



Com-plete

Version 6.3.1

Installation

This document applies to Complete Version 6.3.1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

© Copyright Software AG 2003
All rights reserved.

The name Software AG and/or all Software AG product names are either trademarks or registered trademarks of Software AG. Other company and product names mentioned herein may be trademarks of their respective owners.

Table of Contents

Overview of the Installation Documentation	1
Overview of the Installation Documentation	1
Installation and Migration	2
Installation and Migration	2
General Information	3
General Information	3
Supported Operating Systems	3
Installation Jobs	3
Using System Maintenance Aid	4
Installation Data Set Naming Convention	4
All Operating Systems:	4
z/OS only:	4
VSE only:	5
z/OS Installation	6
z/OS Installation	6
The Installation Tape	6
Tape Contents	6
Copying Contents of the Tape to Disk	7
Installation Steps	8
Startup Procedure	17
What Next?	18
VSE Installation	19
VSE Installation	19
The Installation Tape	19
Tape Contents	19
Copying Contents of the Tape to Disk	20
Installation Steps	21
Startup Procedure	26
What Next?	27
Migration	29
Migration	29
Changes to the Start-up Procedure	29
Changes to the SYSPARMS	30
COMSYS Data Containers	30
Reducing the Thread Region Size for most Complete Utilities	30
z/OS Sample JCL	31
z/OS Sample JCL	31
A. Download Datasets from the Supplied Tape	31
B. JCLINST1	32
C. JCLINST2	33
D. JCLINST3	33
E. JCLINST4	35
F. JCLINST5	35
G. JCLINST7	36
H. JCLINST8	37
I. JCLINST9	38
J. JCLINSTB	38
K. JCLINSTC	39

L. JCLINSTE	40
VSE Sample JCL	42
VSE Sample JCL	42
A. JCLINST1	42
B. JCLINST2	43
C. JCLINST4	43
D. JCLINST5	46
E. JCLINST6	47
F. JCLINST7	48
G. JCLINST8	49
H. JCLINST9	50
Installation and Customization of TPF	51
Installation and Customization of TPF	51
Installing TPF in the Host Environment	52
Installing TPF in the Host Environment	52
TPF Facilities Overview	52
Installation under CICS	53
Installation under TSO (z/OS only)	55
Installation under Com-plete	57
What Next?	58
General Logon Authorization	59
General Logon Authorization	59
User ID Considerations	59
Program Authorization	59
Password Considerations	59
The ACSTAB Table	61
The ACSTAB Table	61
ACSDEF Subtable	61
ACSTBL Subtable	62
ACSSCHC Subtable (CICS only)	63
TPFXTAB Subtable	63
Passing Data	69
Overriding Information	70
CICS: Special Considerations	73
CICS: Special Considerations	73
Printing Under TPF	73
Upper/Lower Case Considerations	74
Pre CICS 3.3	74
Post CICS 3.3	75
Transparency (CICS and TSO) and Reverse Access (CICS)	76
Transparency (CICS and TSO) and Reverse Access (CICS)	76
Transparency Under CICS and TSO	76
The ACCESS Program	76
The Transfer Program	76
Reverse Access (CICS only)	77
Calling Reverse Access with Immediate Return	78
Calling Reverse Access with Delayed Return	80
Exits for the TPF Component	81
Exits for the TPF Component	81
USERID exit	81
NODEID exit	82

TPXFEX exit 83

Exits in ACCESS46 83

Starting Access from a User Program 84

Starting Access from a User Program 84

Overview of the Installation Documentation

This documentation contains all information required to install Com-plete 6.3 as a first-time installation on all supported platforms, and provides migration procedures for sites wishing to migrate from an earlier version of Com-plete.

If yours is a first-time installation, refer to the System Programming documentation for details as to customizing the system to the needs of your site once Com-plete is successfully installed.

The Com-plete installation documentation is organized as follows:

• Installation and Migration	Describes procedures for installing Com-plete on the supported operating systems, including sample JCL.
• Installation and Customization of TPF	Describes procedures for installing and customizing the TPF component in the supported environments.

Installation and Migration

This part of the Complete installation documentation describes installation procedures for z/OS and VSE and provides sample JCL for installation jobs.

This information is organized under the following headings:

- General Information
- z/OS Installation
- VSE Installation
- Migration
- z/OS Sample JCL
- VSE Sample JCL

General Information

This chapter covers the following topics:

- Supported Operating Systems
 - Installation Jobs
 - Using System Maintenance Aid
 - Installation Data Set Naming Convention
-

Supported Operating Systems

Com-plete 6.3 runs under the most current levels of the IBM operating systems z/OS and VSE. In this documentation, the terms "z/OS" and "VSE" are used to refer to all supported levels.

Supported levels of z/OS:

Lowest Supported:	OS/390 2.8
Highest Tested:	z/OS 1.4

Supported levels of VSE:

Lowest Supported:	VSE/ESA 2.5
Highest Tested:	VSE/ESA 2.7

Note:

The lower level of the supported operating systems indicated above is the more important, as Com-plete 6.3 does NOT support any release before this level. The higher level is used as a general indicator, as the Com-plete code is upward-compatible and therefore can run on a higher level of the operating system. However, you are recommended to contact Com-plete support if you wish to install the product under an operating system level that falls outside the indicated range.

Installation Jobs

The installation of SOFTWARE AG products is performed by installation jobs. These jobs are either created "documentationonly" or generated by SYSTEM MAINTENANCE AID (SMA).

For each step of the installation procedure described in this section, the job number of a job performing the respective task is indicated. This job number refers to an installation job generated by SMA.

If you are not using SMA, an example installation job of the same number is provided in the installation job library on the Com-plete installation tape; you must adapt this example job to your requirements. Please note that the installation job numbers on the tape are preceded by a product code.

It is always advisable to have the distributed examples available, and you are therefore recommended to save modified examples in an installation specific data set, for example COM.USER.SOURCE.

Note:

On VSE systems, all example jobs are cataloged with MEMBERTYPE=J.

All output classes are specified as "?", indicating that the output class should be taken from the job card. You must set the class before submitting the job.

Using System Maintenance Aid

Information on using Software AG's System Maintenance Aid (SMA) for the installation process is provided by the *System Maintenance Aid* documentation.

Installation Data Set Naming Convention

Throughout all Com-plete documentation, the various Com-plete data sets are consistently referred to by the same name and the Com-plete short ID, if applicable. You must change these names to reflect your installation standards and/or to enable the installation of multiple copies of Com-plete (see the description of Com-plete's files in the *Com-plete System Programming* documentation)

Below is a list of the data sets together with a description of their content and an indication of how they are created. These data set names refer to the installation data sets of the current release. See also the contents of the installation tape on the report of your tape creation system.

All Operating Systems:

COM.COMSYS.xxxx

These are the Com-plete system files in VSAM REPRO format. They are created during the first part of the installation process and contain all data sets that must be loaded into the Com-plete COMSYS files.

COM.SPOOL

This is the printout spool / message data set and must be defined as a VSAM data set. It is used by Com-plete to hold messages and printouts created using the Com-plete printout spooling and message switching facilities. It is created and initialized as part of the standard installation process.

COM.SD

This data set is used by Com-plete to hold SD files and dumps created during Com-plete execution. It is allocated and initialized as part of the installation process and must be defined as a VSAM data set.

z/OS only:

COM.SOURCE

This is the source data set distributed with Com-plete. It is created as the first part of the installation process and contains all source members necessary to install and work with Com-plete (for example, macros, sample JCL, sample exits).

COM.LOAD

This is one of the two load libraries containing the Com-plete base system. It is created during the first part of the installation process and contains the Com-plete specific load modules.

APS.LOAD

This is the load library containing the Software AG Application Server. It is created during the first part of the installation process and contains modules which can be shared between Com-plete and other Software AG products.

COM.MAPS

This data set is used to contain any maps that users of Com-plete can create or modify using the Com-plete utility UMAP (see the Com-plete Utilities documentation). It is allocated during standard installation and is initially empty.

COM.USER.LOAD

This data set is a load library used for user modules and/or customized Com-plete modules. It is allocated during standard installation processing.

COM.USER.SOURCE

This data set is a user data set which you can use to store sample members from the distributed source data set which must be changed for the installation. You can use this data set to ensure that the distributed samples stay intact. It is allocated during standard installation processing.

VSE only:

COM.LIBRARY

This data set is created as the first part of the installation process and contains all members necessary to install and work with Com-plete (for example, macros, sample JCS, sample exits).

COM.DUMPFIL

This Com-plete file must be defined as a VSAM data set. It is used by Com-plete to save the abend information created during an abnormal termination of Com-plete. It is allocated as part of the installation process.

COM.USER.LIBRARY (Sublib=SAGLIB.COMUSER)

This data set is a load library used for user modules and/or customized Com-plete modules. This data set can also be used to store example members from the supplied source data set that must be changed for your installation. Use this data set to ensure that the supplied samples stay intact. This data set is allocated during standard installation processing.

z/OS Installation

The installation steps described in this document must be followed in sequence in order to install Com-plete successfully. You are recommended to use the provided examples up to successful initialization in order to ensure that the installation was successful. Once a functional system is available, you can start customizing the product as required.

The installation tape is created using IBM standard labels with a volume serial number of COM*vrs*. The tape can be ordered in any format supported by the SOFTWARE AG tape system.

Notes:

1. The notation *vrs* in the table below stands for version, release and system maintenance level.
2. For a description of installing the ADASVC, refer to the Adabas documentation.
3. The Com-plete Batch utilities require access batch (that is, the Adabas SVC must be installed).

This document covers the following topics:

- The Installation Tape
 - Installation Steps
 - Startup Procedure
 - What Next?
-

The Installation Tape

Tape Contents

Note:

The files may not be on the installation tape in the order shown below: please consult the report of your tape creation system. There may be no patch level libraries for a component if this is a new release of the respective component.

File Nr.	File Name	Description
1	SMT111.TABS	SMA tables
2	COMvrs.SRCE	Source data set
3	COMvrs.SRCT	TPF source data set
4	COMvrs.LOAD	Load library
5	COMvrs.LOA3	TPF load library
6	COMvrs.BASE	System file
7	COMvrs.USER	User ID file
8	COMvrs.CTLG	Catalog file
9	COMvrs.NS22	NSPOOL INPL data set
10	COMvrs.JOBS	Com-plete sample JCL library
11	APSVrs.SARG	Not currently used with Com-plete
12	APSVrs.SXML	Not currently used with Com-plete
13	APSVrs.LB00	Not currently used with Com-plete
14	APSVrs.LD00	SMARTS load library
15	APSVrs.LE00	Not currently used with Com-plete
16	APSVrs.SRCE	SMARTS source library
17	APSVrs.LDnn	SMARTS load library update (current patch level)
18	APSVrs.SRnn	SMARTS source library update (current patch level)
19	HTPVrs.LOAD	HTTP server load library
20	HTPVrs.SRCE	HTTP server sample source library
21	HTPVrs.JOBS	HTTP server sample JCL library
22	HTPVrs.INPL	HTTP server INPL data set
23	HTPVrs.GIFS	PDS containing GIF images for Natural Web Interface demo
24	HTPVrs.Lnnn	Most current patch level of HTPVrs.LOAD
25	HTPVrs.Snnn	Most current patch level of HTPVrs.SRCE
26	HTPVrs.Innn	Most current patch level of HTPVrs.INPL

Copying Contents of the Tape to Disk

Step 1: Unloading the Tape

1. Copy all files on the installation tape to disc using the IEBCOPY utility for the source and load data sets, and the IEBGENER utility for the system data. Please refer to section *z/OS Sample JCL* for sample JCL.

Note:

APSVrs.LDnn is mandatory for Com-plete 6.3.

2. Check for a README file on the delivered source. If a README file exists, it may contain important information concerning the installation and migration procedure.

Installation Steps

Step 1: Authorize Com-plete

Note:

Com-plete must run as an authorized user. To achieve this, you need only place module TLIN xx in an authorized data set, where xx can stand for "OS" for z/OS systems . Use one of the following alternative steps.

- Copy the module from the APSVrs.LDnn to a library that is already authorized and use the library as the STEPLIB for Com-plete. Sample JCL to copy the module is contained in member JCLINST1. A sample link job for the module is provided in JCLINST9.

Note:

If you do not use JCLINST1, you must ensure that whatever method you use to copy the module retains the correct AMODE/RMODE attributes and authorization code for the module.

Or:

Authorize the distributed Com-plete load library permanently by adding an entry for the data set in member IEAAPF nn of library SYS1.PARMLIB, where nn is the suffix used at your site.

Note:

This requires an IPL of the system. You can avoid having to perform an IPL for the initial installation if you have a product installed that allows automatic authorization of the data set.

Step 2: Allocate and Initialize the Com-plete Non-VSAM Data Sets

- Allocate the data sets needed to run Com-plete by modifying sample job JCLINST2 to reflect appropriate size and placements of data sets.

This job allocates all non-VSAM data sets:

```
COM.USER.SOURCE
COM.USER.LOAD
COM.MAPS
```

Step 3: Allocate and Initialize the Com-plete VSAM System Data Containers

- Only for a new installation:

Allocate, initialize and load the Com-plete system information data sets by modifying sample job JCLINST3 to reflect appropriate size and placement of the files. Please refer to the Com-plete Files and Associated User Files in the *Com-plete System Programming* documentation for more information on this VSAM file.

Step 4: Allocate and Initialize the Capture Data Set(s)

- Required only if capture is used at your installation.

Com-plete capture data sets are defined as VSAM data sets in the current version. Sample job JCLINST4 provides JCL to allocate and initialize two capture data sets. You must modify the job to reflect appropriate size and placement of the capture data sets.

Step 5: Allocating and Initializing the Com-plete SD Dataset

- Required.

The Com-plete SD dataset must be defined as a VSAM dataset in the current version. For detailed information on this file, please refer to the Com-plete Files and Associated User Files in the *Com-plete System Programming* documentation. To allocate and initialize the SD dataset, modify sample job JCLINSTE to reflect your requirements and run this job.

Step 6: Allocate and Initialize the Com-plete Spool Data Set

- Required.

The current version of Com-plete requires the spool data set to reside on a VSAM file. To allocate and initialize this file, modify sample job JCLINST5 to reflect appropriate size and placement of the data set. Then run the job.

Note:

The DD name used for this dataset in the Com-plete startup procedure must be *COMSPL*.

Step 7: Install the Com-plete Batch Interface

Required.

The Com-plete BATCH utilities and migration jobs require ACCESS when performing some or all of their functions.

Batch applications that use Com-plete services communicate with the target Com-plete via ACCESS. This communication is based on an Adabas SVC (ACCESS SVC) and a DBID (ACCESS NODE ID).

The Com-plete BATCH interface module loads a module with name ACSTAB and search this module for an entry name BATCH.

1. Modify the sample ACSTAB module on the source data set to reflect the system requirements for SVC and NODE ID.
2. Assemble and linkedit the module into the COM.USER.LOAD library. Modify sample job JCLINSTF to do this. The library specified as SYSLMOD must then be contained in the STEPLIB DD chain of the batch jobs.

Note that the DBID chosen may be greater than 255 and that the value of both DBID and SVC must correspond to the values of ACCESS-ID and ACCESS-SVC specified in the SYSPARM member.

Step 8: Assemble and Link the Adabas Link Module

- This module is delivered in source format as ADALCO on the Adabas distribution libraries. It must be assembled and linked in to a dataset contained in the COMPLIB concatenation.

If the version of Adabas you use is less than 6.3, the following source change must be applied to module ADALCO before assembly:

Locate the line

```
BP GOTWAIT
```

and replace it by

```
BNZ GOTWAIT
```

Note:

This module *must* be linked with AMODE=31

Step 9: Eligible Device Table Considerations

Required only if you are installing Com-plete under versions of z/OS prior to ESA V4.

The COM.SOURCE data set contains the operating system-dependent module UDSEDT. This module interrogates the EDT (Eligible Device Table) for UDS. The supplied module runs for all levels of z/OS after V4. If you are installing Com-plete under an z/OS version prior to ESA V4, proceed as follows:

1. Simply assemble the UDSEDT module (the module itself determines the level of z/OS for which it is being assembled).
2. Relink UDS using the linkedit step within the applicable operating system link job. See the description of Modifications to Com-plete Modules in the *Com-plete System Programming* documentation for more information on Com-plete linkedits.

See the *System Programming* documentation for more details on updating Com-plete utilities.

Step 10: Install the Com-plete High Level Language Interface

- Optional.

Applications that use Com-plete functions must be linked with the Com-plete high level language interface (HLLI) modules. These modules provide a standard interface between the application and Com-plete.

