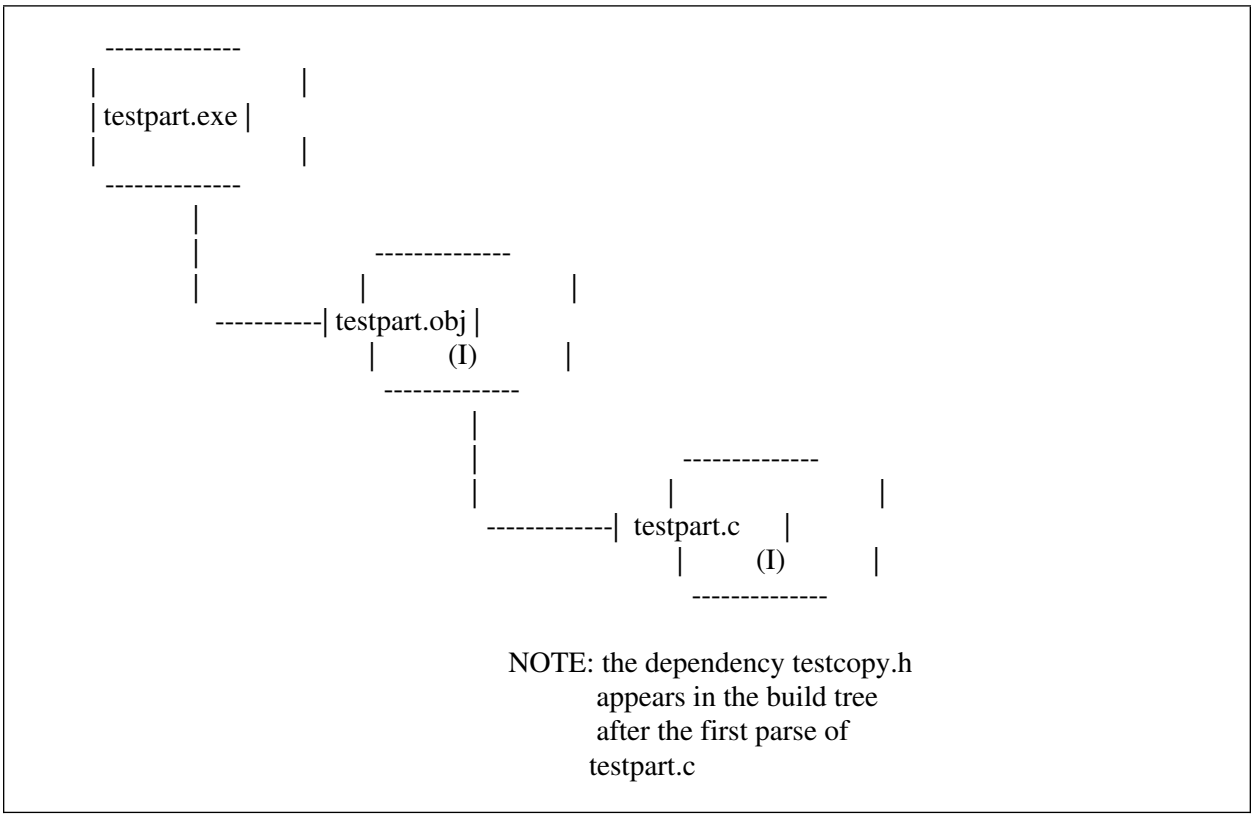



**Figure 18. Build Tree for testpart.exe**

## Start the OS/2 Build Agent and Build Processor

This section shows how to start an OS/2 build agent and build processor. Together this pair is called a build server. Build events wait in the job queue for a build agent to start.

| Action | Description |
|---|---|
| Go to an OS/2 window (recommended on the machine that is running the TeamConnection family server). | |
| Make sure the TC_DBPATH environment variable is set to the proper value (e:\family\mvssclm for example).<br><br>You must also have PATH set to e:\family\mvssclm if that is where teamagnt.exe and teamproc.exe are located. Otherwise you could get a SYS1041 error from OS/2. | |
| From an OS/2 window issue the command<br><br>**teamagnt -f mvssclm -e os2 -p pool1 -s bldsock** | The build agent starts in the OS/2 window. |
| Minimize the OS/2 window - DO NOT CLOSE. | |
| From an OS/2 window issue the command<br><br>**teamproc -s bldsock** | The build processor starts in the OS/2 window. |
| Minimize the OS/2 window - DO NOT CLOSE. | |

## Build the OS/2 Application

This section builds the OS/2 application. **Build** puts build events on the job queue and the build agent removes them. The build agent sends the build script (from the builder) and the inputs to the build processor. The build processor runs the build script.

| Action | Description |
|---|---|
| From the "TeamConnection - BuildView" window, select **testpart.exe**. | |
| Select **right mouse** button, and then select **Build**. | The "Build Parts" window appears. |
| Set the following fields and radio buttons:<br>**Path name** testpart.exe<br>**Type** TCPart<br>**Release** release1<br>**Work area** initwa<br>**Pool** pool1<br>**Build mode** normal<br>**Parameters**<br>**Detail file**<br>Press the **OK** push button. | |
| Monitor the "Build Progress" window, press the **OK** push button when the build is complete, and close the "TeamConnection - BuildView" window. | The "TeamConnection - BuildView" window appears. |
| Press the **F5** function key to refresh the "TeamConnection - BuildView" window. | The "TeamConnection - BuildView" window reappears.<br>If the build was successful, testpart.exe should have a green checkmark. |
| Close the "TeamConnection - BuildView" window. | The "TeamConnection - Tasks" window appears. |

## Test the OS/2 Application

This section extracts the executable part **testpart.exe** and runs it for the test.

| Action | Description |
|---|---|
| Select **Actions**, then **Parts**, and then **Extract**. | The "Extract Parts" window appears. |
| Set the following fields:<br>**Path names** testpart.exe<br>**Release** release1<br>**Work area** initwa<br>**Destination directory** e:\mytemp<br>Press the **OK** push button. | |
| Go to an OS/2 window.<br>Run the command **e:\mytemp\testpart**.<br>Minimize or close the OS/2 window. | The messages from testpart.exe should appear:<br>I am testpart.c<br>I was built by the Team Connection integrated Build Tool!!! |

## Integrate the Work Area to the Release

This section integrates the parts in the work area with parts in the release (there are no parts in the release for the initial load).

| Action | Description |
|---|---|
| Select **Actions**, then **Work Areas**, and then **Integrate**. | The "Integrate Work Areas" window appears. |
| Enter the following:<br>**Work areas** initwa<br>**Releases** release1<br>Press the **OK** push button. | The "TeamConnection - Tasks" window appears. |

## Create a Work Area for Application Development

This section creates a work area for application development work. A defect or feature is not needed with the no_track release process.

| Action | Description |
|---|---|
| Select **Actions**, then **Work areas**, and then **Create**. | The "Create Work Areas" window appears. |
| Enter the following information:<br>**Work areas** mvstcwa<br>**Defects/Features**<br>**Releases** release1<br>**Owner** SuprUser<br>Press the **OK** push button. | The "TeamConnection - Tasks" window appears. The work area has been created. |

## Optional - Save the Family Database

Saving the family database at this point allows you to restart the demonstration from this point. Save the family by doing the following steps.

| Action | Description |
|---|---|
| From the TeamConnection Family Server, in the "mvssclm - Family Servers" window, press the **Stop** push button for the family server. Also press the **Stop** push button for the notification server if it has been started. | The family and notification servers are stopped. |
| From an OS/2 window, enter the following:<br>`cd e:\famsave\mvssclm`<br>`copy e:\family\mvssclm\mvssclm.tcd mvssclm2.tcd` | You might have to stop the build agents and/or build processors to be able to copy the family database. |
| From the TeamConnection Family Server, in the "mvssclm - Family Servers" window, press the **Start** push button for the family server. Also press the **Start** push button for the notification server if you want it to be started. | The family server and, optionally, the notification server are started.<br>If you had to stop any build agents and/or build processors, restart them now. |

# APPENDIX C. ISPF C/S WORKSTATION INSTALLATION USING TCP/IP

The connection between MVS and the TeamConnection client or family server machine is through the ISPF 4.2 Client/Server function. The ISPF Client/Server software must be installed on the workstation in order for MVS to communicate with the workstation. The following instructions will help you install the ISPF Client/Server software using TCP/IP.