These modules are provided on the supplied load data set. If you are migrating from a previous version of Com-plete, some applications may need to be relinked with the new version of the HLLI modules. For details, refer to the description of Migration procedures.

Step 11: Install the JES Server Interface

The JES Interface modules are distributed in source to facilitate support of different levels of JES. DO NOT modify the code else Software AG cannot provide support for the modules.

The highest JES level supported is indicated in the source member. If the JES to which you wish to interface is at a higher level it might work, however it is recommended to contact the local support center for any updates.

The JES Interface uses the Com-plete sysparm SERVER to initialize, terminate and communicate with the JES interface server. Prior to defining any Com-plete sysparms, you must follow one of the following procedures.

1. Installing the Common JES2/JES3 Server Interface

This interface requires the following prerequisites:

OS/390 Version 2.9 or above; z/OS 1.1 and above	
JES2	The PTFs for APARs OW44349, OW52411, OW55671, OW55856 and OW57486 must be applied.
JES3	The PTFs for APARs OW34753, OW36022, OW49881, OW50990, OW56608, OW57110, OA01553, OA01715 and OA02735 must be applied. See also Information APAR II11784 for recommended JES3 SAPI maintenance.

1. For JES2, assemble TTJ2MVS and TTJIMVS using your current JES2 MACLIB. As an example you can use JCLINSTB.

For JES3, assemble TTJ3MVS and TTJIMVS using your current JES3 MACLIB. As an example you can use JCLINSTB.

2. Add the following SERVER Statement to the Com-plete sysparms:

For JES2:

```
SERVER=( ssss ,TLINJES2 )
```

For JES3:

```
SERVER=( ssss ,TLINJES3 )
```

where:

sss	is a unique server name within your Com-plete.
-----	--

2. If your IBM OS version does not meet the prerequisites to use the common JES interface, you can still use the old JES interfaces JES2SERV and JES3SERV. They are distributed in source to facilitate the support of different levels of JES. DO NOT modify the code, or Software AG cannot provide support for the modules anymore.

Installing the JES2 Server Interface

1. Assemble the JES2 Server interface module JES2SERV.

The JES2SERV module found on the source library must be assembled using the JES2 SYS1.SHASMAC (OS/390) or SYS1.HASPSRC (z/OS) data set for the JES2 with which you wish to work and the SYS1.MACLIB and SYS1.AMODGEN (SYS1.MODGEN for ESA V4) under which the JES2 subsystem will work. Member JCLINSTB on this source data set is a sample JCL stream to do this. You can give the load module resulting from this assembly and link any name you like.

2. Assemble the z/OS Active display format module TTJIMVS.

The TTJIMVS module found on the source library must be assembled using the SYS1.MACLIB and SYS1.AMODGEN (SYS1.MODGEN for ESA V4) for the z/OS level on which the JES2 interface will work. Member JCLINSTC on this source data set is a sample JCL stream to do this. The load module resulting from this assembly and link must be called TTJIMVS.

Note:

The two modules created by the jobs as described in steps 1 and 2 above must be linked with RMODE=ANY and AMODE=31, otherwise abends may occur at initialization.

3. Specify the SERVER sysparm as follows:

```
SERVER=( ssss ,TLINJES2 ,mmmmmmmm ,subs )
```

where:

ssss	is a unique server name within your Com-plete (e.g., 'JES2').
mmmmmmmm	is the name of the assembled and linked JES2 module.
subs	is the name of the JES2 subsystem on your system. The default is "JES2". Use this parameter if your JES2 system has a different name, or if you have more than one JES2 subsystems installed.

For example, the statement

```
SERVER=( JES ,TLINJES2 ,JES2SERV ,JES2 )
```

will cause Com-plete to use the JES2 server module JES2SERV, which will in turn attempt to interface with JES2 subsystem JES2, while the Com-plete server name will be JES.

Note:

If an error occurs during startup, a return and reason code is written in the message to the console. The reason codes can be found in member TTJIEQU on the supplied source dataset.

Installing the JES3 Server Interface

1. Allocate the JES3 communication file.

Note:

Applies only when installing Com-plete for the first time.

The installation job JCLINST7 allocates and initializes the z/OS file used by the DSP and the JIM. After this installation job has successfully completed, add the following DD card to the Com-plete start up procedure.

```
//UQJ3JOBS DD DSN=COM.JES3.UQJOBS,DISP=SHR
```

and add the following DYNALLOC statement to the JES3 initialization deck.

```
DYNALLOC,DDNAME=UQJOBS,DSNAME=COM.JES3.UQJOBS
```

2. Assemble the JES3 interface modules JES3SERV and IATUQJ3.

The JES3 modules JES3SERV and IATUQJ3 found on the source library must be assembled using the JES3 SYS1.JES3MAC data set for the level of JES3 with which you work and the SYS1.MACLIB and SYS1.AMODGEN (SYS1.MODGEN for ESA V4) under which the JES3 subsystem will work.

Member JCLINST8 is a sample JCL stream to do this.

Note:

Both JES3SERV and IATUQJ3 contain assembly variables which must be set to correspond to the version of JES3 under which they are to run. Module IATUQJ3 is upward compatible with previous releases of Com-plete (and also Natural system server).

The assembled and linked JES3SERV module must be placed in a load library contained in the COMPINIT concatenation. The module IATUQJ3 must be placed in a load library contained in the JES3 startup concatenation.

3. Add the DSP dictionary entry.

Note:

Applies only when installing Com-plete for the first time.

Add a DSP dictionary entry for the UQJ3 DSP, this update applies to the JES3 module IATGRPT.

You must first verify that the USERMOD name (SAG0001) and the sequence numbers do not conflict with existing installation modifications to JES3.

The following SMP statements are a sample to install the UQJ3 DSP entry for XA22 (JES3/HJS2220).

```
//SMPPTFIN DD *
++ USERMOD (SAG0001)
++ VER (Z038) FMID(HJS2220)
++ SRCUPD (IATGRPT)
./ CHANGE NAME=IATGRPT
      TITLE 'DSP FOR UQJ3'
SAG001
*-----
*SAG001          DSP DIRECTORY ENTRY FOR UQJ3 DSP
                SAG001
*-----
*SAG001
UQJ3          IATYSD PRTY=10,XABLE=YES,DRVR=IATUQJ3
SAG001
                SPACE
SAG001
//SMPCNTL DD *
      SET BDY(GLOBAL) .
```

```

RECIEVE S(SAG0001) SYSMOD .
SET BDY(XATRGZN) .
  APPLY S(SAG0001) .
/*

```

Once the above changes have been made, restart JES3 to pick them up. A HOTSTART restart can be done and the Com-plete DSP can then be started by entering "*X UQJ3", the DSP will continue to run and will restart itself over HOTSTART and WARMSTARTs.

Note:

The DSP must be active for the first time before you start Com-plete.

4. Specify the SERVER sysparm as follows:

```
SERVER=( ssss ,TLINJES3 ,mmmmmmmmmm )
```

where

ssss	is a unique server name within your Com-plete (e.g. JES3)
mmmmmmmmmm	is the name of the assembled and linked JES3 module (default is JES3SERV)

Step 12: Install the UDEBUG Environment

Optional.

Com-plete provides an online debugging utility (UDEBUG). In order to establish the working environment, proceed as follows:

1. Allocate two UDEBUG profile and symbol datasets as described in Com-plete Files and Associated User Files in the System Programming documentation.
2. Include the list of required UDEBUG residentpage modules in your SYSPARM member. A list of these modules is given in member DBUGSAMP in the supplied source datasets.

The UDEBUG facility is described in detail in the Com-plete Utilities documentation.

Step 13: Enable VTAM Interface

- To enable the VTAM interface, an ACB and Logmodes must be defined in the VTAM libraries. Refer to the description of the VTAM interface in the *Com-plete System Programming* documentation for the required definitions and parameters.

Step 14: Check Com-plete Startup Parameters (SYSPARM)

- A sample SYSPARM member is distributed in the Com-plete source library. This member can be used as a base for Com-plete customization. Although most parameters need not be changed, ensure that the following parameters reflect your environment:

Parameter	Value						
ACCESS-SVC	Specify the correct Adabas SVC number (see Step 7).						
ACCESS-ID	Specify the database ID for ACCESS (see Step 7).						
PATCHAR	Choose a unique single character for this installation (see Complete Startup Procedure in the System Programming documentation).						
SECSYS	Specify the security system running at your site (see Software Interfaces in the Complete System Programming documentation for details on the SECSYS parameter).						
SERVER *	<p>Required for Complete 6.3 and above:</p> <p><code>SERVER=(OPERATOR , TLINOPER) SERVER=(COMPLETE , TLINCOMP)</code></p> <p>Optional: Code one SERVER statement for each installed server according to the required syntax:</p> <table> <tr> <td>JES SERVER</td> <td>See Step 11 above.</td> </tr> <tr> <td>CONSOLE SERVER</td> <td>See Extended Console Server in the System Programming documentation.</td> </tr> <tr> <td>Natural Buffer Pool Manager</td> <td>See Natural Buffer Pool Manager in the System Programming documentation.</td> </tr> </table> <p>These SERVER statements must be put after the Complete server.</p>	JES SERVER	See Step 11 above.	CONSOLE SERVER	See Extended Console Server in the System Programming documentation.	Natural Buffer Pool Manager	See Natural Buffer Pool Manager in the System Programming documentation.
JES SERVER	See Step 11 above.						
CONSOLE SERVER	See Extended Console Server in the System Programming documentation.						
Natural Buffer Pool Manager	See Natural Buffer Pool Manager in the System Programming documentation.						
VTAMAPPL	Must specify Complete's ACB name (see Step 13).						

* You can optionally code one SERVER statement for each installed server according to the required syntax:

Step 15: Install NSPOOL

Optional.

The printout spooling facility NSPOOL is described in the Complete Application Programming documentation. The required dataset for running NSPOOL is on the Complete installation tape and is loaded as part of the Complete installation. The dataset name is COMvrs.NS22.

1. Run an INPL of NSPOOL.
2. If NSPOOL is to be used in a Natural environment running under Natural Security, you must stow the following modules in the Natural Security system library; otherwise, proceed to the next point.

```

SPSE01-N,
SPSE02-S,
SPSE01-&,
SPSE02-&,
SPSE03-&
SPAPPL-N *
SPUSER-N *

```

* required only for Natural Security version 2.2.8 and above.

See also the section *Default Authorization* below.

3. With Natural 2.2, you must place the following modules in the Natural system library and delete them from the NSPOOL library:

```

SPSEC01N
SECERROR *
NSCDAU *
NSCDU *
NSCLU-G *
NSCLU-M *
SNAAREXT *
SNAASEXT *
SNAUREXT *
SNAUSEXT *

```

* not required for Natural Security version 2.2.8 and above

4. Set the parameter MEMRES in the NATPARM module to 10k to allow Natural to load the program at execution time.
5. The Natural startup application (for example, NAT22) must be cataloged with ULIB as PV (privileged).

Step 16: Install the POSIX Server

The Telnet tn3270 support, HTTP server and LPD print server support of Com-plete 6.3 require the POSIX runtime to be active.

All load modules used by the POSIX runtime are contained in APSvrs.LDnn which is mandatory for Com-plete 6.3 even if you do not plan to use any POSIX-based components.

1. To activate the POSIX runtime, add the following SERVER statement to your SYSPARMs:

```
SERVER=( POSIX , PAENKERN )
```

Note:

Servers are started in the order in which their SERVER= statements are found in the SYSPARMs. The servers OPERATOR, POSIX, COMPLETE must be defined in this order.

2. POSIX runtime parameters can be specified directly in the Com-plete SYSPARM member, or in a separate member concatenated to the Com-plete SYSPARMs in the startup JCL procedure. The POSIX runtime may require additional parameters to be set, please refer to the *SMARTS Installation and Operations* for a detailed description of those parameters. In particular, the above mentioned TCP/IP based Com-plete components need to know which TCP/IP stack to use. This definition must be made using the following POSIX runtime parameter:

```
CDI_DRIVER=('tcpip,PAAXSOCK [,...]' )
```

The HTTP server requires an additional CDI driver definition:

```
CDI_DRIVER=('cgistdio,HAANPCGI' )
```

Note:

Servers are started in the order in which their SERVER= statements are found in the SYPARMs. The servers OPERATOR, POSIX, COMPLETE must be defined in this order.

Step 17: Activate the TELNET Server

Prerequisite: the POSIX server must be installed.

- The TELNET server is activated by simply assigning an IP port number for it. Contact the IP administrator of your host for a port number, and specify it using the Com-plete startup parameter TELNETPORT=port_number.

No SERVER= statement is required for the TELNET server.

Step 18: Install the HTTP Server

Install the HTTP server as described in the *HTTP Server* documentation.

In general, the following steps are necessary:

1. Make sure the POSIX runtime is setup properly, including the CDI drivers (see above).
2. Add a SERVER statement to your SYPARMs:

```
SERVER=(HTTP,HAENSERV,CONF=/my/parmlib/haanconf) .
```

Note that this server definition must be placed behind the definitions of the servers OPERATOR, POSIX, and COMPLETE in the SYSPARMs.

3. Use the sample module HAANCONF from the HTTP server source library as a pattern to setup your own configuration. This is the place where you specify, for example, the port number your HTTP server will be using.

Startup Procedure

Com-plete can be run as a started task or as a job. In order to start Com-plete, proceed as follows:

Step 1: Modify Procedure COMJCL

- Review sample procedure COMJCL in the source library for correct data set names. Then either copy the procedure to a procedure library to be started, or use it as a basis for a job to be submitted.

Note:

For more information on the data sets and their purpose, see the Com-plete System Programming documentation.

Step 2: Initialize Com-plete for the First Time

- Start Com-plete either by submitting the created job or starting the created procedure. Monitor the startup messages carefully for any signs of problems. When the initialization is complete, a message to that effect is written to the operator console.

A message is also sent to the console to notify you that the VTAM and ACCESS interfaces have initialized.

Step 3: Log On

- Log on through VTAM to the specified VTAM application name. The Com-plete ULOG menu is displayed, prompting you for user ID and password. Logon to Com-plete using user ID SAGADMIN and password ADMIN.

This user ID has a definition on the Com-plete system data container with the required authorization to define more user IDs to the system using the user ID maintenance facility (see the Com-plete Utilities documentation).

After defining other user IDs, you are recommended to change the password for user SAGADMIN and use this user ID in emergency cases only.

What Next?

Com-plete is now up and ready for work. How you continue depends on whether you have installed Com-plete for the first time or whether you are migrating from a previous version.

- If you have installed Com-plete for the first time, no further migration is necessary. You can continue with customization steps described in the System Programming documentation and the Complete Utilities documentation.
- If you are migrating from a previous version of Com-plete, see *Migration*.

VSE Installation

The installation steps described in this section must be followed in sequence in order to install Complete successfully. You are recommended to use the provided examples up to successful initialization in order to ensure that the installation was successful. Once a functional system is available, you can start customizing the product as required.

All sample JCL referred to in the installation steps is illustrated in *VSE Sample JCL*.

This document covers the following topics:

- The Installation Tape
 - Installation Steps
 - Startup Procedure
 - What Next?
-

The Installation Tape

Tape Contents

The installation tape is created using IBM standard labels with volume serial number COMvrs. The tape can be ordered in any format supported by Software AG.

File Nr.	File Name	Description / Remarks
1	SMT111.TABS	SMA tables
2	COMvrs.BASE	System file
3	COMvrs.USER	User ID file
4	COMvrs.CTLG	Catalog file
5	COMvrs.NS22	NSPOOL INPL data set
6	COMvrs.LIBR	Library data set in LIBR backup format. Rewind the tape and use the number in the first column to operate the tape using the statement // MTC FSF ...
7	APSVrs.LIBR	Library data set in LIBR backup format. Rewind the tape and use the number in the first column to operate the tape using the statement // MTC FSF ...
8	APSVrs.Lnnn	Most current patch level of APSvrs.LIBR
9	HTPvrs.LIBR	Library data set in LIBR backup format. Rewind the tape and use the number in the first column to operate the tape using the statement // MTC FSF ...
10	HTPvrs.INPL	HTTP server INPL data set
11	HTPvrs.Lnnn	Most current patch level of HTPvrsLIBR
12	HTPvrs.Inn	Most current patch level of HTPvrsINPL

Notes:

1. The files may not be on the installation tape in the order shown above: please consult the report of your tape creation system.
2. The notation *vrs* stands for current version, release and maintenance level.
3. There may be no patch level libraries for a component if this is a new release of the respective component.

Copying Contents of the Tape to Disk**Step 1: Unload the Tape**

- Copy the files on the installation tape to disk using the LIBR utility for the Com-plete library. Refer to *VSE Sample JCL* for sample JCL (JCLINST1).

Check for a README file on the delivered source. If a README file exists, it may contain important information concerning the installation and migration procedure.

Note:

Restore of the APSvrs.LIBR dataset is mandatory for Com-plete 6.3.

Installation Steps

Step 1: Create the Com-plete User Data library

- The SAGLIB.COMUSER library contains installation-dependent modules and user programs. This sublibrary is recommended, since future maintenance to the system may replace the private distribution libraries completely, thereby destroying any user modules in that library.

Create the library using the sample job JCLINST2 (Sublib=SAGLIB.COMUSER).

Step 2: Allocate and Initialize the Com-plete VSAM System Data Containers

- Only for a new installation:

Allocate, initialize and load the Com-plete system information data sets by modifying sample job JCLINST4 to reflect appropriate size and placement of the files. Please refer to the Com-plete Files and Associated User Files in the *Com-plete System Programming* documentation for more information on this VSAM file.

Step 3: Allocate and Initialize the Capture Data Set(s)

- Required only if capture is used at your installation.

Com-plete capture data sets are defined as VSAM data sets in the current version. Sample job JCLINST5 provides JCL to allocate and initialize two capture data sets. You must modify the job to reflect appropriate size and placement of the capture data sets.

Note:

In VSE, the Com-plete account records (SMF-records) are written to the capture data set.

Step 4: Allocate and Initialize the Com-plete SD Dataset

- The Com-plete SD dataset must be defined as a VSAM dataset in the current version. For detailed information on this file, please refer to the Com-plete Files and Associated User Files in the *Com-plete System Programming* documentation. To allocate and initialize the SD dataset, modify sample job JCLINST6 to reflect your requirements and run this job.

Step 5: Allocate and Initialize the Com-plete VSAM Spool Data Set

- Required.

The Com-plete spool data set reside on a VSAM file. To allocate and initialize this file, modify sample job JCLINST7 to reflect appropriate size and placement of the data set. Then run the job.

Step 6: Allocate and Initialize the Com-plete Dump File

- Required.

The Com-plete Dump File data set must be defined as a VSAM dataset. For detailed information on this file, refer to the Com-plete System Programming documentation. To allocate and initialize the Dump File dataset, modify sample job JCLINST8 to your requirements and run the job.

Step 7: Update VTAM Definitions

- Required only if the Com-plete interface with VTAM is to be used.

To enable the Com-plete VTAM interface, you must define an ACB to VTAM in the PRD2.CONFIG data set (MEMBER=VTMAPPL.B). The following sample definition generates an ACB called "COM46" with the necessary privileges for Com-plete:

```
VBUILD TYPE=APPL
COM46  APPL      AUTH=(ACQ,PASS)
```

To use this feature, you must specify the Com-plete system parameter =COM46.

Please refer to VTAM documentation for more information.

Step 8: Install the Com-plete Batch Online Interface

Required.

The Com-plete BATCH utilities and migration jobs require ACCESS when performing some or all of their functions.

Batch applications that use Com-plete services communicate with the target Com-plete via ACCESS. This communication is based on an Adabas SVC (ACCESS SVC) and a DBID (ACCESS NODE ID).

The Com-plete BATCH interface module loads a module with name ACSTAB and search this module for an entry name BATCH.

1. Modify the sample ACSTAB module on the source data set to reflect the system requirements for SVC and NODE ID.
2. Assemble and linkedit the module into the SUB=SAGLIB.COMUSER library. Modify sample job JCLINSTF to do this.

Note that the DBID chosen may be greater than 255 and that the value of both DBID and SVC must correspond to the values of ACCESS-ID and ACCESS-SVC specified in the SYSPARM member.

Step 9: Set the Com-plete System Parameters (SYSPARM)

- Sample job COMJCVSE refers to a set of Com-plete parameters in COMPARM.COMTEST1. Copy them to the library SAGLIB.COMUSER and ensure that the following parameters reflect your installation requirements:

PATCHAR	Choose a unique single character value for your installation.
VTAMAPPL	Must be changed to the ACB name you have set up in installation step <i>Update VTAM Definitions</i> .
ACCESS-SVC	Must be set to the Adabas SVC number (see installation step <i>Install the Com-plete Batch Online Interface</i>).
ACCESS-ID	Must be set to the database ID (see installation step <i>Install the Com-plete Batch Online Interface</i>).

The following two statements are mandatory in Com-plete 6.3:

```
SERVER=( OPERATOR , TLINOPER )
SERVER=( COMPLETE , TLINCOMP )
```

The SERVER statements for Console, Natural and Power must be put after the Com-plete server.

Step 10: Add VSE/POWER Interface

- Add the following statement after the SERVER statement for COMPLETE:

```
SERVER=( POWER , TLINPOW0 )
```

Step 11: Install the Com-plete High Level Language Interface

- Optional.

Applications that use Com-plete functions must be linked with the Com-plete high level language interface modules. These modules provide a standard interface between the application and Com-plete.

These modules are provided on the supplied load data set. If you are migrating from a previous version of Com-plete, some applications may need to be relinked with the new version of the HLLI modules. For details, see *Migration*.

Step 12: Install the UDEBUG Environment

Optional.

Com-plete provides an online debugging utility (UDEBUG). In order to establish the working environment, proceed as follows:

1. Allocate two UDEBUG profile and symbol datasets by running job JCLINST1 and include these datasets in the Com-plete startup procedure.
2. Include the list of required UDEBUG residentpage modules in your SYSPARM member. A list of these modules is given in member DBUGSAMP in the supplied source datasets.

The UDEBUG facility is described in detail in the Com-plete Utilities documentation.

Step 13: Installing the Extended Console Server (UQ M Function)

- Specify a SERVER sysparm for the Console Server:

Syntax:

```
SERVER=( name , TLINCONS , slots , consname , hcset )
```

where:

name	is a unique server name within Com-plete
TLINCONS	is the name of the server initialization program
slots	specify the number of messages held in the incore table
consname	is the console name for MCSOPER Macro. It must be unique in the system.
hcset	specifies wether the hardcopy set is to be received by this console. Default is N.

Example:

```
SERVER=(CONSOLE,TLINCONS,2000,COMP51A,Y)
```

Step 14: Install NSPOOL

Optional.

The printout spooling facility NSPOOL is described in the Com-plete Application Programming documentation. The required dataset for running NSPOOL is on the Com-plete installation tape and is loaded as part of the Com-plete installation. The dataset name is COMvrs.NS22.

1. Run an INPL of NSPOOL.
2. If NSPOOL is to be used in a Natural environment running under Natural Security, you must stow the following modules in the Natural Security system library; otherwise, proceed to the next point.

```
SPSE01-N,
SPSE02-S,
SPSE01-&,
SPSE02-&,
SPSE03-&
SPAPPL-N *
SPUSER-N *
```

* required only for Natural Security version 2.2.8 and above.

See also the section *Default Authorization* below.

3. With Natural 2.2, you must place the following modules in the Natural system library and delete them from the NSPOOL library:

```
SPSEC01N
SECERROR *
NSCDAU *
NSCDU *
NSCLU-G *
NSCLU-M *
SNAAREXT *
SNAASEXT *
SNAUREXT *
SNAUSEXT *
```

* not required for Natural Security version 2.2.8 and above

4. Set the parameter MEMRES in the NATPARM module to 10k to allow Natural to load the program at execution time.
5. The Natural startup application (for example, NAT22) must be cataloged with ULIB as PV (privileged).

Step 15: Install the POSIX Server

The Telnet tn3270 support, HTTP server and LPD print server support of Com-plete 6.3 require the POSIX runtime to be active.

All load modules used by the POSIX runtime are contained in APSvrs.LOAD which is mandatory for Com-plete 6.3 even if you do not plan to use any POSIX-based components.

1. To activate the POSIX runtime, add the following SERVER statement to your SYSPARMs:

```
SERVER=(POSIX,PAENKERN)
```

Note:

Servers are started in the order in which their SERVER= statements are found in the SYPARMs. The servers OPERATOR, POSIX, COMPLETE must be defined in this order.

2. POSIX runtime parameters can be specified directly in the Com-plete SYSPARM member, or in a separate member concatenated to the Com-plete SYSPARMs in the startup JCL procedure. The POSIX runtime may require additional parameters to be set, please refer to the *SMARTS Installation and Operations* for a detailed description of those parameters. In particular, the above mentioned TCP/IP based Com-plete components need to know which TCP/IP stack to use. This definition must be made using the following POSIX runtime parameter:

```
CDI_DRIVER=('tcpip,PAAXSOCK [,...]')
```

The HTTP server requires an additional CDI driver definition:

```
CDI_DRIVER=('cgistdio,HAANPCGI')
```

Note:

Servers are started in the order in which their SERVER= statements are found in the SYPARMs. The servers OPERATOR, POSIX, COMPLETE must be defined in this order.

If your TCP/IP connection uses an ID other than 00, insert the line

```
//OPTION SYSPARM='nn'
```

Step 16: Activate the TELNET Server

Prerequisite: the POSIX server must be installed.

- The TELNET server is activated by simply assigning an IP port number for it. Contact the IP administrator of your host for a port number, and specify it using the Com-plete startup parameter TELNETPORT=port_number.

No SERVER= statement is required for the TELNET server.

Step 17: Install the HTTP Server

Install the HTTP server as described in the *HTTP Server* documentation.

In general, the following steps are necessary:

1. Make sure the POSIX runtime is setup properly, including the CDI drivers (see above).
2. Add a SERVER statement to your SYSPARMs:

```
SERVER=(HTTP,HAENSERV,CONF=/lib/sublib/haanconf.j).
```

Note that this server definition must be placed behind the definitions of the servers OPERATOR, POSIX, and COMPLETE in the SYSPARMs.

3. Use the sample module HAANCONF from the HTTP server source library as a pattern to setup your own configuration. This is the place where you specify, for example, the port number your HTTP server will be using.

Startup Procedure

Com-plete can run in any partition. If it's started in a dynamic partition you need Adabas with ESA features for ACCESS support.

Step 1: Modify Procedure COMJCVSE

- Review sample job COMJCVSE in the source library for correct data set names.

Note:

For more information on the data sets and their purpose, see the Com-plete System Programming documentation.

Step 2: Initialize Com-plete System Intercept

- In VSE environments, communication between the Com-plete nucleus and the user program is established using SVC 200.

The program COMSIP is used to load the intercept module into the SVA and to put it into the chain of VSE's First Level Interrupt Handlers anchored in lowcore. Additional information about the intercept module is stored into CRSVATBL, a module which must have been loaded into the SVA using SET SDL prior to running COMSIP.

Executing the COMSIP Program

```
* $$ JOB    JNM=COMSIP,DISP=D,CLASS=?
* $$ LST    DISP=D,CLASS=A
// JOB      COMSIP    initialize Com-plete system adapter V51
// ASSGN    SYSLST,IGN
// DLBL     SAGLIB,'.....LIBRARY'
// EXTENT   ,vvvvvv
// LIBDEF   PHASE,SEARCH=SAGLIB.APSvrs,TEMP
// UPSI     00000000
// EXEC     COMSIP,SIZE=AUTO
```

```
// ASSGN  SYSLST,UA
/*
/ &
* $$ EOJ
```

Important:

This step should never run while another SAG product running on SMARTS (APS) is up. CRSVATBL and COMSIP must be loaded from the same library. The modules are downward compatible and can be used by every currently supported version of Com-plete.

If you have software from CA and/or Macro 4 installed, this step must run AFTER the initialization of the CA system and BEFORE the initialization of Macro 4.

In a multi-processor environment, this step can only run before the second CPU is started.

Insert the following into the ASI BG JCL procedure according to the above mentioned requirements before the START of the POWER partition so that the Com-plete SVC intercept will be installed automatically during each IPL.

```
// DLBL      SAGLIB,'.....LIBRARY'
// EXTENT   SYS010,vvvvvv
// ASSGN    SYS010,DISK,VOL=vvvvvv,SHR
// LIBDEF   PHASE,SEARCH=SAGLIB.APSvrs,TEMP
SET SDL
CRSVATBL,SVA
/*
// UPSI     00000000
// EXEC     COMSIP,SIZE=AUTO
```

Step 3: Initialize Com-plete for the First Time

- Start Com-plete by submitting the created job. Monitor the startup messages carefully for any signs of problems. When the initialization is complete, a message to that effect is written to the operator console.

A message is also sent to the console to notify you that the VTAM interface has initialized.

Step 4: Log On

- Log on through VTAM to the specified VTAM application name. The Com-plete ULOG menu is displayed, prompting you for user ID and password. Log on to Com-plete using user ID SAGADMIN and password ADMIN.

This user ID has a definition on the Com-plete system data container with the required authorization to define more user IDs to the system, using the user ID maintenance facility described in the Com-plete Utilities documentation.

After defining other user IDs, you are recommended to change the password for user SAGADMIN and use this user ID in emergency cases only.

What Next?

Com-plete is now up and ready for work. How you continue depends on whether you have installed Com-plete for the first time or whether you are migrating from a previous version.

- If you have installed Com-plete for the first time, no further migration is necessary. You can continue with customization steps described in the System Programming documentation and the Complete Utilities documentation.
- If you are migrating from a previous version of Com-plete, see *Migration*.

Migration

If you are upgrading from version 6.1 to version 6.3, then all you need to do is replace your APS and COM datasets with the new ones.

This section describes the steps required to migrate from Com-plete 5.1 to Version 6.3.

It is assumed that your current Com-plete version is 5.1.x.

This chapter covers the following topics:

- Changes to the Start-up Procedure
 - Changes to the SYSPARMS
 - COMSYS Data Containers
 - Reducing the Thread Region Size for most Com-plete Utilities
-

Changes to the Start-up Procedure

Add the SMARTS Load Library

While previous versions of Com-plete used to come in one load library, Com-plete 6 is delivered as the application server SMARTS which can be shared with other Software-AG products, and a Com-plete-specific part. Add the application server load library `APSVrs.LDnn` to your COMPLIB library chain behind the Com-plete load library.

Remove the COMPINIT Load Library Chain

If you were using a separate COMPINIT chain, Software AG recommends that you remove it.

Remove COMSYS2

The Com-plete error message texts are now delivered in load module format; the VSAM file `COMSYS.MESSAGES` (DD name / DLBL name `COMSYS2`) is no longer used. Remove this statement from your JCL startup procedure, keeping `COMSYS1`, `COMSYS3`, and `COMSYS4` unchanged.

Log File APSLOG (z/OS-like Systems Only)

This file is used by Com-plete to log all the messages that previously went to the system console, of course including those that still do go to the console. If you do not specify `APSLOG`, then Com-plete allocates it dynamically as

```
//APSLOG DD SYSOUT=X
```

Changes to the SYSPARMs

In the new application server architecture, Com-plete must be defined as a server running on SMARTS. Also, the operator communication runs as a separate server. Please add the following two lines to your SYSPARMs:

```
SERVER=( OPERATOR , TLINOPER )
SERVER=( COMPLETE , TLINCOMP )
```

It is now possible to wrap SYSPARM lines (after a comma within parentheses). This allows you to write complex parameter definitions like THREAD-GROUP in a more readable way, but it implies an enhanced syntax checking. As a consequence, parameter values containing special characters must now be enclosed in apostrophes. For example, RECALLCHAR== must be changed into RECALLCHAR='=', otherwise an error will be indicated.

COMSYS Data Containers

Software AG recommends that you backup your COMSYS files before running this upgrade program.

To upgrade your COMSYS Data Containers, run the BATCH program CSYSUPGR using sample job JCLINSTD. Note that there is no COM*vrs*.ALLMSGs file on the installation tape because the messages are now delivered in load module format. Message texts that you have translated into another language will be converted and migrated into COMSYS4 if you define your COMSYS.MESSAGES data set as COMSYS2 in the JCL of the upgrade job.

Reducing the Thread Region Size for most Com-plete Utilities

The following utilities have been modified to be able run above the 16MB line:

CSPOOL, UBATCH, UCMND, UCOPY, UCTRL, UDEF, UDISP, UDUMP, UEBCB, UFLEX, UGLIB, UHELP, UHELPM, ULIBID, ULOG, ULOGM, ULOW, UM, UMATH, UMSG, UNUMR, UPF, UPFKS, UPROF, UPWD, USCHC, USPOOL, USTACK, UTIBGR, UTMMSG, UTRAC1, UUP, UZAPS.