## WORKSTATION PREPARATIONS

| Action | Description |
|---|---|
| Make a directory called **e:\ispfcs**. | This directory will contain the ISPF Client/Server files. |
| Verify that you are able to ftp (transfer files using the file transfer program) to the OS/2 machine.<br>1. echo %etc% to see where the TCP/IP information is located<br>2. go to that directory<br>3. find, or create, the **TRUSERS** file | The TRUSERS file is of the form:<br>user: userid password<br>rd: e:\ispfcs<br>wr: e:\ispfcs<br>"userid" and "password" have nothing to do with other userids and passwords in TeamConnection They are unique to the TRUSERS file.<br>"rd" identifies directories for read access and "wr" identifies directories for write access.<br>When you are finished with the ISPF Client/Server installation you can delete the TRUSERS file to prevent other ftp activity to your workstation. |
| Go to an OS/2 window and start ftp by entering **ftpd**. | |

## MVS

| Action | Description |
|---|---|
| Logon to TSO and go into ISPF. | The "ISPF Primary Options Menu" appears. |
| Select **Utilities** from the "ISPF Primary Options Menu". This is usually option 3 on the menu. | The "Utility Selection Panel" appears. |

| Action | Description |
| --- | --- |
| Select **ISPF C/S** from the "Utility Selection Panel". This is usually option 7 on the menu.<br><br>Press the **Enter** key. | The "ISPF Client/Server Component" box appears. |
| Select the proper "Operating System - Network" for the ISPF Client Server machine. (For this example assume **OS/2 - TCP/IP** is selected.) | The "ISPF Client/Server Component Install with TCP/IP on OS/2" box appears. Note that this box will tell you that the following products must be installed and/or running:<br><br>OS/2 - Operating System/2<br><br>Transmission Control Protocol/Internet Protocol<br><br>File Transfer Protocol<br><br>File Transfer Protocol Server |
| Press the **Enter** key. | |
| Enter the following in the box that appears for information regarding the selection of **OS/2 - TCP/IP**:<br><br>**IP Address or Host Name to OPEN:** 9.37.195.43<br>**User to login: (Case sensitive)**: userid<br>**Password for login: (Case sensitive)** password<br>**Directory to copy file to:** e:\ispfcs<br>**Data Set to copy file from:** 'SYS1.ISP.SISPGUI'<br>**Language** English<br><br>Under **Enter "/" to select option** put / beside:<br>**Browse FTP output only if an error occurs.** | "userid" and "password" are those values found in the TRUSERS file on the workstation.<br><br>The product data set 'SYS1.ISP.SISPGUI' might have a different name on your MVS system.<br><br>"English" and "Browse" options are selected for this example. |
| Press the **Enter** key.<br><br>You will see the messages indicating the commands were successful and the file transfer is complete. | The "e:\ispfcs" directory on the workstation will contain the ISPF Client/Server self extracting ".exe" file. |
| Run **ispfinst** from the e:\ispfcs directory.<br><br>Press the **Install** key.<br><br>Press the **No** key for the DBCS translation tables.<br><br>Press the **OK** key for adding e:\ispfcs to LIBPATH in config.sys.<br><br>Press the **Quit** key. | The ISPF Client/Server files are now in e:\ispfcs. |

| Action | Description |
| --- | --- |
| Start the ISPF C/S on the workstation by entering **ispfcs** from the "e:\ispfcs" directory. | The "ISPF C/S" window appears on the workstation.<br>Do not close the window. |

# APPENDIX D. SCLM PROJECT DEFINITION SETUP

Some knowledge of SCLM will be helpful in running the examples. There should be sufficient detail in these instructions for the novice. See Appendix A, “TeamConnection Parts and Scripts” on page 69 for information about obtaining the files, language definitions, and project definition used in these examples.

The SCLM setup involves:

1. creating a USERINFO file for TeamConnection command language definitions
2. creating an ACTINFO file for TeamConnection command language definitions
3. creating TeamConnection command language definitions
4. creating a project definition that includes TeamConnection command language definitions
5. assembling and linking the project definition
6. allocating the data sets controlled by the SCLM project definition
7. creating the VSAM cluster for the SCLM accounting records

The following detailed instructions are for establishing a new SCLM project definition for demonstrating a TeamConnection BUILD as part of an SCLM BUILD. This project definition is very simple so that the focus is on the TeamConnection BUILD.

An experienced SCLM user could modify existing SCLM project definitions and the related USERINFO and ACTINFO files. This would provide a more complex example of a coordinated SCLM and TeamConnection BUILD and would be typical of what an existing SCLM project definition would need in order to incorporate a TeamConnection BUILD.

## CREATE USERINFO AND ACTINFO FILES

For demonstration purposes it is usually easiest to use your own TSO user id as the high level qualifier for these data sets.

| Action | Description |
|---|---|
| Allocate the sequential data set **'TSOUSER.SCLM.USERINFO'**.<br><br>Use RECFM=FB and 80 byte records.<br><br>Copy the USERINFO file in Figure 20 on page 109 with appropriate changes for the directories to be used on your workstation. | This data set name is consistent with other documentation. Be sure the data set name in the language definitions matches the one you use.<br><br>See Appendix A, “TeamConnection Parts and Scripts” on page 69 for information on a softcopy of this file. |

| Action | Description |
|---|---|
| Allocate the sequential data set **'TSOUSER.PROJDEFS.ACTINFO'**.<br><br>Use RECFM=FB and 80 byte records.<br><br>Copy the ACTINFO file in Figure 21 on page 110 with appropriate changes for the directories to be used on your workstation. | This data set name is consistent with other documentation. Be sure the data set name in the language definitions matches the one you use.<br><br>See Appendix A, "TeamConnection Parts and Scripts" on page 69 for information on a softcopy of this file. |

## CREATE SCLM LANGUAGE DEFINITIONS AND PROJECT DEFINITION DATA SETS

For demonstration purposes it is usually easiest to use your own TSO user id as the high level qualifier of the project definition data sets.

| Action | Description |
|---|---|
| Allocate the partitioned data set **'TSOUSER.PROJDEFS.SOURCE'**.<br><br>Use RECFM=FB, 80 byte records, and 10 directory blocks. | This data set will contain the SCLM language definitions and project definition. |
| Allocate the partitioned data set **'TSOUSER.PROJDEFS.OBJ'**.<br><br>Use RECFM=FB, 80 byte records, and 10 directory blocks. | This data set will contain the SCLM project definition object code. |
| Allocate the partitioned data set **'TSOUSER.PROJDEFS.LOAD'**.<br><br>Use RECFM=U, block size of 6144, and 10 directory blocks. | This data set will contain the SCLM project definition load module. The block size may be set to a value typically in use on your MVS system. |
| Create the following members in **'TSOUSER.PROJDEFS.SOURCE'** (make sure that references to "TSOUSER" in these members are changed to the high level data set qualifier of your choice):<br><br>**DUMMY** (see Figure 22 on page 114)<br><br>**CHECKOUT** (see Figure 23 on page 115)<br><br>**CHECKIN** (see Figure 24 on page 119)<br><br>**TCBUILD** (see Figure 25 on page 122) | See Appendix A, "TeamConnection Parts and Scripts" on page 69 for information on softcopies of these files.<br><br>Be sure that the USERINFO and ACTINFO data set names in the language definitions match the names you are using. |