The COMSYS upgrade program CSYSUPGR changes the region size settings for these applications to 0 or 4K, so you must use thread extensions above the line (THSIZEABOVE=200 at least) in order to be able to run any of these applications. Software AG recommends that you adjust any existing user exits for these utilities so they can also run above the line. If this appears unfeasible, then use ULIB to increase the region size again for the utilities affected. It is planned to move all remaining Com-plete utilities above the line in one of the next releases, so it is a good idea to make plans for adjusting your user exits for those utilities also.

In case you cannot log on to Com-plete after the upgrade because of insoluble problems with your ULOG exit routine ULOGX1, then restore the ULIB entry for ULOG. Using BATCH utility TULIB is not an option in this case because it also involves ULOG, but you can use standard tools like IDCAMS:

```
REPRO INDATASET (comsys.catalog.backup) OUTDATASET(comsys.catalog) -
FROMKEY(X'002000E4D3D6C740404040') -
TOKEY(X'002000E4D3D6C740404040') -
REPLACE
```

z/OS Sample JCL

This chapter contains sample JCL of the jobs referred to in the installation procedure.

- A. Download Datasets from the Supplied Tape
- B. JCLINST1
- C. JCLINST2
- D. JCLINST3
- E. JCLINST4
- F. JCLINST5
- G. JCLINST7
- H. JCLINST8
- I. JCLINST9
- J. JCLINSTB
- K. JCLINSTC
- L. JCLINSTE

A. Download Datasets from the Supplied Tape

```
//JOBNAME ....valid installation job card
/**
/** This job downloads the datasets, as supplied on the
/** &cmon installation tape, to disk for use during the
/** installation procedure.
/**
/**IEBCOPY EXEC PGM=IEBCOPY
/**SYSPRINT DD SYSOUT=*
/**SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(3,1))
/**SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(3,1))
/**
/**SOURCEI DD DSN=COMvrs.SRCE, note 2
/** UNIT=uuuu, note 1
/** LABEL=(1,SL),
/** VOL=(,RETAIN,SER=COMvrs), note 2
/** DISP=(OLD,PASS)
/**
/**SOURCEO DD DSN=COM.SOURCE,
/** UNIT=uuuu, note 4
/** VOL=SER=vvvvvvv, note 3
/** SPACE=(CYL,(7,1,50)),
/** DISP=(NEW,CATLG,DELETE),
/** DCB=(DSORG=PO,RECFM=FB,BLKSIZE=3120,LRECL=80)
/**
/**LOADI DD DSN=COMvrs.LOAD, note 2
```

```

//          UNIT=uuuu,                note 1
//          LABEL=(2,SL),
//          VOL=(,RETAIN,REF=*.SOURCEI),
//          DISP=(OLD,PASS)
//*
//LOADO    DD DSN=COM.LOAD,
//          UNIT=uuuu,                note 4
//          VOL=SER=vvvvvv,          note 3
//          SPACE=(CYL,(18,1,200)),
//          DISP=(NEW,CATLG,DELETE),
//          DCB=(DSORG=PO,RECFM=U,BLKSIZE=6447)
//SMDOC1   DD DSN=COMvrs.INFO,        note 2
//          UNIT=uuuu,                note 1
//          LABEL=(5,SL),
//          VOL=(,RETAIN,SER=COMvrs), note 2
//          DISP=(OLD,PASS)
//SOURCEO  DD DSN=COM.INFO,
//          UNIT=uuuu,                note 4
//          VOL=SER=vvvvvv,          note 3
//          SPACE=(CYL,(3,1,50)),
//          DISP=(NEW,CATLG,DELETE),
//          DCB=(DSORG=PO,RECFM=FB,BLKSIZE=3120,LRECL=80)
//SYSIN    DD *
COPY INDD=SOURCEI,OUTDD=SOURCEO

COPYMOD INDD=LOADI,OUTDD=LOADO
COPY SNDD=SMDOC1,OUTDD=SMDOC0
/*

```

Notes:

1. The "uuuu" here must be set to an installation unit name relating to the tape device to be used to load the tape.
2. "vrs" in these cases relates to the Version, Release and SM Level of COM-LETE being installed, for example, the tape volser for COM-LETE 6.3.1 would be called COM631.
3. The volser "vvvvvv" should be changed to the volume serial number upon which you wish the dataset to be downloaded.
4. The unit "uuuu" must be changed to a unit name which is valid for the specified volume serial number.

B. JCLINST1

```

//JOBNAM1  JOB.... JOB CARD INFORMATION ..
//*
/** THIS JOB COPIES THE MEMBER TLINOS FROM THE DISTRIBUTED
/** APS LOAD LIBRARY INTO AN APF AUTHORISED LIBRARY.
/**
/** THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB:-
/**
/** 1. INSERT A VALID JOBCARD.
/** 2. CHANGE THE DATASET NAMES TO SUIT YOUR SITE'S REQUIREMENTS.
/**
//COPY     EXEC   PGM=IEBCOPY
//SYSPRINT DD   SYSOUT=*
//IN       DD     DISP=SHR,DSN=APS.LDnn
//OUT      DD     DISP=SHR,DSN=ANY.APF.AUTHORISED.LIBRARY

```

```
//SYSIN DD *
COPYMOD I=((IN,R)),O=OUT
S M=TLINOS
/*
```

C. JCLINST2

```
//JOBNAM2 JOB.... JOB CARD INFORMATION ..
/*
/* THIS JOB ALLOCATES AND INITIALISES THE NON-VSAM AND USER DATA SETS.
/*
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB:-
/*
/* 1. INSERT A VALID JOBCARD.
/* 2. CHANGE PREFIX TO SUIT YOUR SITE'S REQUIREMENTS
/* 3. CHANGE UUUU TO THE REQUIRED UNIT.
/* 4. CHANGE VVVVVV TO THE REQUIRED VOLSER(S).
/*
//ALLOCATE EXEC PGM=IEFBR14
/* ROLL DATASET
/*
/* NOTE: IF YOU DECIDE TO USE VSAM ROLL DATASET(S), COMMENT OUT ALL
/* LINES OF THE R1 DD STATEMENT AND IN ADDITION TO THIS JOB,
/* RUN JOB JCLINST6.
/*
/* USER SOURCE FILE
//USERSOUR DD DSN=PREFIX.USER.SOURCE,
// DISP=(NEW,CATLG,DELETE),UNIT=UUUU,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120,DSORG=PO),
// SPACE=(CYL,(2,,28)),
// VOL=SER=VVVVVV
/* USER LOAD LIBRARY
//USERLOAD DD DSN=PREFIX.USER.LOAD,
// DISP=(NEW,CATLG,DELETE),UNIT=UUUU,
// DCB=(RECFM=U,BLKSIZE=6447,DSORG=PO),
// SPACE=(CYL,(10,,28)),
// VOL=SER=VVVVVV
/* USER MAP LIBRARY
//MAPS DD DSN=PREFIX.MAPS,
// DISP=(NEW,CATLG,DELETE),UNIT=UUUU,
// DCB=(RECFM=U,BLKSIZE=6447,DSORG=PO),
// SPACE=(CYL,(2,,28)),
// VOL=SER=VVVVVV
```

D. JCLINST3

```
//JOBNAM3 JOB.... JOB CARD INFORMATION ..
/*
/* THIS JOB ALLOCATES AND INITIALISES THE SYSTEM DATA CONTAINERS
/*
/* NOTE THAT YOU MAY HAVE TO RUN JOB JCLINSTD TO UPDATE THE SYSTEM
/* DATASET TO THE LATEST SM LEVEL AFTER RUNNING THIS JOB.
/*
/* PERFORM THE FOLLOWING CHANGES BEFORE RUNNING THIS JOB:
/*
/* 1. INSERT A VALID JOBCARD.
/* 2. CHANGE PREFIX TO SUIT YOUR SITE'S REQUIREMENTS
/* 3. CHANGE VVVVVV TO THE REQUIRED VOLSER.
/* 4. FOR SYSPLEX-WIDE SHARED DATASETS, ADD LOG(NONE) TO EACH CLUSTER.
/*
```

```

//ALLBASE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE (PREFIX.COMSYS.BASE)
DEFINE CLUSTER(NAME(PREFIX.COMSYS.BASE) -
  VOLUME(VVVVVV) -
  INDEXED -
  RECORDSIZE(4089 4089) -
  RECORDS(512 64) -
  KEYS(16 0) -
  CISZ(4096))-
DATA(NAME(PREFIX.COMSYS.BASE.DATA) -
  SHAREOPTIONS(2,3)) -
INDEX(NAME(PREFIX.COMSYS.BASE.INDX) -
  SHAREOPTIONS(2,3))
/*
//ALLUSER EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE (PREFIX.COMSYS.USERDEF)
DEFINE CLUSTER(NAME(PREFIX.COMSYS.USERDEF) -
  VOLUME(VVVVVV) -
  INDEXED -
  RECORDSIZE(2041 2041) -
  RECORDS(4096 512) -
  KEYS(16 0) -
  CISZ(4096))-
DATA(NAME(PREFIX.COMSYS.USERDEF.DATA) -
  SHAREOPTIONS(2,3)) -
INDEX(NAME(PREFIX.COMSYS.USERDEF.INDX) -
  SHAREOPTIONS(2,3))
/*
//ALLCATLG EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE (PREFIX.COMSYS.CATALOG)
DEFINE CLUSTER(NAME(PREFIX.COMSYS.CATALOG) -
  VOLUME(VVVVVV) -
  INDEXED -
  RECORDSIZE(313 313) -
  RECORDS(512 64) -
  KEYS(16 0) -
  CISZ(4096))-
DATA(NAME(PREFIX.COMSYS.CATALOG.DATA) -
  SHAREOPTIONS(2,3)) -
INDEX(NAME(PREFIX.COMSYS.CATALOG.INDX) -
  SHAREOPTIONS(2,3))
/*
//*
//INIT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//COMSYS1 DD DSN=PREFIX.COMSYS.BASE,DISP=SHR
//COMSIN1 DD DSN=PREFIX.SYSTEM.BASE,DISP=SHR
//COMSYS3 DD DSN=PREFIX.COMSYS.USERDEF,DISP=SHR
//COMSIN3 DD DSN=PREFIX.SYSTEM.USERDEF,DISP=SHR
//COMSYS4 DD DSN=PREFIX.COMSYS.CATALOG,DISP=SHR
//COMSIN4 DD DSN=PREFIX.SYSTEM.CATALOG,DISP=SHR
//SYSIN DD *

```

```

REPRO INFILE(COMSIN1) OUTFILE(COMSYS1)
REPRO INFILE(COMSIN3) OUTFILE(COMSYS3)
REPRO INFILE(COMSIN4) OUTFILE(COMSYS4)
/*

```

E. JCLINST4

```

//JOBNAM4 JOB.... JOB CARD INFORMATION ..
/*
/* THIS JOB ALLOCATES AND INITIALISES THE CAPTURE DATA SETS.
/*
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB
/*
/* 1. INSERT A VALID JOBCARD.
/* 2. CHANGE PREFIX TO SUIT YOUR SITE'S REQUIREMENTS
/* 3. CHANGE VVVVVV TO THE REQUIRED VOLSER(S).
/*
//ALLOC EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE PREFIX.VSAM.CAPTUR1
DELETE PREFIX.VSAM.CAPTUR2
DEFINE CLUSTER -
    ( NAME (PREFIX.VSAM.CAPTUR1) -
      NONINDEXED -
      SHAREOPTIONS (1) -
      RECORDSIZE (0032 4096) -
      SPEED REUSE ) -
  DATA -
    ( NAME (PREFIX.VSAM.CAPTUR1.DATA) -
      CISZ (8192) -
      VOLUMES (VVVVVV) -
      CYLINDERS (3 0))
DEFINE CLUSTER -
    ( NAME (PREFIX.VSAM.CAPTUR2) -
      NONINDEXED -
      SHAREOPTIONS (1) -
      RECORDSIZE (0032 4096) -
      SPEED REUSE ) -
  DATA -
    ( NAME (PREFIX.VSAM.CAPTUR2.DATA) -
      CISZ (8192) -
      VOLUMES (VVVVVV) -
      CYLINDERS (3 0))
/*
//INIT EXEC PGM=TUSACAPT,PARM='5000'
//STEPLIB DD DSN=PREFIX.LOAD,DISP=SHR
//CAPTUR1 DD DSN=PREFIX.VSAM.CAPTUR1,DISP=SHR
//CAPTUR2 DD DSN=PREFIX.VSAM.CAPTUR2,DISP=SHR
/*

```

F. JCLINST5

```

//JOBNAM5 JOB.... JOB CARD INFORMATION ..
/*
/* THIS JOB ALLOCATES AND INITIALISES THE SPOOL DATA SET.
/*
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB
/*
/* 1. INSERT A VALID JOBCARD.

```

```

/* 2. CHANGE PREFIX TO SUIT YOUR SITE'S REQUIREMENTS
/* 3. CHANGE VVVVVV TO THE REQUIRED VOLSER.
/*
/*
//ALLOC EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE PREFIX.SPOOL
DEFINE CLUSTER -
    ( NAME (PREFIX.SPOOL) -
      NUMBERED -
      SHAREOPTIONS (2 3) -
      RECORDSIZE (2040 2040) -
      SPEED REUSE ) -
  DATA -
    ( NAME (PREFIX.SPOOL.DATA) -
      CISZ (2048) -
      VOLUMES (VVVVVV) -
      CYLINDERS (5 0))
/*
//INIT EXEC PGM=TUMSUTIL
//STEPLIB DD DSN=PREFIX.LOAD,DISP=SHR
//COMSPL DD DSN=PREFIX.SPOOL,DISP=SHR
//SYSLST DD SYSOUT=*
//SYSIN DD *
INIT RECS=1350
/*

```

G. JCLINST7

```

//JOBNAM7 JOB.... JOB CARD INFORMATION ..
/*
/* THIS JOB ALLOCATES AND INITIALISES THE Com-plete JES3 DATA SET.
/*
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB:-
/*
/* 1. INSERT A VALID JOBCARD.
/* 2. CHANGE PREFIX TO SUIT YOUR SITE'S REQUIREMENTS
/* 3. CHANGE UUUU TO THE REQUIRED UNIT.
/* 4. CHANGE BLKSIZE TO THE REQUIRED BLOCKSIZE.
/* 5. CHANGE CMDS TO THE REQUIRED NUMBER OF COMMANDS.
/* 6. CHANGE VVVVVV TO THE REQUIRED VOLSER.
/*
/* VALID VALUES OF BLKSIZE AND CMDS FOR YOUR DISKTYPE CAN BE OBTAINED
/* FROM THE FOLLOWING TABLE:-
/*
/*          UNIT  CMDS  BLKSIZE
/*          ----  ----  -
/*          2314   39   7280
/*          3330   61  12960
/*          3340   34   8320
/*          3350   72  19040
/*          3375   76  32720
/*          3380   83  32720
/*          3390   78  32720
/*
//J3ALLOC EXEC PGM=IEFBR14
//J3UQ DD DSN=PREFIX.JES3.UQJOBS,
//      DISP=(NEW,CATLG,DELETE),UNIT=UUUU,
//      DCB=(RECFM=U,BLKSIZE=BLKSIZE),
//      SPACE=(TRK,5,,CONTIG),

```

```

//          VOL=SER=VVVVVV
//*
//J3INIT1 EXEC PGM=IEBDG
//SYSPRINT DD SYSOUT=*
//OUTPUT DD DSN=&T1,DISP=(,PASS),UNIT=UUUU,
//          SPACE=(TRK,4),VOL=SER=VVVVVV,
//          DCB=(RECFM=U,BLKSIZE=BLKSIZE)
//SYSIN DD *
DSD OUTPUT=(OUTPUT)
CREATE QUANTITY=4,FILL=X'00'
/*
//J3INIT2 EXEC PGM=IEBDG
//SYSPRINT DD SYSOUT=*
//OUTPUT DD DSN=&T2,DISP=(,PASS),UNIT=UUUU,
//          SPACE=(TRK,1),VOL=SER=VVVVVV,
//          DCB=(RECFM=U,BLKSIZE=80)
//SYSIN DD *
DSD OUTPUT=(OUTPUT)
CREATE QUANTITY=CMDS,FILL=X'00'
/*
//J3INIT3 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DSN=&T1,DISP=(OLD,DELETE)
//          DD DSN=&T2,DISP=(OLD,DELETE)
//SYSUT2 DD DSN=PREFIX.JES3.UQJOBS,DISP=SHR

```

H. JCLINST8

```

//JOBNAM8 JOB.... JOB CARD INFORMATION ..
//*
/* THIS JOB ASSEMBLES AND LINKS THE JES3 INTERFACE MODULES.
/*
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB:-
/*
/* 1. INSERT A VALID JOBCARD.
/* 2. CHANGE PREFIX TO SUIT YOUR SITE'S REQUIREMENTS
/* 3. WHEN ASSEMBLING ON SYSTEMS WHERE THE 'HL' ASSEMBLER IS
/*     NOT AVAILABLE, CHANGE THE PROCEDURE VARIABLE 'ASMBLR' FROM
/*     'ASMA90' TO 'IEV90'.
/*
//ASMJ3 PROC MAC1='PREFIX.SOURCE',
//          MAC2='SYS1.JES3MAC',
//          MAC3='SYS1.MACLIB',
//          MAC4='SYS1.AMODGEN',
//          N=TEMPNAME,XREF='XREF(SHORT)',
//          ASMBLR=ASMA90,LINECNT=55,
//          SLIB='PREFIX.SOURCE',
//          LLIB='PREFIX.USER.LOAD',
//          CLIB='PREFIX.LOAD',
//          UNIT='SYSDA',
//          CC=9,NCAL=NCAL,LSIZE='(120K,30K)'
/*
//ASM EXEC PGM=&ASMBLR,REGION=1024K,
//          PARM='LOAD,NODECK,TERM,&XREF,LINECOUNT(&LINECNT)'
//SYSLIB DD DSN=&MAC1,DISP=SHR,DCB=BLKSIZE=32000
//          DD DSN=&MAC2,DISP=SHR
//          DD DSN=&MAC3,DISP=SHR
//          DD DSN=&MAC4,DISP=SHR
//SYSUT1 DD SPACE=(CYL,(5,2)),UNIT=&UNIT,DISP=(NEW,PASS)

```

```
//SYSGO      DD      DUMMY
//SYSLIN     DD      UNIT=&UNIT,SPACE=(TRK,(30,20),RLSE),
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),DISP=(NEW,PASS)
//SYSPRINT   DD      SYSOUT=*
//SYSTEM     DD      SYSOUT=*
//SYSIN      DD      DSN=&SLIB.(&N),DISP=SHR
//*
//LKED       EXEC   PGM=IEWL,COND=(&CC,LT,ASM),
//              PARM='XREF,LIST,LET,&NCAL,SIZE=&LSIZE'
//SYSLIN     DD      DSN=*.ASM.SYSLIN,DISP=(OLD,DELETE)
//              DD      DDNAME=SYSIN
//SYSLIB     DD      DSN=&CLIB,DISP=SHR
//SYSLMOD    DD      DSN=&LLIB.(&N),DISP=SHR
//SYSPRINT   DD      SYSOUT=*
//SYSUT1     DD      SPACE=(CYL,(5,2)),DSN=*.ASM.SYSUT1,DISP=(OLD,DELETE)
// PEND
//*
//ASML1     EXEC   ASMJ3,N=IATUQJ3,LLIB='JES3.LOADLIB'
//ASML2     EXEC   ASMJ3,N=JES3SERV
//*
```

I. JCLINST9

```
//JOBNAM1    JOB.... JOB CARD INFORMATION ..
//*
//* THIS JOB LINKS THE TLINF4 MODULE FROM THE DISTRIBUTED
//* LOAD LIBRARY INTO AN APF AUTHORISED LIBRARY.
//*
//* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB:-
//*
//* 1. INSERT A VALID JOBCARD.
//* 2. CHANGE PREFIX TO SUIT YOUR SITE'S REQUIREMENTS
//* 3. CHANGE THE OUT DSN TO THE REQUIRED LIBRARY.
//*
//LINK       EXEC   PGM=IEWL,PARM='LIST,XREF,AC=1'
//SYSPRINT   DD      SYSOUT=*
//SYSUT1     DD      UNIT=SYSDA,SPACE=(CYL,(3,1))
//LINKLIB    DD      DISP=SHR,DSN=PREFIX.LOAD
//SYSLMOD    DD      DISP=SHR,DSN=ANY.APF.AUTHORISED.LIBRARY
//SYSLIN     DD      *
//              INCLUDE LINKLIB(TLINF4)
//              MODE AMODE(31),RMODE(24)
//              NAME      TLINF4(R)
//*
```

J. JCLINSTB

```
//JOBNAMB    JOB.... JOB CARD INFORMATION ..
//*
//* THIS JOB ASSEMBLES THE Com-plete JES2 SERVER MODULE.
//*
//* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB:-
//*
//* 1. INSERT A VALID JOBCARD.
//* 2. CHECK THAT THE VARIOUS DATASET NAMES ARE CORRECT
//* 3. WHEN ASSEMBLING ON SYSTEMS WHERE THE 'HL' ASSEMBLER IS
//*      NOT AVAILABLE, CHANGE THE PROCEDURE VARIABLE 'ASMBLR' FROM
//*      'ASMA90' TO 'IEV90'.
//* 4. FOR z/OS SYSTEMS BEFORE OS/390, CHANGE 'SHASMAC' TO 'HASPSRC'
```

```

/** 5. ENSURE THAT THE CMOSTYPE MACRO HAS BEEN UPDATED TO REFLECT THE
/** OPERATING SYSTEM FOR WHICH YOU ARE ASSEMBLING AS PER THE
/** INSTALLATION documentation.
/**
/**ASMJ2   PROC   MAC1='SYS1.SMPSTS' ,
/**         MAC2='SYS1.SMPSTS' ,
/**         MAC3='SYS1.SHASMAL' ,           <-- OS/390
/**         MAC4='COM.SOURCE' ,
/**         MAC4='COM.SOURCE' ,
/**         MAC5='SYS1.MACLIB' ,
/**         MAC6='SYS1.AMODGEN' ,
/**         N=TEMPNAME,XREF='XREF(SHORT)' ,
/**         ASMBLR=ASMA90,LINECNT=55 ,     <-- OS/390
/**         SLIB='COM.SOURCE' ,
/**         LLIB='COM.USER.LOAD' ,
/**         UNIT='SYSDA' ,
/**         CC=9,NCAL=NCAL,LSIZE='(228K,48K)'
/**
/**ASM     EXEC   PGM=&ASMBLR,REGION=1024K,
/**           PARM='LOAD,NODECK,TERM,&XREF,LINECOUNT(&LINECNT)'
/**SYSLIB  DD     DSN=&MAC1,DISP=SHR,DCB=BLKSIZE=32000
/**         DD     DSN=&MAC2,DISP=SHR
/**         DD     DSN=&MAC3,DISP=SHR
/**         DD     DSN=&MAC4,DISP=SHR
/**         DD     DSN=&MAC5,DISP=SHR
/**         DD     DSN=&MAC6,DISP=SHR
/**SYSUT1  DD     SPACE=(CYL,(5,2)),UNIT=&UNIT,DISP=(NEW,PASS)
/**SYSUT2  DD     SPACE=(CYL,(5,2)),UNIT=&UNIT,DISP=(NEW,PASS)
/**SYSUT3  DD     SPACE=(CYL,(5,2)),UNIT=&UNIT,DISP=(NEW,PASS)
/**SYSGO   DD     DUMMY
/**SYSLIN  DD     UNIT=&UNIT,SPACE=(TRK,(30,20),RLSE) ,
/**           DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),DISP=(NEW,PASS)
/**SYSPRINT DD    SYSOUT=*
/**SYSTEM  DD     SYSOUT=*
/**SYSIN   DD     DSN=&SLIB.(&N),DISP=SHR
/**
/**LKED    EXEC   PGM=IEWL,COND=(&CC,LT,ASM) ,
/**           PARM='AMODE=31,RMODE=ANY,XREF,LIST,LET,REUS,&NCAL,SIZE=&LSIZE'
/**SYSLIN  DD     DSN=* .ASM.SYSLIN,DISP=(OLD,DELETE)
/**         DD     DDNAME=SYSIN

/**SYSLIB  DD     DSN=&LLIB,DISP=SHR
/**SYSLMOD DD     DSN=&LLIB.(&N),DISP=SHR
/**SYSPRINT DD    SYSOUT=*
/**SYSUT1  DD     SPACE=(CYL,(5,2)),DSN=* .ASM.SYSUT1,DISP=(OLD,DELETE)
/** PEND
/**
/**ASML1   EXEC   ASMJ2,N=JES2SERV
/**

```

K. JCLINSTC

```

//JOBNAMC JOB.... JOB CARD INFORMATION ..
/**
/** THIS JOB ASSEMBLES THE Com-plete z/OS ACTIVE DISPLAY FORMAT MODULE
/**
/** THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB:-
/**
/** 1. INSERT A VALID JOBCARD.
/** 2. CHECK THAT THE VARIOUS DATASET NAMES ARE CORRECT

```

```

/** 3. WHEN ASSEMBLING ON SYSTEMS WHERE THE 'HL' ASSEMBLER IS
/** NOT AVAILABLE, CHANGE THE PROCEDURE VARIABLE 'ASMBLR' FROM
/** 'ASMA90' TO 'IEV90'.
/** 4. ENSURE THAT THE CMOSTYPE MACRO HAS BEEN UPDATED TO REFLECT THE
/** OPERATING SYSTEM FOR WHICH YOU ARE ASSEMBLING AS PER THE
/** INSTALLATION documentation.
/**
/**ASMLKED PROC MAC1='SYS1.SMPSTS',
/**          MAC2='SYS1.SMPMTS',
/**          MAC3='COM.SOURCE',
/**          MAC4='SYS1.MACLIB',
/**          N=TEMPNAME,XREF='XREF(SHORT)',
/**          ASMBLR=ASMA90,LINECNT=55,
/**          SLIB='COM.SOURCE',
/**          LLIB='COM.USER.LOAD',
/**          UNIT='SYSDA',
/**          CC=9,NCAL=NCAL,LSIZE='(228K,48K)'
/**
/**ASM EXEC PGM=&ASMBLR,REGION=1024K,
/**          PARM='LOAD,NODECK,TERM,&XREF,LINECOUNT(&LINECNT)'
/**SYSLIB DD DSN=&MAC1,DISP=SHR,DCB=BLKSIZE=32000
/**        DD DSN=&MAC2,DISP=SHR
/**        DD DSN=&MAC3,DISP=SHR
/**        DD DSN=&MAC4,DISP=SHR
/**        DD DSN=&MAC5,DISP=SHR
/**SYSUT1 DD SPACE=(CYL,(5,2)),UNIT=&UNIT,DISP=(NEW,PASS)
/**SYSUT2 DD SPACE=(CYL,(5,2)),UNIT=&UNIT,DISP=(NEW,PASS)
/**SYSUT3 DD SPACE=(CYL,(5,2)),UNIT=&UNIT,DISP=(NEW,PASS)
/**SYSGO DD DUMMY
/**SYSLIN DD UNIT=&UNIT,SPACE=(TRK,(30,20),RLSE),
/**          DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),DISP=(NEW,PASS)
/**SYSPRINT DD SYSOUT=*
/**SYSTEM DD SYSOUT=*
/**SYSIN DD DSN=&SLIB.(&N),DISP=SHR
/**
/**LKED EXEC PGM=IEWL,COND=(&CC,LT,ASM),
/** PARM='AMODE=31,RMODE=ANY,XREF,LIST,LET,&NCAL,SIZE=&LSIZE,REUS'
/**SYSLIN DD DSN=*.ASM.SYSLIN,DISP=(OLD,DELETE)
/**        DD DDNAME=SYSIN
/**SYSLIB DD DSN=&LLIB,DISP=SHR
/**SYSLMOD DD DSN=&LLIB.(&N),DISP=SHR
/**SYSPRINT DD SYSOUT=*
/**SYSUT1 DD SPACE=(CYL,(5,2)),DSN=*.ASM.SYSUT1,DISP=(OLD,DELETE)
/** PEND
/**
/**ASML1 EXEC ASMLKED,N=TTJIMVS
/**

```

L. JCLINSTE

```

//JOBNAME JOB... JOB CARD INFORMATION ..
/**
/** THIS JOB ALLOCATES AND INITIALISES THE SD DATA SET.
/**
/** THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB:-
/**
/** 1. INSERT A VALID JOBCARD.
/** 2. CHANGE PREFIX TO SUIT YOUR SITE'S REQUIREMENTS
/** 3. CHANGE VVVVVV TO THE REQUIRED VOLSER.
/** 4. CHECK THE IDCAMS PARAMETERS CISZ, RECORDSIZE, CYLINDERS.
/** 5. IF DESIRED, ENTER VALUES FOR RECORDS, SDFILES, DMPSPAC.

```

```
//*
//ALLOC EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE PREFIX.VSAM.SD
DEFINE CLUSTER -
    ( NAME (PREFIX.VSAM.SD) -
      NUMBERED -
      SHAREOPTIONS (2) -
      SPEED REUSE ) -
    DATA -
    ( NAME (PREFIX.VSAM.SD.DATA) -
      CISZ (4096) -
      RECORDSIZE (4080 4080) -
      VOLUMES (VVVVVV) -
      CYLINDERS (70))
/*
//INIT EXEC PGM=TUSDUTIL,REGION=1024K,
// PARM='INIT,RECORDS=,SDFILES=,DMPSPAC='
/*
/* RECORDS= NUMBER OF VSAM RECORDS TO BE INITIALIZED.
/* DEFAULT: MAX THAT FITS INTO EXTENTS CURRENTLY ALLOCATED
/*
/* SDFILES= DIRECTORY SIZE (MAX NUMBER OF SDFILES). DEFAULT: 500
/*
/* DMPSPAC= SPACE IN MBYTES THAT SHALL BE ASSIGNED FOR THREAD DUMPS.
/* DEFAULT: 50% OF RECORDS. MAX: 32 MBYTE (IN ANY CASE).
/* REST WILL BE USED FOR SDFILES.
/*
//STEPLIB DD DSN=PREFIX.LOAD,DISP=SHR
//COMSD DD DSN=PREFIX.VSAM.SD,DISP=OLD
//SYSPRINT DD SYSOUT=*
```

VSE Sample JCL

This chapter contains sample JCL of the jobs referred to in the installation procedure.

- A. JCLINST1
- B. JCLINST2
- C. JCLINST4
- D. JCLINST5
- E. JCLINST6
- F. JCLINST7
- G. JCLINST8
- H. JCLINST9

A. JCLINST1

```
* $$ JOB JNM=JCLINST1, ..... JECL CARD INFORMATION .....
* $$ LST DISP=D,CLASS=A
// JOB      JCLINST1      ..... JOB CARD INFORMATION .....
/*
/* THIS IS THE INSTALLATION JOB1.
/*
/* THIS JOB DOWNLOADS THE DATASETS, AS SUPPLIED ON THE
/* INSTALLATION TAPE, TO DISK FOR USE DURING THE
/* INSTALLATION PROCEDURE.
/*
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB
/*
/* 1. INSERT A VALID JECL AND JOB CARD.
/* 2. CHANGE vrs      TO THE VERSION, RELEASE AND SM LEVEL.
/* 3. CHANGE vvvvvv  TO THE REQUIRED VOLSER.
/* 4. CHANGE ttttt   TO THE REQUIRED TRACK.
/* 5. CHANGE nnnn    TO THE REQUIRED NUMBER OF TRACKS.
/* 6. CHANGE cuu     TO THE REQUIRED TAPE/CASS UNIT.
/* 7. CHANGE xx      TO THE REQUIRED FILE NUMBER.
/*
/*
// PAUSE      ..... WAIT FOR TAPE UNIT
// ASSGN      SYS006,CUU
// DLBL       SAGLIB,'.....LIBRARY'
// EXTENT     ,VVVVVV,1,0,TTTTTT,NNNN
/*
/* =====
* RESTORE SAGLIB.COMvrs      DISTRIBUTION COM-LETE LIBRARY
*                               SOURCE, OBJ and PHASE
/* =====
/*
// MTC        REW,SYS006
// MTC        FSF,SYS006,xx
```

```
// EXEC    LIBR
   RESTOR  SUB=SAGLIB.COMvrs : SAGLIB.COMvrs  -
           R=Y TAPE=SYS006

/*
```

Notes:

1. The cuu here must be set to an installation unit name relating to the tape device to be used to load the tape.
2. vrs in relates to the Version, Release and SM Level of the COM‐PLETE being installed, for example, the tape volser for COM‐PLETE 5.1.0 would be called COM510.
3. The volser vvvvvv should be changed to the volume serial number on which you wish the data set to be downloaded.
4. The track ttttt must be changed to the track for the specified volume.
5. The number nnnn must be changed to the number of tracks for the specified data set.
6. For the value of xx, rewind the tape and use the relevant number from the Report of Tape Creation.