## CREATE THE SCLM PROJECT DEFINITION MEMBER

| Action | Description |
|---|---|
| Create member **SCLMTC** (see Figure 26 on page 125) in **'TSOUSER.PROJDEFS.SOURCE'**. | See Appendix A, “TeamConnection Parts and Scripts” on page 69 for information on a softcopy of this member. |
| Edit **'TSOUSER.PROJDEFS.SOURCE(SCLMTC)'** to change all occurrences of **TSOUSER** to your user id. | |

## ASSEMBLE AND LINK THE PROJECT DEFINITION

| Action | Description |
|---|---|
| Select **Foreground** from the "ISPF Primary Option Menu". This is usually option 4. | The "Foreground Selection Panel" appears. |
| Select **High Level Asm/Asm H** from the "Foreground Selection Panel".<br><br>This is usually option 1. | The "Foreground High Level Assembler / Assembler H" panel appears. |
| Enter the following on the "Foreground High level Assembler / Assembler H" panel:<br><br>**Project** TSOUSER<br>**Group** PROJDEFS<br>**Type** SOURCE<br>**Member** SCLMTC<br>**Assembler Level** HASM<br>**Additional input Libraries** 'SYS1.ISP.SISPMACS'<br><br>Press the **Enter** key. | The listing from the assembly of the project definition will appear.<br><br>The project definition could also be assembled and linked from a batch job.<br><br>The data set 'SYS1.ISP.SISPMACS' might have a different name on your MVS system. This is the data set that contains the SCLM project definition macros. |
| Go to the bottom of the assembler listing and look for the message "**NO STATEMENTS FLAGGED IN THIS ASSEMBLY**". | Correct all assembly errors before proceeding to the next step. |
| Return to the "Foreground Selection Panel". | |
| Select **Binder/Link editor** from the "Foreground Selection Panel". This is usually option 7. | The "Foreground Binder/Linkage Editor" panel appears. |

| Action | Description |
|---|---|
| Enter the following on the "Foreground Binder/Linkage Editor" panel:<br>**Project** TSOUSER<br>**Group** PROJDEFS<br>**Type** OBJ<br>**Member** SCLMTC<br>**Binder** BINDER<br>**Other** LET,LIST,MAP<br>Press the **Enter** key. | The listing from the linkage editor will appear.<br>The project definition could also be linked from a batch job.<br>The project definition will be the member SCLMTC in the data set 'TSOUSER.PROJDEFS.LOAD'. |
| Look for the message **"IEW2008I PROCESSING COMPLETED. RETURN CODE = 0."** in the linkage editor listing. | Correct all linkage errors before proceeding. |

## ALLOCATE THE DATA SETS CONTROLLED BY THE SCLM PROJECT DEFINITION

For demonstration purposes it is usually easiest to use your own TSO user id as the high level qualifier.

| Action | Description |
|---|---|
| Allocate these partitioned data sets with RECFM=FB and record length of 80.<br>**'TSOUSER.DEVELOP.ARCHDEF'**<br>**'TSOUSER.DEVELOP.SOURCE'**<br>**'TSOUSER.DEVELOP.SRC'**<br>**'TSOUSER.DEVELOP.BIN'** | 'TSOUSER.DEVELOP.BIN' is not used in the demonstration but might be useful for extending the demonstration to include binary parts. |

## CREATE THE VSAM CLUSTER FOR THE SCLM ACCOUNTING FILE

For demonstration purposes it is usually easiest to use your own TSO user id as the high level qualifier of the VSAM cluster.

| Action | Description |
|---|---|
| Modify the job in Figure 19 on page 108 to have the proper VOL value and the proper cluster name for your MVS system. | |

| Action | Description |
| --- | --- |
| Submit the job to first delete the cluster and then create the cluster. | Note: The first time this job runs it will end with a nonzero return code if the cluster does not exist. This job is useful for creating a fresh SCLM account file for the project definitions that use it.<br><br>See Appendix A, “TeamConnection Parts and Scripts” on page 69 for information on a softcopy of this job. |
| The SCLM project definition **SCLMTC** is now ready for use. | |

```
//             YOUR JOB CARD
//
//  MODIFICATIONS - USE YOUR JOB CARD
//                          USE YOUR CLUSTER NAME FOR TSOUSER.ACCOUNT.FILE
//                          USE YOUR VOL VALUE
//                          USE YOUR INDEX NAME FOR TSOUSER.ACCOUNT.FILE.INDEX
//                          USE YOUR DATA    NAME FOR TSOUSER.ACCOUNT.FILE.DATA
//
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=
//SYSIN DD
      DELETE 'TSOUSER.ACCOUNT.FILE' CLUSTER
      DEFINE CLUSTER (NAME('TSOUSER.ACCOUNT.FILE') -
                       CYL(1 1) -
                      VOL(XXXXXX)  -
                       KEYS(26  ) -
                       RECORDSIZE(264 32   ) -
                      SHAREOPTIONS(4,3)  -
                       IMBED SPEED UNIQUE SPANNED) -
               INDEX   (NAME('TSOUSER.ACCOUNT.FILE.INDEX'))  -
               DATA     (NAME('TSOUSER.ACCOUNT.FILE.DATA')  -
                       CISZ(2 24)  -
                       FREESPACE(5   5 ))
/
//STEP2 EXEC PGM=IDCAMS
//INPUT DD
                        VSAM DATABASE INITIALIZATION RECORD
/
//SYSPRINT DD SYSOUT=
//SYSIN      DD
    REPRO INFILE(INPUT) OUTDATASET('TSOUSER.ACCOUNT.FILE')
/
//
```

**Figure 19. vsamallc.jcl SCLM VSAM account file allocation**

```
  Directory info - DATA_DIR and WSDIR must point to the same directory

DATA_DIR          e:\mytemp
WSDIR             e:\mytemp

  Response file - not used in this Technical Report

RESPONSE_FILE   e:\tcdemo2\mvssclm\response.fil

  Mode. Workstation build default is minimized. This will override any
  value set in the ACTINFO file.
  This is the mode for the workstation command. Valid values are:
  MAXimized, MINimized, VISible and INVISible. See the WSCMDV section
  of the ISPF SELECT service for additional information.

mode     max

  Del_cmd. Default is 'del'.
  This keyword specifies the delete command to be used to remove the
  message file(s) from previous builds or build steps from the
  workstation before executing the workstation command.

del_cmd erase
```

**Figure 20. userinfo.txt SCLM TeamConnection USERINFO file**

ACTINFO FILE FOR TEAMCONNECTION COMMANDS

GENERAL NOTES

THIS FILE IS AN EXAMPLE THAT CAN SERVE AS A REFERENCE IN THE CONSTRUCTION AND CUSTOMIZATION OF ACTIONS FOR A PARTICULAR WORKSTATION TOOL AND ENVIRONMENT.

CUSTOMIZATION NOTES

THIS IS AN EXAMPLE OF AN ACTINFO FILE THAT MIGHT BE USED WITH TEAMCONNECTION. IT CONTAINS SAMPLE ACTIONS TO INVOKE TEAMCONNECTION COMMANDS.

MODIFY THE TYPE INFORMATION AT THE TOP OF THIS FILE AND THE INFORMATION FOR EACH ACTION TO MATCH THE CONVENTIONS USED IN YOUR PROJECT AND ON YOUR WORKSTATION.

CPARM PARAMETERS WILL APPEAR ON THE WORKSTATION COMMAND LINE. RPARM PARAMETERS WILL BE WRITTEN TO A RESPONSE FILE. THE DEFAULT MESSAGE_ FILE IS C:\SCLM.MSG.

NLS CONSIDERATIONS

THIS FILE WAS WRITTEN FOR USE WITH THE ENGLISH CODE PAGE. THE CHARACTERS IN THIS FILE MAY NOT HAVE THE SAME HEX VALUE IN ALL LANGUAGES. WHEN USING THIS FILE WITH A CODE PAGE OTHER THAN ENGLISH, UPDATE THE CHARACTERS IN THIS FILE TO MATCH THE CODE PAGE IN USE. AS AN EXAMPLE, THE BACKSLASH CHARARCTER IS X'E ' IN THE ENGLISH CODE PAGE. THIS VALUE WOULD HAVE TO BE CHANGED TO X'EC' FOR GERMAN OR X'B2' FOR KANJI. THE AT SIGN CHARACTER WOULD HAVE TO BE CHANGED FROM THE ENGLISH X'7C' TO 'B5' FOR GERMAN. THE CHARACTERS REQUIRING TRANSLATION WILL DEPEND ON THE CODE PAGE IN USE.