B. JCLINST2

```
* $$ JOB    JNM=JCLINST2,      ..... JECL CARD INFORMATION ....
* $$ LST    DISP=D,CLASS=A
// JOB      JCLINST2          ..... JOB CARD INFORMATION .....
/*
/* THIS IS THE Com-plete INSTALLATION JOB2
/*
/* THIS JOB CREATES THE Com-plete USER DATA LIBRARY
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS
/* JOB.
/*
/* 1. INSERT A VALID JECL AND JOB CARD
/* 2. CHANGE THE OUT DSN TO THE REQUIRED LIBRARY.
/*
/* =====
* CREATE    SUBLIB SAGLIB.COMUSER
/* =====
/*
// EXEC     LIBR
   DEFINE   SUB=SAGLIB.COMUSER  -
           REPLACE=YES

/*
/&
* $$ EOJ
```

C. JCLINST4

```
* $$ JOB    JNM=JCLINST4,      ..... JECL CARD INFORMATION .....
* $$ LST    DISP=D,CLASS=A
// JOB      JCLINST4          ..... JOB CARD INFORMATION .....
/*
/* THIS IS Com-plete INSTALLATION JOB4
/*
/* THIS JOB ALLOCATES AND INITIALISES THE Com-plete SYSTEM DATASETS.
/*
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS
/* JOB:#
/*
/*
```

```

/* 1. INSERT A VALID JECL AND JOB CARD.
/* 2. CHANGE VVVVVV TO THE REQUIRED VOLSER.
/* 3. CHANGE vrs TO THE Com-plete VERSION, RELEASE AND SM LEVEL.
/* 4. CHANGE CCCC TO THE REQUIRED NUMBER OF CYLINDERS.
/* 5. CHANGE ????? TO THE REQUIRED CATALOG FILE NAME.
/*
/* =====
* CREATE Com-plete COMSYS.BASE DATA SET (VSAM)
* CREATE Com-plete COMSYS.USERS DATA SET (VSAM)
* CREATE Com-plete COMSYS.CATALOG DATA SET (VSAM)
/* =====
// EXEC IDCAMS
/* DELETE (COM.COMSYS.BASE) CLUSTER - */
/* CATALOG (?????????) */
/* DELETE (COM.COMSYS.USERS) CLUSTER - */
/* CATALOG (?????????) */
/* DELETE (COM.COMSYS.CATALOG) CLUSTER - */
/* CATALOG (?????????) */
/* */
/* COMSYS.BASE */
/* */
DEFINE CLUSTER -
( NAME (COM.COMSYS.BASE) -
INDEXED -
SHAREOPTIONS (2 3) -
RECORDSIZE (4089 4089) -
KEYS (16 0) -
SPEED REUSE) -
DATA -
( NAME (COM.COMSYS.BASE.DATA) -
CISZ (4096) -
VOLUMES (VVVVVV) -
CYLINDERS (0001 01) ) -
INDEX -
( NAME (COM.COMSYS.BASE.INDEX) -
CISZ (2048) -
VOLUMES (VVVVVV) -
TRACKS (001 01) ) -
CATALOG (?????????)

IF LASTCC = 0 -
THEN
LISTCAT ENT (COM.COMSYS.BASE) ALL -
CATALOG (?????????)
/* */
/* COMSYS.USERS */
/* */
DEFINE CLUSTER -
( NAME (COM.COMSYS.USERS) -
INDEXED -
SHAREOPTIONS (2 3) -
RECORDSIZE (2041 2041) -
KEYS (16 0) -
SPEED REUSE) -
DATA -
( NAME (COM.COMSYS.USERS.DATA) -
CISZ (4096) -
VOLUMES (VVVVVV) -
CYLINDERS (0003 03) ) -
INDEX -
( NAME (COM.COMSYS.USERS.INDEX) -

```

```

        CISZ (2048) -
        VOLUMES (VVVVVV) -
        TRACKS (002 01) ) -
    CATALOG (????????)
IF LASTCC = 0 -
    THEN
        LISTCAT ENT (COM.COMSYS.USERS) ALL -
            CATALOG (????????)
/*                                                    */
/*    COMSYS.CATALOG                                */
/*                                                    */

DEFINE CLUSTER -
    ( NAME (COM.COMSYS.CATALOG) -
      INDEXED -
      SHAREOPTIONS (2 3) -
      RECORDSIZE (313 313) -
      KEYS (16 0) -
      SPEED REUSE) -
    DATA -
        ( NAME (COM.COMSYS.CATALOG.DATA) -
          CISZ (4096) -
          VOLUMES (VVVVVV) -
          TRACKS (0005 02) ) -
    INDEX-
        ( NAME (COM.COMSYS.CATALOG.INDEX) -
          CISZ (2048) -
          VOLUMES (VVVVVV) -
          TRACKS (001 01) ) -
    CATALOG (????????)
IF LASTCC = 0 -
    THEN
        LISTCAT ENT (COM.COMSYS.CATALOG) ALL -
            CATALOG (????????)
/*
/*
// IF      $RC > 0 THEN      /* DEFINE CLUSTER SUCCESSFULL ??? */
// GOTO    $EOJ              /* NO - GO TO END OF JOB          */
/*
/*
/* =====
*  LOAD COMSYS.BASE          VSAM DATA SET
/* =====
// PAUSE      WAITING FOR TAPE UNIT
// ASSGN     SYS004,???      <----- CUU FOR TAPE UNIT
// MTC       REW,SYS004
// TLBL      INBAS,'COMSYS.BASE',,,,2
// TLBL      INUSE,'COMSYS.USERS',,,,3
// TLBL      INCAT,'COMSYS.CATALOG',,,,4
/*
// DLBL      OUTBAS,'COM.COMSYS.BASE',,VSAM,CAT=COMCAT
// DLBL      OUTUSE,'COM.COMSYS.USERS',,VSAM,CAT=COMCAT
// DLBL      OUTCAT,'COM.COMSYS.CATALOG',,VSAM,CAT=COMCAT
/*
// DLBL      COMCAT,'????????',,VSAM      <----- VSAM CATALOG NAME
/*
// EXEC      IDCAMS
/* ===== */
/*    LOAD COMSYS.BASE                                */
/* ===== */
REPRO INFILE (INBAS -
        ENVIRONMENT -

```

```

                (PDEV (2400)      - /* TAPE INPUT */
                RECFM (VARBLK)    -
                BLKSZ (16384)    -
                REWIND  ))        -
        OUTFILE (OUTBAS)  REPLACE REUSE

/* ===== */
/*      LOAD COMSYS.USERS      */
/* ===== */
REPRO  INFILE (INUSE -
        ENVIRONMENT -
        (PDEV (2400)      - /* TAPE INPUT */
        RECFM (VARBLK)    -
        BLKSZ (16384)    -
        REWIND  ))        -
        OUTFILE (OUTUSE)  REPLACE REUSE
/* ===== */
/*      LOAD COMSYS.CATALOG    */
/* ===== */
REPRO  INFILE (INCAT -
        ENVIRONMENT -
        (PDEV (2400)      - /* TAPE INPUT */
        RECFM (VARBLK)    -
        BLKSZ (16384)    -
        REWIND  ))        -
        OUTFILE (OUTCAT)  REPLACE REUSE

/*
/&
* $$ EOJ

```

D. JCLINST5

```

* $$ JOB   JNM=JCLINST5,      ..... JECL CARD INFORMATION .....
* $$ LST   DISP=D,CLASS=A
// JOB     JCLINST5           ..... JOB CARD INFORMATION .....
/*
/*      THIS IS THE Com-plete INSTALLATION JOB5.
/*
/*      THIS JOB ALLOCATES AND INITIALISES THE Com-plete CAPTURE DATA SETS
/*
/*      THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB
/*
/*      1. INSERT A VALID JOBCARD.
/*      2. CHANGE VVVVVV TO THE REQUIRED VOLSER(S).
/*      3. CHANGE ?????? TO THE REQUIRED CATALOG FILE NAME.
/*
// EXEC    IDCAMS
/* DELETE (COM.VSAM.CAPTUR1) cluster - */
/*      CATALOG (?????????)          */
/* DELETE (COM.VSAM.CAPTUR2) cluster - */
/*      CATALOG (?????????)          */
DEFINE CLUSTER -
        ( NAME (COM.VSAM.CAPTUR1) -
          NONINDEXED -
          SHAREOPTIONS (1) -
          RECORDSIZE (0032 4096) -
          SPEED REUSE ) -
        DATA -
        ( NAME (COM.VSAM.CAPTUR1.DATA) -
          CISZ (8192) -
          VOLUMES (VVVVVV) -

```

```

        CYLINDERS (3 0)) -
        CATALOG (????????)
DEFINE CLUSTER -
        ( NAME (COM.VSAM.CAPTUR2) -
          NONINDEXED -
          SHAREOPTIONS (1) -
          RECORDSIZE (0032 4096) -
          SPEED REUSE ) -
        DATA -
        ( NAME (COM.VSAM.CAPTUR2.DATA) -
          CISZ (8192) -
          VOLUMES (VVVVVV) -
          CYLINDERS (3 0)) -
        CATALOG (????????)
/*
*   INIT CAPTURE FILES
/*
// DLBL      COMCAT,'????????',,VSAM      <----- VSAM CATALOG NAME
/*
// DLBL      CAPTUR1,'COM.VSAM.CAPTUR1',,VSAM,CAT=COMCAT
// DLBL      CAPTUR2,'COM.VSAM.CAPTUR2',,VSAM,CAT=COMCAT
// LIBDEF    PHASE,SEARCH=SAGLIB.COMvrs,TEMP
// EXEC      TUSACAPT,SIZE=220K
/*
/&
* $$ EOJ

```

E. JCLINST6

```

* $$ JOB    JNM=JCLINST6,      ..... JECL CARD INFORMATION .....
* $$ LST    DISP=D,CLASS=A
// JOB      JCLINST6      ..... JOB CARD INFORMATION .....
/*
/*   THIS IS THE Com-plete INSTALLATION JOB6.
/*
/*   THIS JOB ALLOCATES AND INITIALISES THE Com-plete SD-FILE DATA SET
/*
/*   THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB
/*
/*   1. INSERT A VALID JOBCARD.
/*   2. CHANGE VVVVVV TO THE REQUIRED VOLSER.
/*   3. CHANGE ?????? TO THE REQUIRED CATALOG FILE NAME.
/*
/*
/*
/* =====
* DEFINE Com-plete COM.VSAM.SDFILE DATA SET
/* =====
// EXEC      IDCAMS
/*   DELETE (COM.VSAM.SDFILE)      CLUSTER -   */
/*           CATALOG (????????)      */
DEFINE CLUSTER -
        ( NAME (COM.VSAM.SDFILE)      -
          NUMBERED -
          SHAREOPTIONS (2 3) -
          RECORDSIZE (4080 4080) -
          SPEED REUSE) -
        DATA -
        ( NAME (COM.VSAM.SDFILE.DATA) -
          CISZ (4096) -
          VOLUMES (VVVVVV) -

```

```

        CYLINDERS (0020 00) ) -
        CATALOG (????????)

/*
/* =====
*   INIT   Com-plete COM.VSAM.SDFILE
/* =====
// LIBDEF   PHASE,SEARCH=SAGLIB.COMvrs,TEMP
// ASSGN    SYS003,SYSLST
/*
// DLBL     COMCAT,'????????',,VSAM           <----- VSAM CATALOG NAME
/*
// DLBL     COMSD,'COM.VSAM.SDFILE',,VSAM,CAT=COMCAT
// EXEC     TUSDUTIL,SIZE=220K,PARM='INIT,SDFILES=0200,DMPSPAC=8'
/*
/* =====
*   DISPLAY Com-plete COM.VSAM.SDFILE
/* =====
// EXEC     IDCAMS
           LISTCAT ENT (COM.VSAM.SDFILE) ALL -
           CATALOG (????????)

/*
/&
* $$ EOJ

```

F. JCLINST7

```

* $$ JOB   JNM=JCLINST7,      ..... JECL CARD INFORMATION .....
* $$ LST   DISP=D,CLASS=A
// JOB     JCLINST7          ..... JOB CARD INFORMATION .....
/*
/*   THIS IS THE Com-plete INSTALLATION JOB7.
/*
/*   THIS JOB ALLOCATES AND INITIALISES THE Com-plete
/*   PRINTOUT-SPOOL DATA SET
/*
/*   THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB
/*
/*   1. INSERT A VALID JOBCARD.
/*   2. CHANGE VVVVVV TO THE REQUIRED VOLSER.
/*   3. CHANGE ?????? TO THE REQUIRED CATALOG FILE NAME.
/*
/*
/* =====
*   DEFINE Com-plete COM.VSAM.SPOOL DATA SET
/* =====
// EXEC     IDCAMS
           /*   DELETE (COM.VSAM.SPOOL)   CLUSTER -   */
           /*           CATALOG (????????)           */
DEFINE     CLUSTER -
           ( NAME (COM.VSAM.SPOOL) -
             NUMBERED -
             SHAREOPTIONS (2 3) -
             RECORDSIZE (4080 4080) -
             SPEED REUSE) -
           DATA -
           ( NAME (COM.VSAM.SPOOL.DATA) -
             CISZ (4096) -
             VOLUMES (VVVVVV) -
             CYLINDERS (0010 00) ) -
           CATALOG (????????)

/*

```

```

/* =====
* INIT Com-plete COM.VSAM.SPOOL
/* =====
// LIBDEF PHASE,SEARCH=SAGLIB.COMvrs,TEMP
// ASSGN SYS003,SYSLST
/*
// DLBL COMCAT,'????????',,VSAM <---- VSAM CATALOG NAME
/*
/* RECS= NUMCYL * NUMTRC * NUMREC
/* =====
/* WHERE:
/* NUMCYL = NUMBER OF CYLINDER
/* NUMTRC = NUMBER OF TRACKS PER CYLINDER
/* NUMREC = NUMBER CONTROLINTERVAL PER TRACK
/*
/*
// DLBL COMSPL,'COM.VSAM.MSGLIB',,VSAM,CAT=COMCAT
// EXEC TUMSUTIL,SIZE=220K
INIT RECS=1500
/*

/* =====
* DISPLAY Com-plete COM.VSAM.SPOOL
/* =====
// EXEC IDCAMS
LISTCAT ENT (COM.VSAM.SPOOL) ALL -
CATALOG (????????)
/*
/&
* $$ EOJ

```

G. JCLINST8

```

* $$ JOB JNM=JCLINST8, ..... JECL CARD INFORMATION .....
* $$ LST DISP=D,CLASS=A
// JOB JCLINST8 ..... JOB CARD INFORMATION .....
/*
/* THIS IS THE COM-LETE INSTALLATION JOB8.
/*
/* THIS JOB ALLOCATES AND INITIALISES THE COM-LETE DUMP DATA SET
/*
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB
/*
/* 1. INSERT A VALID JECL AND JOB CARD.
/* 2. CHANGE VVVVVV TO THE REQUIRED VOLSER.
/* 3. CHANGE ??????? TO THE REQUIRED CATALOG FILE NAME.
/*
/*
/* =====
* DEFINE COM-LETE COM.VSAM.DUMPFIL
/* =====
// EXEC IDCAMS
/* DELETE (COM.VSAM.DUMPFIL) CLUSTER - */
/* CATALOG (???????) */
DEFINE CLUSTER -
( NAME (COM.VSAM.DUMPFIL) -
NUMBERED -
SHAREOPTIONS (2 3) -
RECORDSIZE (4080 4080) -
SPEED REUSE) -

```

```

DATA -
  ( NAME (COM.VSAM.DUMPFILe.DATA) -
    CISZ (4096) -
    VOLUMES (VVVVVV) -
    CYLINDERS (0120 00) ) -
  CATALOG (??????)
/* ===== */
/* DISPLAY COM-plete COM.VSAM.DUMPFILe */
/* ===== */
LISTCAT ENT (COM.VSAM.DUMPFILe) ALL -
  CATALOG (??????)
/*

```

H. JCLINST9

```

* $$ JOB JNM=JCLINST9, ..... JECL CARD INFORMATION .....
* $$ LST DISP=D,CLASS=A
// JOB JCLINST9 ..... JOB CARD INFORMATION .....
/*
/* THIS IS THE Com-plete INSTALLATION JOB9
/*
/* DEFINE THE VTAM APPLICATION FOR Com-plete
/*
/* THE FOLLOWING CHANGES HAVE TO BE PERFORMED BEFORE RUNNING THIS JOB
/*
/* 1. INSERT A VALID JOBCARD.
/* 2. CHANGE THE VTAM APPLICATION ID TO THE REQUIRED NAME.
/*
/* =====
* DEFINE THE VTAM APPLICATION FOR Com-plete
* =====
// EXEC LIBR,PARM='MSHP'
ACC SUB=PRD2.CONFIG
CATALOG APPLCOM.B R=Y
APPLCOM VBUILD TYPE=APPL
COMPLETE APPL AUTH=(ACQ,PASS),APPC=YES
/+
/*
/&
* $$ EOJ

```

Installation and Customization of TPF

This documentation describes installation and customization of the TPF component in the supported environments CICS, TSO and Com-plete.