SCLM type to workstation file name mapping

The following statements define the mapping between SCLM type names and workstation file extensions as well as the transfer format for the data. The statements are processed in order. If the member being processed does not match the first TYPE criteria, then it will be compared to the next TYPE criteria and so on until a match is found.

**Figure 21 (Part 1 of 3). actinfo.txt SCLM TeamConnection ACTINFO file**

```
  Types containing BINARY data (numbers are referenced in the text and
                                       are not part of the statements)

  This mapping indicates that the host SCLM type BIN will be the
  workstation file extensions of BIN. The BIN in the type name will
  be used to indicate that members in this type contains binary data.
  As an example, the workstation file 'member.BIN' will be transferred
  as binary (no ASCII-to-EBCIDIC conversion) and stored in the SCLM
  library 'BIN(member)'.

TYPE                BIN                                                (1)
EXTENSION           .BIN                                               (1)
TRANSFER_FORMAT BINARY                                                    (2)
SUBDIRECTORY      \                                                    (3)


  Types containing ASCII data (numbers are referenced in the text and
                                       are not part of the statements)

  This mapping indicates that the host SCLM type SRC will be the
  workstation file extension of C. As an example, the workstation file
  'member.C' will be stored in the SCLM library 'SRC(member)'.

TYPE                SRC                                                (1)
EXTENSION           .C                                                 (1)
TRANSFER_FORMAT ASCII                                                     (2)
SUBDIRECTORY      \                                                    (3)

  Types containing ASCII data (numbers are referenced in the text and
                                       are not part of the statements)

  This mapping indicates that any members whose type does not match
  the previous criteria (TYPE   will always match), will be processed
  as ascii/text files.    The workstation file extension will be the
  same as the SCLM type.    As an example, a member in the CPP type in
  SCLM will have the workstation file name 'member.CPP'.

TYPE                                                                  (1)
EXTENSION           .                                                 (1)
TRANSFER_FORMAT ASCII                                                     (2)
SUBDIRECTORY      \                                                    (3)
```

**Figure 21 (Part 2 of 3). actinfo.txt SCLM TeamConnection ACTINFO file**

Workstation Commands (numbers are referenced in the text and are not part of the statements)

The following statements define the actions/commands to be run on the workstation.

```
 checkout

ACTION      CHECKOUT                                          (4)
COMMAND       teamc
CPARM       part
CPARM       -checkout
CPARM       testpart.c
CPARM       -release
CPARM       release1
CPARM       -workarea
CPARM       mvstcwa

 checkin

ACTION      CHECKIN                                           (4)
COMMAND       teamc
CPARM       part
CPARM       -checkin
CPARM       testpart.c
CPARM       -release
CPARM       release1
CPARM       -workarea
CPARM       mvstcwa

 build

ACTION      BUILD                                             (4)
COMMAND       teamc
CPARM       part
CPARM       -build
CPARM       testpart.exe
CPARM       -type
CPARM       TCPart
CPARM       -release
CPARM       release1
CPARM       -workarea
CPARM       mvstcwa
CPARM       -pool
CPARM       pool1
CPARM       -normal
```

**Figure 21 (Part 3 of 3). actinfo.txt SCLM TeamConnection ACTINFO file**

```
         SCLM LANGUAGE DEFINITION FOR TEAMCONNECTION DEMONSTRATION
                          DUMMY  LANGUAGE

         THIS LANGUAGE IS USEFUL FOR SIMULATING COMPILES, LINKS,
         AND OTHER ACTIVITIES FOR MEMBERS THAT ARE NOT INVOLVED IN
         TEAMCONNECTION COMMANDS. A MORE SOPHISTICATED EXERCISE WOULD
         INVOLVE THE ACTUAL USE OF COMPILES, LINKERS, AND OTHER MVS
        TRANSLATORS.

DUMMY    FLMLANGL    LANG=DUMMY,                                          C
                VERSION=1,CHKSYSLB=IGNORE

 EXECUTE TEAMCONNECTION COMMAND - DUMMY

          FLMTRNSL    CALLNAM='TC  DUMMY',                                   C
                FUNCTN=BUILD,                                             C
                CALLMETH=ATTACH,                                            C
                COMPILE=IEFBR14,                                          C
                VERSION=1,                                                C
                GOODRC= ,                                                 C
                PORDER=
```

**Figure 22. dummy.lng SCLM DUMMY Language Definition**

```
        SCLM LANGUAGE DEFINITION FOR TEAMCONNECTION CHECKOUT

    THIS LANGUAGE DEFINITION IS AN EXAMPLE THAT CAN SERVE AS A
    REFERENCE IN THE CONSTRUCTION AND CUSTOMIZATION OF LANGUAGE
    DEFINITIONS FOR EXECUTING TEAMCONNECTION COMMANDS.

    THE NUMBERS IN PARENTHESIS ARE USED FOR REFERENCE IN THE EXPLANATION
    THAT FOLLOWS AND ARE NOT PART OF THE LANGUAGE DEFINITION

CHECKOUT FLMLANGL       LANG=CHECKOUT,                                  (1)      C
                VERSION=1,CHKSYSLB=IGNORE

 EXECUTE TEAMCONNECTION COMMAND - CHECKOUT
       ACTION is defined in TSOUSER.PROJDEFS.ACTINFO
          FLMTRNSL     CALLNAM='TC  CHECKOUT',                         (2)      C
                FUNCTN=BUILD,                                                   C
                CALLMETH=ISPLNK,                                                C
                COMPILE=SELECT,                                                 C
                VERSION=1,                                                      C
                GOODRC= ,                                                       C
                PORDER=1,                                                       C
                OPTIONS='CMD(FLMLTWST  ACTION=CHECKOUT,BMAPINFO=@@FLM$MP,C
                SCLMINFO=@@FLMINF,BLDINFO=@@FLMBIO,PARMS='
             ( OUT1 )
             FLMALLOC    IOTYPE=O,RECFM=VB,LRECL=256,                            C
                RECNUM=4   ,DDNAME=OUT1,CATLG=Y,                                C
                KEYREF=OUT1,DFLTTYP=SRC
             ( USERINFO )
             FLMALLOC    IOTYPE=A,DDNAME=USERINFO                        (3)
               FLMCPYLB  @@FLMUID.SCLM.USERINFO
             ( ACTINFO )
             FLMALLOC    IOTYPE=A,DDNAME=ACTINFO                         (4)
               FLMCPYLB  @@FLMUID.PROJDEFS.ACTINFO
             ( MESSAGE )
             FLMALLOC    IOTYPE=W,RECFM=VB,LRECL=256,DISP=MOD,                   C
                RECNUM=4   ,DDNAME=MESSAGE,PRINT=I
             ( MSGXFER )
             FLMALLOC    IOTYPE=W,RECFM=VB,LRECL=256,CATLG=Y,                    C
                RECNUM=4   ,DDNAME=MSGXFER
             ( BMAP )
             FLMALLOC    IOTYPE=W,RECFM=VB,LRECL=256,                            C
                RECNUM=4   ,DDNAME=BMAP,PRINT=I
             ( FILES )
             FLMALLOC    IOTYPE=W,RECFM=VB,LRECL=256,                            C
                RECNUM=4   ,DDNAME=FILES,PRINT=I
             ( RESPONSE )
             FLMALLOC    IOTYPE=W,RECFM=VB,LRECL=256,                            C
                RECNUM=4   ,DDNAME=RESPONSE,PRINT=I,CATLG=Y
```

**Figure 23 (Part 1 of 3). checkout.lng SCLM TeamConnection CHECKOUT Language Definition**

The FLMLANG macro at statement (1) identifies the name of the language definition with the LANG keyword value. "CHECKOUT" is used so that it is easy to remember that this language definition is used for the TeamConnection CHECKOUT command.

The FLMTRNSL macro at statement (2) tells SCLM to pass control to ISPF SELECT (CALLMETH=ISPLNK and COMPILE=SELECT) when a member with this language is built (FUNCTN=BUILD).

The SELECT function receives the parameter string identified by the FLMTRNSL OPTIONS keyword parameter value. This parameter string tells SELECT to run the program FLMLTWST with the parameters ACTION=CHECKOUT,BMAPINFO=@@FLM$MP,... and so on. FLMLTWST looks at this parameter string and reads through the ACTINFO file (Figure 21 on page 11 ) looking for "ACTION CHECKOUT". The COMMAND and CPARM values that follow "ACTION CHECKOUT" along with directory information from the USERINFO file (Figure 2 on page 1 9) are used to construct the workstation command.

"ACTION CHECKOUT" is used so that it is easy to remember that this is used for the TeamConnection CHECKOUT command.
The LANG keyword value does not have to match the ACTION value.

The FLMALLOC macro at statement (3) allocates the file USERINFO to the MVS data set @@FLMUID.SCLM.USERINFO (where @@FLMUID is replaced by the TSO user id) at run time. This data set can have any other name and must exist at run time. The name used here is consistent with the USERINFO data set name used in "TR 29.2 34 Using ISPF's Software Configuration and Library Manager (SCLM) to Manage Workstation and Cooperative Applications".

The FLMALLOC macro at statement (4) allocates the file ACTINFO to the MVS data set @@FLMUID.PROJDEFS.ACTINFO (where @@FLMUID is replaced by the TSO user id) at run time. This data set can have any other name and must exist at run time. The name used here is consistent with the ACTINFO data set name used in "TR 29.2 34 Using ISPF's Software Configuration and Library Manager (SCLM) to Manage Workstation and Cooperative Applications".

The other five FLMALLOC macros are for work data sets (IOTYPE=W).

FLMLTWST also looks at the build map to determine what members to send to the workstation and what files to bring back from the workstation. The ACTINFO file is used to map the SCLM type (low level qualifier of the SCLM controlled library) to the workstation file extension. As an added bonus FLMLTWST erases the message files before running the workstation command so that messages from a previous command are not mistaken for messages from the current command.

**Figure 23 (Part 2 of 3). checkout.lng SCLM TeamConnection CHECKOUT Language Definition**

FLMLTWST is provided as a sample with ISPF and can be found in a system data set such as 'SYS1.ISP.SISPSAMP'. You can use FMLTWST "as is" or modify it to meet other needs. The techniques described in this paper use FLMLTWST as delivered in ISPF 4.2.

**Figure 23 (Part 3 of 3). checkout.lng SCLM TeamConnection CHECKOUT Language Definition**

```
        SCLM LANGUAGE DEFINITION FOR TEAMCONNECTION CHECKIN

    THIS LANGUAGE DEFINITION IS AN EXAMPLE THAT CAN SERVE AS A
    REFERENCE IN THE CONSTRUCTION AND CUSTOMIZATION OF LANGUAGE
    DEFINITIONS FOR EXECUTING TEAMCONNECTION COMMANDS.

    THE NUMBERS IN PARENTHESIS ARE USED FOR REFERENCE IN THE EXPLANATION
    THAT FOLLOWS AND ARE NOT PART OF THE LANGUAGE DEFINITION

CHECKIN   FLMLANGL      LANG=CHECKIN,                                    (1)      C
                VERSION=1,CHKSYSLB=IGNORE

 EXECUTE TEAMCONNECTION COMMAND - CHECKIN
        ACTION is defined in TSOUSER.PROJDEFS.ACTINFO
           FLMTRNSL     CALLNAM='TC  CHECKIN',                          (2)      C
                FUNCTN=BUILD,                                                C
                CALLMETH=ISPLNK,                                             C
                COMPILE=SELECT,                                              C
                VERSION=1,                                                   C
                GOODRC= ,                                                    C
                PORDER=1,                                                    C
                OPTIONS='CMD(FLMLTWST  ACTION=CHECKIN,BMAPINFO=@@FLM$MP,SC
                CLMINFO=@@FLMINF,BLDINFO=@@FLMBIO,PARMS='
           ( USERINFO )
           FLMALLOC   IOTYPE=A,DDNAME=USERINFO                           (3)
              FLMCPYLB  @@FLMUID.SCLM.USERINFO
           ( ACTINFO )
           FLMALLOC   IOTYPE=A,DDNAME=ACTINFO                            (4)
              FLMCPYLB  @@FLMUID.PROJDEFS.ACTINFO
           ( MESSAGE )
           FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,DISP=MOD,                    C
                RECNUM=4   ,DDNAME=MESSAGE,PRINT=I
           ( MSGXFER )
           FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,CATLG=Y,                     C
                RECNUM=4   ,DDNAME=MSGXFER
           ( BMAP )
           FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,                             C
                RECNUM=4   ,DDNAME=BMAP,PRINT=I
           ( FILES )
           FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,                             C
                RECNUM=4   ,DDNAME=FILES,PRINT=I
           ( RESPONSE )
           FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,                             C
                RECNUM=4   ,DDNAME=RESPONSE,PRINT=I,CATLG=Y
```

**Figure 24 (Part 1 of 3). checkin.lng SCLM TeamConnection CHECKIN Language Definition**

The FLMLANG macro at statement (1) identifies the name of the language definition with the LANG keyword value. "CHECKIN" is used so that it is easy to remember that this language definition is used for the TeamConnection CHECKIN command.

The FLMTRNSL macro at statement (2) tells SCLM to pass control to ISPF SELECT (CALLMETH=ISPLNK and COMPILE=SELECT) when a member with this language is built (FUNCTN=BUILD).

The SELECT function receives the parameter string identified by the FLMTRNSL OPTIONS keyword parameter value. This parameter string tells SELECT to run the program FLMLTWST with the parameters ACTION=CHECKIN,BMAPINFO=@@FLM$MP,... and so on. FLMLTWST looks at this parameter string and reads through the ACTINFO file (Figure 21 on page 11 ) looking for "ACTION CHECKIN". The COMMAND and CPARM values that follow "ACTION CHECKIN" along with directory information from the USERINFO file (Figure 2 on page 1 9) are used to construct the workstation command.

"ACTION CHECKIN" is used so that it is easy to remember that this is used for the TeamConnection CHECKIN command.
The LANG keyword value does not have to match the ACTION value.

The FLMALLOC macro at statement (3) allocates the file USERINFO to the MVS data set @@FLMUID.SCLM.USERINFO (where @@FLMUID is replaced by the TSO user id) at run time. This data set can have any other name and must exist at run time. The name used here is consistent with the USERINFO data set name used in "TR 29.2 34 Using ISPF's Software Configuration and Library Manager (SCLM) to Manage Workstation and Cooperative Applications".

The FLMALLOC macro at statement (4) allocates the file ACTINFO to the MVS data set @@FLMUID.PROJDEFS.ACTINFO (where @@FLMUID is replaced by the TSO user id) at run time. This data set can have any other name and must exist at run time. The name used here is consistent with the ACTINFO data set name used in "TR 29.2 34 Using ISPF's Software Configuration and Library Manager(SCLM) to Manage Workstation and Cooperative Applications".

The other five FLMALLOC macros are for work data sets (IOTYPE=W).

FLMLTWST also looks at the build map to determine what members to send to the workstation and what files to bring back from the workstation. The ACTINFO file is used to map the SCLM type (low level qualifier of the SCLM controlled library) to the workstation file extension. As an added bonus FLMLTWST erases the message files before running the workstation command so that messages from a previous command are not mistaken for messages from the current command.