This information is organized under the following headings:

- Installing TPF in the Host Environment
- General Logon Authorization
- The ACSTAB Table
- CICS: Special Considerations
- Transparency (CICS and TSO) and Reverse Access (CICS)
- Exits for the TPF Component
- Starting Access from a User Program

Installing TPF in the Host Environment

The TPF component uses the Adabas SVC to communicate and pass data between the host and target systems. This component and the appropriate Adabas linkage routine must therefore be installed prior to the installation of TPF. Please consult the appropriate Adabas documentation.

This document covers the following topics:

- TPF Facilities Overview
 - Installation under CICS
 - Installation under TSO (z/OS only)
 - Installation under Com-plete
 - What Next?
-

TPF Facilities Overview

TPF provides a number of facilities to allow you to customize the product to your site's requirements. This section gives a brief outline of them.

- **The ACSTAB table:**
Site-specific items on the host side of an TPF installation can be specified in the ACSTAB table. These include, for example, the Adabas SVC number, DBID and the linkage routine to be used.
- **The Direct Call feature:**
This feature enables the user to define transactions that are automatically routed to the TPF target system, where a specific program or transaction is started. The Direct Call feature is described in detail in the section *TPFXTAB Subtable in The ACSTAB Table*.
- **Printing from TPF:**
CICS sites can define a local printer in the TPF target system's TIBTAB. Printouts from the target system are printed on the corresponding CICS local printer.
- **The Transparency feature:**
Transparency is a feature of TPF which simplifies the conversation between a Natural CICS/TSO environment and an environment in which Natural runs under TPF. The transparency feature allows Natural programs which formerly ran under CICS or TSO to run under TPF with no changes. More details on transparency are given in *TPF Transparency under CICS and TSO*.
- **Reverse Access programming:**
Many Natural programs currently under CICS call 3GL subroutines to perform certain functions. If these subroutines do not directly use any CICS facilities, then they can simply be installed to run in the TPF address space/partition. However, if a 3GL subroutine uses CICS for any reason, (for example, EXEC CICS statements), then it must run in the CICS address space/partition. Reverse Access provides this capability as a feature of TPF transparency, and is described in *TPF Transparency under CICS and TSO*.

Installation under CICS

To install the TPF component under CICS, proceed as follows:

Step 1: Unload the Installation Data Set

- Unload the COMvrs.SCRT and COMvrs.LOA3 data sets from the installation tape. Please consult the report from the tape creation system to determine the order of the files on the installation tape. The COMvrs.LOA3 library contains the TPF load modules required for CICS.

On VSE systems, note that the required library is the TPFvrs sublibrary unloaded in the installation procedure using the sample JCS JCLINST1.

Step 2: Modify the ACSTAB Table

- *The ACSTAB Table* contains a full description of the ACSTAB table. An example ACSTAB is provided in the source library called ACSTAB, use this and modify it to suit your installation requirements. The table is subdivided in four parts. Each part can be modified to suit your own installation requirements.
 1. ACSDEF describes the ACCESS Default Table which is used to define data used by the host node. This subtable must be unique for each host region.
 2. ACSTBL describes the ACCESS Node Table which lists the various target systems available for communication with the host system and also the host system with an entry for every CICS host node.
 3. ACSSCHC describes the ACCESS screen-to-hardcopy table which defines the hardcopy device printer where printout will be routed for every terminal in the system.

Note:
If you do not require the screen-to-hardcopy function, then you need not specify this subtable.
 4. TPFXTAB describes the Transparency table which describes the various transactions you can use to enter your TPF environment and the way you give user ID, password and data to your TPF.

Step 3: Assemble and Link the ACSTAB Table

- After modification, assemble and link the ACSTAB table using the sample job from the source library (JCLASMTB).

Step 4: Link the CICS Modules

- Using the sample link job in the source library (JCLLNKCI), link the TPF modules to a load library which is concatenated in the CICS start-up procedure. You are recommended *not* to use the load library you loaded from the tape.

Step 5: Define the CICS Environment

1. Include the TPF modules in the program tables or define them with the CICS CEDA online transaction.

The following entries must be defined in the PPT or with CEDA.

```
DFHPPT TYPE=ENTRY,PROGRAM=ACCESS46,RES=YES
DFHPPT TYPE=ENTRY,PROGRAM=ACSBEG46
DFHPPT TYPE=ENTRY,PROGRAM=ACSSPL46
DFHPPT TYPE=ENTRY,PROGRAM=ACSTAB46
DFHPPT TYPE=ENTRY,PROGRAM=TPFXF46
DFHPPT TYPE=ENTRY,PROGRAM=ACSUCTR )see note d
```

The following entries must be defined in the PCT or with CEDA.

```
DFHPCT TYPE=ENTRY,PROGRAM=ACCESS46, *
      TRANSID=XT46,TWASIZE=250 )see note a *
DFHPCT TYPE=ENTRY,PROGRAM=TPFXF46, *
      TRANSID=ADMN,TWASIZE=250 )see note b *
DFHPCT TYPE=ENTRY,PROGRAM=TPFXF46, *
      TRANSID=ATPF,TWASIZE=250 )see note b *
DFHPCT TYPE=ENTRY,PROGRAM=TPFXF46, *
      TRANSID=BTPF,TWASIZE=250 )see note b *
DFHPCT TYPE=ENTRY,PROGRAM=ACSBEG46, *
      TRANSID=AB46,TWASIZE=250 )see note c *
DFHPCT TYPE=ENTRY,PROGRAM=ACSSPL46, *
      TRANSID=AP46,TWASIZE=360 )see note c *
DFHPCT TYPE=ENTRY,PROGRAM=ACSUCTR,
      TRANSID=ACSU,TWASIZE=250 )see note d
```

Notes:

1. 'XT46' or the TRANSID transaction ID specified in the ACSDEF sub-table.
2. Plus any other user transactions defined in the TPFXTAB sub-table.
3. 'AB46' and 'AP46' are reserved transaction IDs for printing under TPF.
4. These entries are only required if the upper case translation feature is to be implemented. The program ACSUCTR and transaction ACSU are redundant under CICS 3.3 and above, and should not be installed (see the section on upper case translation in the section *The ACSTAB Table*).

If the transaction AB46 and AP46 are not defined, then the terminal functions of TPF will be available as normal but printing functions will not be available.

2. In order to automatically activate printer support each time CICS starts, include program ACSBEG46 in the PLT:

```
DFHPLT TYPE=ENTRY,PROGRAM=ACSBEG46
```

- If ACSBEG46 is not defined in the PLT, printing can be started by starting transaction AB46 from the terminal.

More information about TPF printing is given in the section *Printing under TPF*.

- If CEDA was not used to define the environment, the PLT, PPT and PCT tables must now be assembled and linked in the usual manner, and the CICS system must be restarted.
- If CEDA was used, then the group defined has to be installed.

Step 6: Verify the Installation and Log On

- The installation of the TPF component is now finished.

To test the installation, use the ADMN transaction, which logs on to TPF using the predefined user ID SAGADMIN and password ADMIN:

```
ADMN ( , ,ADMIN)
```

This user ID has a definition on the system data set with the required authorization to define more user IDs to the system using the user ID maintenance facilities of UUTIL (see the Complete Utilities documentation). The initial password required by UUTIL is PASSWORD, unless otherwise specified by the sysparm ULOGM.

After defining other user IDs (see the section *User ID Considerations*), you are recommended to change the password for SAGADMIN and use the user ID SAGADMIN in emergency cases only.

Installation under TSO (z/OS only)

To install the TPF component under TSO in z/OS environments, proceed as follows:

Step 1: Unload the Installation Data Set

- Unload the COMvrs.SRCT and COMvrs.LOA3 libraries from the installation tape.

Please consult the report from the tape creation system in order to determine the order of the files on the tape. The load library contains the relevant load modules for TSO.

Step 2: Modify the ACSTAB Table

- For a full description of the ACSTAB table see the section *The ACSTAB Table*. An example ACSTAB is specified in the source library called ACSTAB, use this and modify it to suit your installation.

The table is subdivided in four parts.

1. ACSDEF describes the ACCESS Default Table which is used to define data used by the host node. This subtable must be unique for each host region.
2. ACSTBL describes the ACCESS Node Table which lists the various target systems available for communication with the host system and also the host system with an entry for every TSO host node.
3. ACSSCHC describes the ACCESS screen-to-hardcopy table. This is not required for TSO.
4. TPFXTAB describes the Transparency table which describes the various transactions you can use to enter your environment and the way you give user ID, password and data to your system.

Each part can be modified to suit your own installation requirements.

Step 3: Assemble and Link the ACSTAB Table

- After modification, the ACSTAB table must be assembled and linked using the sample job from the source library (JCLASMTB).

Step 4: Link the TSO Modules

- Using sample job JCLLNKTS, link the TPF module to an appropriate TSO library. Do *not* use the load you loaded from tape.

Step 5: Create the TSO Environment

- Create and save a CLIST in an appropriate TSO library using the sample given in member JCLCLIST. This member includes two useful examples of a simple way to invoke TPF under TSO.

The TSORUN member supplied consists of a single statement which is used to control the TSO TPF programs. Using this simple ADARUN card the SVC number of the Adabas SVC and the Adabas runtime options can be specified. Please modify the Adabas SVC number in the supplied TSORUN member to suit your installation.

Note that TPF under TSO communicates using the Adabas SVC via the link routine ADALNK. The module TPFXTSO is linked with the Adabas module ADAUSER, the first time ADAUSER is called it loads module ADARUN which subsequently loads the link routine ADALNK to issue the SVC request.

The ADARUN card, the ADALNK, ADARUN and ADAUSER modules must therefore be accessible at run-time.

TPFXTSO will load the ACSTAB during initialization so that ACSTAB must also be accessible.

Step 6: Verify the Installation and Log On

- The installation on the host side is now finished. To test the installation, execute the CLIST created.

To log on directly to TPF with the predefined userid SAGADMIN, use the optional keywords `tranid`, `nodeid`, `userid` and `password` in the CLIST.

```
EX 'xx.xx.xx(clist)' 'TRANID(ADMN) NODEID (COM51) USERID(SAGADMIN)
PASSWORD(ADMIN)'
```

See the example job in the source library (JCLCLIST).

This user ID has a definition on the system data set with the required authorization to define more user IDs to the system using the user ID maintenance facilities of UUTIL (see the Complete Utilities documentation). The initial password required by UUTIL is PASSWORD, unless otherwise specified by the sysparm ULOGM.

After defining other user IDs (see *User ID Considerations*), you are recommended to change the password for SAGADMIN and use the user ID SAGADMIN in emergency cases only.

Installation under Com-plete

To install the TPF component under Com-plete, proceed as follows:

Step 1: Unload the Installation Data Set

- Unload the COMvrs.SRCT and COMvrs.LOA3 libraries from the installation tape.

Please consult the report from the tape creation system in order to determine the order of the file on the tape. The load library contains the load modules for Com-plete.

On VSE systems, note that the host.load library is the TPFvrs.sublibrary unloaded in the installation procedure using the sample JCS JCLINST1.

Step 2: Modify the ACSTAB Table

- For a full description of the ACSTAB table see the section *The ACSTAB Table*. An example ACSTAB is specified in the source library called ACSTAB, use this and modify it to suit your installation.

The table is subdivided in four parts.

1. ACSDEF describes the ACCESS Default Table which is used to define data used by the host node. This subtable must be unique for each host region.
2. ACSTBL describes the ACCESS Node Table which lists the various target systems available for communication with the host system and also the host system with an entry for every TSO host node.
3. ACSSCHC describes the ACCESS screen-to-hardcopy table. This is not required for Com-plete.
4. TPFXTAB describes the Transparency table which describes the various transactions you can use to enter your environment and the way you give user ID, password and data to your system.

Each part can be modified to suit your own installation requirements.

Step 3: Assemble and link the ACSTAB Table

- After modification the ACSTAB table must be assembled and linked using the example job from the source library (JCLASMTB).

Step 4: Link the Module for Com-plete

- With the example job JCLLNKCO, link the TPF module to an appropriate load library specified in the COMPLETE start-up procedure. Please do not link the module to the load you loaded from tape.

Step 5: Create the Com-plete environment

- Catalog the linked TPF module with ULIB using the following command:

```
ULIB CAT,TPF46,RG=32K,PV
```

Step 6: Verify the Installation and Log On

- The installation on the host side is now finished. To test the installation, call the module TPF46 from the USTACK menu. Log on directly to TPF with the predefined user ID SAGADMIN and password ADMIN.

This user ID has a definition on the system data set with the required authorization to define more user IDs to the system using the user ID maintenance facilities of UUTIL (see the Complete Utilities documentation). The initial password required by UUTIL is PASSWORD, unless otherwise specified by the sysparm ULOGM.

After defining other user IDs (see the section *User ID Considerations*), you are recommended to change the password for SAGADMIN and use the user ID SAGADMIN in emergency cases only.

What Next?

TPF is now up and ready for work. How you continue depends on whether you have installed TPF for the first time or whether you are migrating from a previous version.

If you have installed TPF for the first time, no further migration is necessary. You can continue with customization steps described in the Complete System Programming documentation and the Complete Utilities documentation.

If you are going to use the model user ID, SYSCOM, then you will need to modify the definition of this user to suit your requirements (via the UUTIL subfunction UM).

General Logon Authorization

This chapter covers the following topics:

- User ID Considerations
 - Program Authorization
 - Password Considerations
-

User ID Considerations

In order to establish communication with Com-plete, a user must supply a valid user ID and password.

Initially, two user IDs are defined:

- SAGADMIN, the ID for the system administrator;
- SYSCOM, a model user ID for other users.

Using the administrator ID SAGADMIN, you can define further user IDs for other users. If a user ID has been defined to Com-plete, the profile defined for that user ID is used whenever a user logs on with that user ID.

If a user logs on with a user ID unknown to Com-plete, the user is assigned the profile specified by the MODEL parameter in the TPFXTAB subtable (see the section *The ACSTAB Table*). This is typically the provided default SYSTPF, but you can define other model profiles as required.

Program Authorization

Any programs to be executed under Com-plete must be added to the menu list maintained on the system data set, and must also be specified in the user's profile (see the UUTIL functions UM, ML and UP). Users cannot invoke programs that are not listed in the profile assigned to their user ID, except USTACK.

USTACK can be invoked as a direct call, as specified by the TPFPGM parameter {TPFPGM=(DCALL,USTACK)}. Under normal circumstances, you are advised for the sake of convenience, to provide a fairly comprehensive default user profile (SYSTPF). This is recommended because the TPFXTAB subtable ultimately determines which programs can actually be started by the user.

Password Considerations

Without External Security

The following considerations apply if the target Com-plete system is not running under the control of an external security system (SECSYS sysparm is not specified).

1. If the user ID entered by the user is defined in the Com-plete system, the supplied password is validated by Com-plete during the logon process, and if correct, the user is logged on. The user is assigned a profile according to the specifications in the system data set.

Note:

If a user is defined to the target Com-plete system but the logon specifies a model user ID, then the profile will be taken from the model user ID and not from the definition of the user ID logging on. See the MODEL parameter in ACSTAB.

2. If the user ID entered by the user is not defined in the Com-plete system and a MODEL user ID exists, the supplied password is not validated by Com-plete. The user is logged on and assigned a profile according to the MODEL definition.
3. If the user ID entered by the user is not defined in the Com-plete system and no MODEL user ID exists, the logon request is rejected.

With External Security

The following considerations apply if the target Com-plete system is running under the control of an external security system (SECSYS sysparm is specified).

1. If the user ID entered by the user is defined in the Com-plete system, the supplied password is validated by the external security system during the logon process, and if correct, the user is logged on. The user is assigned a profile according to the specifications in the system data set.

Note:

If a user is defined to the target Com-plete system but the logon specifies a model user ID, then the profile will be taken from the model user ID and not from the definition of the user ID logging on. See the MODEL parameter in ACSTAB.