**Figure 24 (Part 2 of 3). checkin.lng SCLM TeamConnection CHECKIN Language Definition**

FLMLTWST is provided as a sample with ISPF and can be found in a system data set such as 'SYS1.ISP.SISPSAMP'. You can use FMLTWST "as is" or modify it to meet other needs. The techniques described in this paper use FLMLTWST as delivered in ISPF 4.2.

**Figure 24 (Part 3 of 3). checkin.lng SCLM TeamConnection CHECKIN Language Definition**

```
        SCLM LANGUAGE DEFINITION FOR TEAMCONNECTION BUILD

    THIS LANGUAGE DEFINITION IS AN EXAMPLE THAT CAN SERVE AS A
    REFERENCE IN THE CONSTRUCTION AND CUSTOMIZATION OF LANGUAGE
    DEFINITIONS FOR EXECUTING TEAMCONNECTION COMMANDS.

    THE NUMBERS IN PARENTHESIS ARE USED FOR REFERENCE IN THE EXPLANATION
    THAT FOLLOWS AND ARE NOT PART OF THE LANGUAGE DEFINITION

TCBUILD   FLMLANGL      LANG=TCBUILD,                                    (1)      C
                VERSION=1,CHKSYSLB=IGNORE

 EXECUTE TEAMCONNECTION COMMAND - BUILD
        ACTION is defined in TSOUSER.PROJDEFS.ACTINFO
           FLMTRNSL     CALLNAM='TC  BUILD',                            (2)      C
                FUNCTN=BUILD,                                                  C
                CALLMETH=ISPLNK,                                                C
                COMPILE=SELECT,                                                C
                VERSION=1,                                                    C
                GOODRC= ,                                                     C
                PORDER=1,                                                     C
                OPTIONS='CMD(FLMLTWST  ACTION=BUILD,BMAPINFO=@@FLM$MP,SCLC
                MINFO=@@FLMINF,BLDINFO=@@FLMBIO,PARMS='
            ( USERINFO )
            FLMALLOC   IOTYPE=A,DDNAME=USERINFO                          (3)
              FLMCPYLB  @@FLMUID.SCLM.USERINFO
            ( ACTINFO )
            FLMALLOC   IOTYPE=A,DDNAME=ACTINFO                           (4)
              FLMCPYLB  @@FLMUID.PROJDEFS.ACTINFO
            ( MESSAGE )
            FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,DISP=MOD,                    C
                RECNUM=4   ,DDNAME=MESSAGE,PRINT=I
            ( MSGXFER )
            FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,CATLG=Y,                     C
                RECNUM=4   ,DDNAME=MSGXFER
            ( BMAP )
            FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,                           C
                RECNUM=4   ,DDNAME=BMAP,PRINT=I
            ( FILES )
            FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,                           C
                RECNUM=4   ,DDNAME=FILES,PRINT=I
            ( RESPONSE )
            FLMALLOC   IOTYPE=W,RECFM=VB,LRECL=256,                           C
                RECNUM=4   ,DDNAME=RESPONSE,PRINT=I,CATLG=Y
```

**Figure 25 (Part 1 of 3). tcbuild.lng SCLM TeamConnection TCBUILD Language Definition**

The FLMLANG macro at statement (1) identifies the name of the language definition with the LANG keyword value. "TCBUILD" is used so that it is easy to remember that this language definition is used for the TeamConnection BUILD command.

The FLMTRNSL macro at statement (2) tells SCLM to pass control to ISPF SELECT (CALLMETH=ISPLNK and COMPILE=SELECT) when a member with this language is built (FUNCTN=BUILD).

The SELECT function receives the parameter string identified by the FLMTRNSL OPTIONS keyword parameter value. This parameter string tells SELECT to run the program FLMLTWST with the parameters ACTION=BUILD,BMAPINFO=@@FLM$MP,... and so on. FLMLTWST looks at this parameter string and reads through the ACTINFO file (Figure 21 on page 11 ) looking for "ACTION BUILD". The COMMAND and CPARM values that follow "ACTION BUILD" along with directory information from the USERINFO file (Figure 2 on page 1 9) are used to construct the workstation command.

"ACTION BUILD" is used so that it is easy to remember that this is used for the TeamConnection BUILD command.
The LANG keyword value does not have to match the ACTION value.

The FLMALLOC macro at statement (3) allocates the file USERINFO to the MVS data set @@FLMUID.SCLM.USERINFO (where @@FLMUID is replaced by the TSO user id) at run time. This data set can have any other name and must exist at run time. The name used here is consistent with the USERINFO data set name used in "TR 29.2 34 Using ISPF's Software Configuration and Library Manager (SCLM) to Manage Workstation and Cooperative Applications".

The FLMALLOC macro at statement (4) allocates the file ACTINFO to the MVS data set @@FLMUID.PROJDEFS.ACTINFO (where @@FLMUID is replaced by the TSO user id) at run time. This data set can have any other name and must exist at run time. The name used here is consistent with the ACTINFO data set name used in "TR 29.2 34 Using ISPF's Software Configuration and Library Manager (SCLM) to Manage Workstation and Cooperative Applications".

The other five FLMALLOC macros are for work data sets (IOTYPE=W).

FLMLTWST also looks at the build map to determine what members to send to the workstation and what files to bring back from the workstation.
The ACTINFO file is used to map the SCLM type (low level qualifier of the SCLM controlled library) to the workstation file extension. As an added bonus FLMLTWST erases the message files before running the workstation command so that messages from a previous command are not mistaken for messages from the current command.

**Figure 25 (Part 2 of 3). tcbuild.lng SCLM TeamConnection TCBUILD Language Definition**

FLMLTWST is provided as a sample with ISPF and can be found in a system data set such as 'SYS1.ISP.SISPSAMP'. You can use FMLTWST "as is" or modify it to meet other needs. The techniques described in this paper use FLMLTWST as delivered in ISPF 4.2.

**Figure 25 (Part 3 of 3). tcbuild.lng SCLM TeamConnection TCBUILD Language Definition**

```
    THIS PROJECT DEFINITION IS DESIGNED TO TEST EXECUTION OF
    TEAMCONNECTION COMMANDS ON A WORKSTATION.

    DO NOT FORGET TO CHANGE TSOUSER TO YOUR TSO ID.
    DO NOT FORGET TO ASSEMBLE and LINK AFTER ANY CHANGE.

          TITLE '    PROJECT DEFINITION FOR TEAM CONNECTION    '
TSOUSER   FLMABEG


            DEFINE THE TYPES (DEFINE THE LOW LEVEL QUALIFIERS)

ARCHDEF   FLMTYPE
SOURCE    FLMTYPE

 WORKSTATION TYPES

SRC         FLMTYPE TSXFER='SEND:ASCII CRLF LRECL(8 ) RECFM(F)                C
                RECEIVE:ASCII'
BIN         FLMTYPE  TSXFER='SEND:    RECEIVE:'


            DEFINE THE GROUPS    (DEFINES MIDDLE QUALIFIER)

DEVELOP   FLMGROUP  KEY=Y,AC=P


                PROJECT CONTROLS (VSAM)

         FLMCNTRL  ACCT=TSOUSER.ACCOUNT.FILE


                LANGUAGE DEFINITION TABLES

          COPY    CHECKOUT             -- TEAM CONNECTION CHECKOUT        --
                                      FOR FLMLTWST LANG.DEF.
          COPY    CHECKIN               -- TEAM CONNECTION CHECKIN         --
                                      FOR FLMLTWST LANG.DEF.
          COPY    TCBUILD               -- TEAM CONNECTION BUILD           --
                                      FOR FLMLTWST LANG.DEF.
          COPY    DUMMY                 -- DUMMY LANGUAGE FOR TESTING     --


 NON-COMPILERS (STANDARD, NO MODIF NEEDED)

          COPY    FLM@ARCD              -- ARCHITECTURE DEF. LANGUAGE --

         FLMAEND
```

**Figure 26. sclmtc.prj SCLM Project Definition**

```
PROC
ALLOC FI(SYSEXEC) DA('sys1.isp.sispsamp') shr reu
```

**Figure 27. CLIST to concatenate a library containing FLMLTWST to SYSEXEC**

# APPENDIX E. TEAMCONNECTION BUILD SCRIPTS AND PARTS FOR SCLM

```
//REXX        EXEC    PGM=IRXJCL,PARM=&TCPARM
//
//  BUILD SCRIPT FOR EXECUTING A REXX PROGRAM
//
//  TEAMPROC MUST RUN IN A TSO ADDRESS SPACE
//
//  IRXJCL WILL EXECUTE THE FIRST PARAMETER IN &TCPARM
//  AS A REXX PROGRAM AND PASS THE REMAINING PARAMETERS
//
//  IRXJCL WILL LOOK INTO SYSEXEC FOR THE REXX PROGRAM
//  BE SURE TO CHANGE TSOUSER TO YOUR TSO USER ID
//
//  SCLMPRM WILL CONTAIN 4 RECORDS FOR THE SCLM COMMAND:
//  COMMAND, GROUP, TYPE, AND MEMBER
//
//SYSEXEC   DD  DSN=TSOUSER.TEAMC.REXX,DISP=SHR
//SCLMPRM   DD  TCEXT=PRM,DISP=(NEW,DELETE),
//              UNIT=VIO,SPACE=(32   ,(3 ,1 )),
//              DCB=(RECFM=FB,LRECL=8 ,BLKSIZE=32  )
//ERRLIST   TCEXT=TCOUT,DISP=(NEW,DELETE),
//              UNIT=SYSDA,SPACE=(133  ,(3 ,3 )),
//              DCB=(RECFM=FB,LRECL=133,BLKSIZE=133 )
//ISPTLIB   DD   DSN=SYS1.ISP.SISPTENU,DISP=SHR
//ISPSLIB   DD   DSN=SYS1.ISP.SISPSENU,DISP=SHR
//ISPMLIB   DD   DSN=SYS1.ISP.SISPMENU,DISP=SHR
//ISPPLIB   DD   DSN=SYS1.ISP.SISPPENU,DISP=SHR
//ISPLOG     DD    DUMMY
//
```

**Figure 28. sclmcmd.jcl - build script to run MVS REXX programs**

```
/  REXX  /
parse arg templib parms
/                                                    /
/    BUILD SCLM COMMAND FROM FILE SCLMPRM RECEIVED FROM TEAMCONNECTION /
/                                                    /
dsinfo = getdsn_last('ISPTLIB')
if dsinfo >   then
   call errorhndlr GETISPTLIB,dsinfo
isptlib = SYSDSNAME

dsinfo = getdsn_last('ISPMLIB')
if dsinfo >   then
   call errorhndlr GETISPMLIB,dsinfo
ispmlib = SYSDSNAME

dsinfo = getdsn_last('ISPSLIB')
if dsinfo >   then
   call errorhndlr GETISPSLIB,dsinfo
ispslib = SYSDSNAME

dsinfo = getdsn_last('ISPPLIB')
if dsinfo >   then
   call errorhndlr GETISPPLIB,dsinfo
ispplib = SYSDSNAME

"ALLOC FI(SYSTERM) BLOCK(8  ) SPACE(5  ,5  ) UNIT(SYSDA) REUSE"
"ALLOC FI(ISPPROF) BLOCK(8  ) SPACE(5  ,5  ) DIR(1 ) UNIT(SYSDA),
          REUSE,DSORG(PO)"
"ALLOC FI(ISPSLIB)       DA('"ispslib"') SHR REUSE"
"ALLOC FI(ISPMLIB)       DA('"ispmlib"') SHR REUSE"
"ALLOC FI(ISPPLIB)       DA('"ispplib"') SHR REUSE"
/                                    /
/                                                                    /
/  To avoid ENQ errors on the ISPTENU system data set a        /
/  cataloged data set (templib) must be concatenated in        /
/  front of ISPTLIB.                                                 /
/                                                                    /
/        if data set specified then                                  /
/             concatenate data set in front of ISPTLIB              /
/       else                                                         /
/             point directly to ISPTLIB and risk enq error          /
/                                                                    /
/                                    /
"FREE   FI(ISPTLIB)"
if templib <> '' then
     "ALLOC FI(ISPTLIB)       DA('"templib"','"isptlib"') SHR REUSE"
else
     "ALLOC FI(ISPTLIB)       DA('"isptlib"') SHR REUSE"
```

**Figure 29 (Part 1 of 4). sclmcmd.rex - MVS REXX program to issue SCLM BUILD**

```
/                                        /
/                                                                    /
/  CONSTRUCT THE FLMCMD PARAMETER STRING FROM FILE SCLMPRM  /
/  BE SURE TO CHANGE TSOUSER TO YOUR TSO USER ID                    /
/                                                                    /
/                                        /

call getsclmprm

buildcmd = sclmcmd                    ||,
                ',tsouser,sclmtc,'  ||,
                sclmgrp                 ||,
                ','                     ||,
                sclmtyp                 ||,
                ','                     ||,
                sclmmem                   ||,
                ',,,c,,y'

say 'starting sclm build'

/                                        /
/                                                                    /
/  ISPSTART RETURN CODE WILL BE ZISPFRC IN THE SHARED POOL. /
/  EXFLMCMD WILL PUT THE RETURN CODE FROM FLMCMD IN ZISPFRC /
/  IN THE SHARED POOL.                                               /
/                                                                    /
/                                        /

"ispstart cmd(exflmcmd "buildcmd")"

   flmcmd_rc = rc

   if flmcmd_rc <  | flmcmd_rc >  then
      call errorhndlr FLMCMD,flmcmd_rc,SYSTERM
  else
     "FREE  FI(SYSTERM)"

say 'finishing sclm build'

exit
/                                              /
```

**Figure 29 (Part 2 of 4). sclmcmd.rex - MVS REXX program to issue SCLM BUILD**

```
getsclmprm: procedure expose sclmcmd sclmgrp sclmtyp sclmmem
/                                                  /
/    GET SCLM PARAMETERS FROM SCLMPRM DATA DEFINITION                          /
/                                                  /

 dsinfo = getdsn_last('SCLMPRM')
 if dsinfo >   then
     call errorhndlr GETSCLMPRM,dsinfo

 sclmprm. = ''
 "ALLOC FI(SCLMPRMX) DA('"SYSDSNAME"') SHR REUSE"
 "execio   diskr SCLMPRMX (stem sclmprm. open finis)"

 /  SCLMPRM contains one line for each SCLM command parameter.            /
 /  The parameters are:                                                    /
 /  SCLM command                                                             /
 /  SCLM group                                                            /
 /  SCLM type                                                             /
 /  SCLM member                                                             /

 sclmcmd = word(sclmprm.1,1)
 sclmgrp = word(sclmprm.2,1)
 sclmtyp = word(sclmprm.3,1)
 sclmmem = word(sclmprm.4,1)

 return

/                                                  /
```

**Figure 29 (Part 3 of 4). sclmcmd.rex - MVS REXX program to issue SCLM BUILD**

```
errorhndlr: procedure expose sysmsglvl1 sysmsglvl2
/                                                   /
/    ISSUE ERROR MESSAGE FOR FUNCTION, ERROR CODE, AND DATA SET             /
/                                                   /
parse arg function, return_code, copyddname
dsinfo = getdsn_last('ERRLIST')
"ALLOC FI(ERROUT) DA('"SYSDSNAME"') MOD REUSE"
if copyddname <> '' then
  do
     "execio   diskr "copyddname" (stem errvar. open finis)"
     "execio   diskw errout (STEM errvar. open finis)"
    "FREE  FI("copyddname")"
  end
else
   'EXECIO 2 DISKW ERROUT (STEM sysmsglvl OPEN FINIS)'
"FREE FI(ERROUT)"
exit(return_code)

getdsn_last: procedure expose sysdsname sysmsglvl1 sysmsglvl2
parse arg ddname parm_rest
getdsn ddname
getdsn_rc = rc
if getdsn_rc <  | getdsn_rc >  then
   sysdsname = ''
else
  do
     last_dsn = sysdsname.
     sysdsname = sysdsname.last_dsn
  end
return getdsn_rc
```

**Figure 29 (Part 4 of 4). sclmcmd.rex - MVS REXX program to issue SCLM BUILD**

```
/ REXX /
parse arg cmdparm

/                                                  /
/                                                                    /
/ receive the flmcmd parameter string                          /
/ run the flmcmd                                                    /
/ save the return code in ZISPFRC                                /
/ put the return code in a shared pool for ISPSTART            /
/                                                                    /
/                                                  /

'flmcmd ' cmdparm
ZISPFRC = rc
ADDRESS ISPEXEC 'VPUT (ZISPFRC) SHARED'
exit
```

**Figure 30. exflmcmd.rex - MVS REXX program to issue SCLM flmcmd**

```
//your job card
//                                        /
//  PROGRAM:    RUNPGMT                                                  /
//  JCL to start TeamConnection MVS Build Server under          /
//  a TSO environment.    Primary use is intended for               /
//  VisualGen support.                                                  /
//                                        /
//     Some dataset names might need to be modified                 /
//     according to your system's customization                     /
//                                        /
//
//RUNPGMT EXEC PGM=IKJEFT 1,REGION=6M,DYNAMNBR=3
//STEPLIB DD DSN=DSNA.DSNLOAD,DISP=SHR
//              DD  DSN=userid.TEAMC.LOADLIB,DISP=SHR
//              DD  DSN=SYS1.CEE.SCEERUN,DISP=SHR
//              DD  DSN=SYS1.COB2COMP,DISP=SHR
//             DD  DSN=SYS1.CEE.SCEELKED,DISP=SHR
//  userid.TEAMC.TEST.JCL must be allocated as a PDS with lrecl=8 ,
//  and recfm=FB
//TEAMPROC DD DSN=userid.TEAMC.TEST.JCL,DISP=SHR
//EDCENV     DD    DSN=userid.TEAMC.ENVVAR,DISP=SHR
//STDOUT     DD  SYSOUT=
//STDERR     DD  SYSOUT=
//SYSUDUMP DD SYSOUT=
//CEEDUMP   DD  SYSOUT=
//SYSTSPRT DD SYSOUT=
//SYSTSIN DD
 CALL 'userid.TEAMC.LOADLIB(TEAMPROC)'    +
        'ENVAR("_CEE_ENVFILE=DD:EDCENV")/-S @32911 -U SYSDA -K IBM-1 47'
//
```

**Figure 31. Sample JCL Using LE/370 for RUNPGMT**

# APPENDIX F. SCLM FLMLTWST EXTENSIONS FOR PARAMETERS

This section describes the extensions to FLMLTWST found in flmltwst.rex in the demonstration materials (Appendix A, “TeamConnection Parts and Scripts” on page 69). These extensions were made to FLMLTWST as found in ISPF 4.2. FLMLTWST is written in REXX.

FLMLTWST in flmltwst.rex is extended through code changes to recognize the following parameters. These keyword parameters can be specified in the OPTIONS keyword parameter value of the FLMTRNSL macro in an SCLM language definition or specified in the PARM or PARMx statements of an SCLM architecture definition.

SENDINPUT

- YES - send inputs from MVS to the workstation
- NO - do not send inputs to the workstation
- DEFAULT = YES

GETOUTPUT

- YES - send outputs from the workstation to MVS
- NO - do not send outputs to MVS
- DEFAULT = YES

TCACTFLAG

- the value 'TCACTFLAG' in a 'CPARM TCACTFLAG' statement from an ACTINFO file (Figure 21 on page 110) is replaced with the value of the TCACTFLAG keyword parameter
- this parameter is intended to be a TeamConnection command action flag but can be used for any CPARM statement
- DEFAULT = -build

PARTNAME

- the value 'PARTNAME' in a 'CPARM PARTNAME' statement from an ACTINFO file (Figure 21 on page 110) is replaced with the value of the PARTNAME keyword parameter
- this parameter is intended to be the name of a TeamConnection part but can be used for any CPARM statement
- DEFAULT = PARTNAME

See “SCLM Architecture Definition Parameters for TeamConnection Commands” on page 61 for a complete discussion of how to use these extensions.

All extensions in flmltwst.rex are surrounded by comments. Altered code is preserved in its original state as part of the comments. In the following figures a few lines of code sometimes appear before and/or after the extensions to assist in locating the changes in flmltwst.rex.

## ADDITIONAL KEYWORD PARAMETERS

FLMLTWST puts keyword parameters in the stem variable parm_list. No changes are required to add additional keyword parameter values to this stem variable because they are picked up by a parse of the parameter string passed to FLMLTWST.

Default values for keyword parameters are assigned following the comment /* **initialize parm defaults** */ in the **initialization** subroutine. The code added for assigning defaults to the additional keyword parameters is in Figure 32.

```
   /  initialize parm defaults                                          /

/                    ADDED FOR TeamConnection       START  /
   parm_name = 'SENDINPUT'
   parm_list.parm_name = 'YES'

   parm_name = 'GETOUTPUT'
   parm_list.parm_name = 'YES'

   parm_name = 'TCACTFLAG'
   parm_list.parm_name = '-build'

   parm_name = 'PARTNAME'
   parm_list.parm_name = 'PARTNAME'
/                    ADDED FOR TeamConnection         END  /

   parm_name = 'ACTION'
   parm_list.parm_name = 'COMPILE'
```

**Figure 32. FLMLTWST - adding keyword parameter defaults**

## AVOIDING TRANSFER OF FILES

FLMLTWST is designed to transfer input files to the workstation and output files from the workstation. The code added to the procedure to avoid transfers based upon the value of **SENDINPUT** and **GETOUTPUT** keyword parameters is in Figure 33 and Figure 34. An additional test is made for the value of the keyword parameter to avoid the transfer of files.

```
        / Send the inputs, and 1st response file /
/                    ADDED FOR TeamConnection       START /
/                                                                          /
   / IF ret_code =   THEN /
  /     CALL send_inputs      /
        parm_name = 'SENDINPUT'
        IF (ret_code =  ) & (parm_list.parm_name = 'YES') THEN
          CALL send_inputs
/                                                                          /
/                    ADDED FOR TeamConnection          END /
```

**Figure 33. FLMLTWST - avoiding transfer of input files**

```
       / Retreive the outputs from the workstation /
/                    ADDED FOR TeamConnection       START /
/                                                                          /
   / IF (ret_code =  ) & (xfer_output.  >  ) THEN /
  /     CALL get_outputs                                  /
        parm_name = 'GETOUTPUT'
        IF (ret_code =  ) & (xfer_output.  >  ) &,
            (parm_list.parm_name = 'YES') THEN
          CALL get_outputs
/                                                                          /
/                    ADDED FOR TeamConnection          END /

EXIT ret_code
```

**Figure 34. FLMLTWST - avoiding transfer of output files**

## PARAMETER FOR A CPARM VALUE

The CPARM values in the ACTINFO file (Figure 21 on page 110) are used by FLMLTWST to create the workstation command. The **get_actinfo** subroutine reads the ACTINFO file for the CPARM values. Code was added to recognize **TCACTFLAG** and **PARTNAME** as CPARM values and replace these values with the respective values of the TCACTFLAG and PARTNAME keyword parameters. The use of stem variables simplifies the code as seen in Figure 35.

```
                WHEN var_name = 'CPARM' THEN
                  DO
/                       ADDED FOR TeamConnection       START  /
                      IF (var_value = 'TCACTFLAG') |,
                          (var_value = 'PARTNAME')       THEN
                         var_value = parm_list.var_value
/                       ADDED FOR TeamConnection         END  /
                      cnt = action_parms.action + 1
                      action_parms.action = cnt
```

**Figure 35. FLMLTWST - CPARM value replaced by parameter value**