2. If the user ID entered by the user is not defined in the Com-plete system and a MODEL user ID exists, the supplied password is validated by the external security system. If the password is correct, the user is logged on and assigned a profile according to the MODEL definition. The security profile (ACEE) defined by the external security system is taken.
3. If the user ID entered by the user is not defined in the Com-plete system and not defined to the external security system, the user is logged on only if a MODEL user ID exists. The user is assigned a profile according to the MODEL definition, but no ACEE is provided. This means that any request for items protected by the external security system will be rejected (but see the note below).

Note:

If the MODEL user ID is defined to the external security system, the security profile (ACEE) defined in the security system for the MODEL user ID is used.

The ACSTAB Table

The ACSTAB table is subdivided into the four subtables ACSDEF, ACSTBL, ACSSCHC and TPFXTAB.

The ACSTAB always starts with the macro ACSSTART, which is required for every ACSTAB. ACSSTART must be the first entry in every ACSTAB.

This chapter covers the following topics:

- ACSDEF Subtable
 - ACSTBL Subtable
 - ACSSCHC Subtable (CICS only)
 - TPFXTAB Subtable
 - Passing Data
 - Overriding Information
-

ACSDEF Subtable

The ACSDEF subtable describes the default values of the system, and defines data used by the host node. These include:

- The host node ID (DBID);
- The name of the Adabas TPF host driver program (APPLID);
- The name of the continuation transaction (TRANSID);
- The name of the program which invokes the Adabas SVC for communication with the target system (Adabas).

The ACSDEF subtable is only needed if either:

- CICS is the host system, or
- for any other system if the size of the format buffer is to be increased to a value higher than the default 3840 bytes.

ACSDEF must be unique for each CICS host system. If you are running Adabas TPF on two CICS you need two ACSTABs, each with a different ACSDEF subtable.

The following parameters can be specified with the ACSDEFLT macro, which describes the ACSDEF subtable:

Parameter	Meaning
DBID	This is the 1 byte value representing the pseudo-DBID to be used to identify the CICS host system. As default, use DBID 127.
TRANSID	This is the 4 byte character string representing the transaction name to be used by the TPF component for internal continuation. As default, use the trans ID "XT46". If you use a different TRANSID, it must be defined in the PCT or with CEDA.
APPLID	This is the 8 byte character string representing the module name of the CICS driver that is invoked with the Reverse Access call (SCALL). The Reverse Access call is described in the section on transparency in this documentation.
CONTID	This is the 4 byte character string representing the transaction name to be used by TPF for the Reverse Access call (SCALL). The Reverse Access call is described later in this documentation.
Adabas	This is the 8 byte character string representing the name of the module to be loaded that performs the link to Adabas (this module must reside in the CICS loadlib).
FORMATSZ	This is the 2 byte character string representing the value for the format buffer size. The default format buffer size is: {(lines x columns) x 2}; for example, (24 x 80) x 2=3840. If you wish to increase the size of the format buffer because of large Natural screens or Reverse Access, you can increase the value here. The maximum value is 32k bytes, otherwise the assembler returns a high return code. No value of less than 3840 is accepted. If you specify a lower value, it is overridden by the default value 3840. If this parameter is required, then the subtable ACSDEF must be specified for all host systems.
PSEUDO	YES or NO, indicating whether CICS processing should be pseudo-conversational or not.

ACSTBL Subtable

The ACSTBL subtable contains a list of target systems available for communication with the various host systems. The table provides a cross-reference of node names to node numbers. Routing requests made to the Adabas TPF task are evaluated according to the contents of this table. For each target system you wish to communicate with Adabas TPF, you need one ACSTABLE macro with its parameters.

The following parameters can be specified with the macro ACSTABLE which describes the ACSTBL subtable:

Parameter	Meaning
DEST	This is the 8 byte character string that is used to identify the Com-plete target node. For access to a target node from a batch job this must be "BATCH".
NODEID	This is the 2 byte value equivalent of the DBID for use with the Adabas SVC. Use the value you specified for the sysparm ACCESS-ID in the target system.
SVC	This is the one byte value equivalent to the Network Router (Adabas SVC). Use the value you specified for the sysparm ACCESS-SVC in the target system. This parameter only has meaning with DEST=BATCH.
MENU	Specifies whether a menu showing all valid destinations is to be displayed under the following circumstances: If the TPF driver was entered directly (that is, bypassing the transparency feature), then if no destination is specified (or the specifies destination is not found), then the menu of all valid destinations is displayed by the TPF terminal task.
COMMENT	This is the 50 byte character string the TPF terminal task displays in MENU mode for each destination.

ACSSCHC Subtable (CICS only)

This subtable is called the screen-to-hardcopy table. The subtable is only needed if you use the screen-to-hardcopy function under TPF. The screen-to-hardcopy function is only available for CICS. One ACSHARDC statement must be specified for each terminal and its assigned printer.

The following parameters can be specified with macro ACSHARDC, which describes the ACSSCHC subtable:

Parameter	Meaning
TERMID	This is the 4 byte character string used to identify the terminal ID.
SCHCNR	This is the 4 byte character string used to identify the terminal ID of the destination hardcopy printer.

TPFXTAB Subtable

This subtable is called the Adabas TPF Transparency table. The purpose of Adabas TPF transparency is to provide an automatic method of integrating Adabas TPF in an existing Natural CICS/TSO installation, and to provide an easy method of implementing new user transactions that invoke Com-plete.

With the user transactions, the startup of a Adabas TPF session is very simple for the user. Every user can be routed to his working environment (Natural) directly, or he can select a transaction that first displays the Com-plete USTACK menu, from which the user can start the required program.

The macro USERDEF describes the TPFXTAB subtable. There are four types of USERDEF statements: GLOBAL, GROUP, TRAN and END. These are described in detail below. The format of a USERDEF macro statement is:

USERDEF type,keyword parameters

- **USERDEF GLOBAL:**

All keyword parameters have installation defaults that can be overridden by a USERDEF GLOBAL statement. When assembly of the table begins and a USERDEF GLOBAL statement is encountered, the values specified for the keyword parameters on that statement become the global defaults for the rest of the assembly, or until a subsequent USERDEF GLOBAL statement provides new defaults. GLOBAL-type USERDEF statements are optional, and are usually used to establish values that are consistent for most transactions, such as NODEID. Note that if a keyword parameter specified on a USERDEF GLOBAL statement is not specified on a subsequent USERDEF statement, it retains the specified value.

- **USERDEF GROUP:**

This type of USERDEF statement is optional and specifies a group of terminals or users within the TPFXTAB to which the specified keyword parameter values are to apply. The specified values override the USERDEF GLOBAL specifications. The values are valid until another GROUP or GLOBAL statement is encountered. Note that if a keyword parameter specified on one USERDEF GROUP statement is not specified on a subsequent USERDEF GROUP statement, it reverts to its original global value (as set by either a previous USERDEF GLOBAL statement or the installation default).

- **USERDEF TRAN:**

The USERDEF TRAN statement is the only statement that actually generates a table entry. This means that at least one USERDEF TRAN statement is required, and one statement must exist for each transaction that can use the transfer program. The table entry is generated using the keyword parameters explicitly specified. If a parameter is not specified on a USERDEF TRAN statement, the value is determined according to the following hierarchy: first the current GROUP value is searched; if not found, the current GLOBAL value is searched; if not found, the installation default is assumed.

A USERDEF TRAN statement is preceded by a 1-4 character name field that specifies the transaction ID to which the table entry is to apply. For CICS, the transaction ID is the actual transaction ID that invokes the transaction. If a valid CICS transaction ID is called which invokes the transfer program, but which has no entry in the TPFXTAB subtable, then the parameters specified for the first transaction entry in TPFXTAB are taken.

- **USERDEF END:**

The USERDEF END statement is required as the last USERDEF statement in the TPFXTAB subtable.

Example:

Following is an example of a TPFXTAB subtable generation:

```

                USERDEF GLOBAL,NODEID=COMXX,USERID=OPID
                TPFPGM=NAT21,ACSPGM=ACCESS46
ATPF  USERDEF  TRAN
BTPF  USERDEF  TRAN,TPFPGM=NAT22,DATA='string1'
                USERDEF  GROUP,TPFPGM=TESTNAT
PAYQ  USERDEF  TRAN,DATA='string2'
PRTK  USERDEF  TRAN,DATA='string3'
                USERDEF  END
                END

```

The table definition provides four transactions: ATPF, BTPF, PAYQ and PRTK.

- The transaction ATPF follows all of the rules specified in the GLOBAL statement; the node ID is COMXX, the user ID is taken from the CICS OPID, the program invoked first on the target side is NAT21, and the name of the TPF ACCESS program is ACCESS46.
- The transaction BTPF also follows the rules specified in the GLOBAL statement, but it invokes program NAT22 on the target side and passes data to it.
- The transaction PAYQ belongs to a group, where the program TESTNAT is invoked, to which data *string2* is passed.
- The description of PAYQ also applies to transaction PRTK, except that different data is passed to the TESTNAT program.

The keyword parameters available for the USERDEF statements are described in the following subsections.

NODEID Parameter

This parameter indicates how the name of the Complete node is to be determined for the transaction. This node name must be defined in the ACSTBL subtable.

Possible values for the NODEID parameter are:

Value	Meaning
REQ	Node ID must be specified via screen input (see section <i>Overriding Information</i>).
(DEFLT,name)	Use specified name as default node ID.
(ALWAYS,name)	Always use name as node ID.
name	Node ID to be used.
EXIT	Use user exit to determine node ID. For more information, see the section <i>Exits for the TPF Component</i> .

USERID Parameter

The USERID parameter indicates how the TPFXFER program is to determine the user's TPF LOGON ID when logging the user on to the associated transaction. A valid user ID must be specified, either on the TRAN statement or on a previous GLOBAL statement; if no valid user ID is specified, an assembly error occurs. One option is to hard-code a value for the transaction. This would mean that every time the given transaction is invoked, ACCESS attempts to log on to Adabas TPF using that ID.

Possible values are:

Value	Meaning
REQ	User ID must be specified via screen input (see section <i>Overriding Information</i>).
EXIT	Use user exit to determine user ID (CICS only). For more information, see the section <i>Exits for the TPF Component</i> .
(DEFLT,name)	Use specified user ID as default.
(ALWAYS,name)	Always use this user ID.
name	se this user ID.
OPID	se CICS OPID as user ID.
TERM	se CICS TERMID as user ID.
USER	Use CICS USERID as user ID. This is especially useful when an external security system, such as RACF, is in place.

For information on how user IDs are defined under Com-plete, see *User ID Considerations* in the installation section.

Note that the password generated is always PASSWORD. If the user ID is defined in the Com-plete system, this field must therefore be overwritten by means of screen input (see the section *Overriding Information* below).

An exception to this is when USERID=USER is specified, when the generated password is set to indicate that the password verification has already been performed by the host system. In this case, if the target Com-plete system is running under the control of an external security system, it will perform a logon without password verification.

If the target Com-plete system is not running with an external security system, the password must be overwritten as described above.

It is not possible under Com-plete for a user to log on with the same user ID more than once. Therefore, under CICS, you must take care when specifying USER=USER or USER=OPID that two (or more) users do not logon with the same user ID. If this does happen, however, the results are undefined, but the first user to log on with the ID will probably be logged off, and the second user to log on with the ID is logged on.

TRANID Parameter

This specifies the CICS transaction associated with this entry. As default the label specified on the statement is taken.

Note:

The label can only start with an alphabetic character, so if an alphanumeric transaction is required, then this parameter must be specified.

TPFPGM Parameter

This parameter is used to indicate the name of a program to be invoked under Com-plete. If the transfer program is being invoked by a 3GL front-end, it will use whatever name was passed to it, overriding the TPFPGM value. This parameter is especially useful when setting up a new transaction which does not use a 3GL front-end.

Possible values:

Value	Meaning
name	Name of the program to be invoked in the Com-plete target system. Users are routed directly to this program, and when they finish work with it, they are returned to the Adabas TPF USTACK menu in TPF mode, from where they can select new programs or logoff from Com-plete with the command LOGOFF. Note: USTACK cannot be called directly in this format (that is, TPFPGM=USTACK is invalid). The program name must be defined on page 1 of the USTACK menu for this user, otherwise access to it is denied.
(DCALL,name)	Direct call in TPFPGM parameter USTACK menu The direct-call, where name is the name of the program to be called directly. When users finish work with this program, they are returned to CICS or TSO, bypassing the Com-plete USTACK menu. However, USTACK can be called in this format: TPFPGM=(DCALL,USTACK). In this case, users are routed to the USTACK menu in Standard mode.
<blank>	If nothing is specified with the parameter TPFPGM on the TRAN level and no GROUP or GLOBAL default is in place, users are routed to the Com-plete menu of USTACK, from where they can select one of the listed programs. When they finish work with the selected program, they are returned to the Com-plete USTACK menu, from where they can logoff with the LOGOFF command.

Important:

The setting of the TPFPGM parameter can be overridden if you specify OVER=ALL (see below). See also the section *Overriding Information*.

ACSPGM Parameter

Specifies the name of the program to which TPFXF46 is to transfer control in order to start up the Com-plete conversation. It simply indicates the proper name of the ACCESS program. Typically, ACCESS46 is the default for this version.

The format of this parameter is:

ACSPGM=name

DATA Parameter

This parameter specifies data to be passed to the program specified by the TPFPGM parameter, for example to pass data to Natural (optionally, this data can consist of the escape character specified by the ESCCHAR parameter to indicate where user-specified data can be embedded.)

This parameter can also be used to define a string of data to be passed to a new transaction which does not have a 3GL front end. The default is: all extra data is passed.

Possible values:

Value	Meaning
string	Data string is passed.
<blank>	Specifies the escape character defined by the ESCCAR parameter.
NO	No data is passed.

For more information, see the section *Passing Data*.

ESCCAR Parameter

Defines an escape character associated with the DATA parameter. This character can be used by the DATA parameter to allow embedding of user-specified data.

The format of this parameter is:

ESCCAR=char

The default is the paragraph sign §. See also the section *Passing Data*.

ENDCHAR Parameter

This parameter specifies a character which can be used internally by this macro in evaluating the DATA parameter. This character cannot appear in the DATA string. The only reason to ever override ENDCHAR is if the default period (full stop) "." appears within a DATA parameter.

The format of this parameter is

ESCCAR=char

The default is the period (full stop) (.).

OVER Parameter

This parameter indicates whether items defined in the TPFXTAB subtable can be overridden by screen input from the user. Up to two items can be specified, for example, the notation OVER=(NODEID,DATA) means the NODEID and DATA parameters can be overridden.

A password can be overridden unless OVER=NO has been specified. If a transaction has OVER=NO defined, as well as DATA=NO, any data on the screen passed during an XCTL is ignored by TPFXF46 (see also the section *Overriding USERDEF Parameters* below).

Possible values include:

Value	Meaning
NODEID	User can override NODEID value.
USERID	User can override USERID value.
DATA	User can override DATA value.
ALL	User can override all parameters.
NO	User cannot override any parameter.

Default: OVER=NO

MODEL Parameter

This parameter specifies the name of a model user ID defined in the Com-plete target system which will be used by the target system to generate the profile for this user. User ID SYSCOM, which can be taken as a model user ID, is provided as default on the supplied installation data set. Note that the MODEL user ID specified here must be defined in the target Adabas TPF system.

If you specify a model user ID with the MODEL parameter, then the user ID used at logon need not be defined in the target Com-plete system, that is, unknown user IDs can log on to the target Com-plete system.

If the parameter MODEL is not specified, then APPLYMOD 57 must be set in the target system to allow unknown user IDs to log on.

The format of the MODEL parameter is

MODEL=userid

There is no default.

Specifying MODEL=NONE forces no generation of the model user ID parameter. You can use this to override a global MODEL specification for a single application.

See also the *User ID Considerations* section.

Passing Data

Data passed to the target program is constructed using the DATA and ESCCHAR parameters associated with a transaction, as well as input data from the screen, or data passed to TPFXFER as part of transfer of control.

A DATA parameter specifies the string of data to be passed to the target program. The data consists of a string of characters enclosed by single quotation marks, for example:

DATA='STACK=(LOGON SASAK)'

A transaction defined with such a parameter always passes the indicated data to the target program. One of the characters within the data string can be the escape character "at" sign (@), for example:

```
DATA=' STACK=(LOGON SASAK;@)'
```

The escape character is a place holder, where any data passed to TPFXFER is placed before it is passed to the target program. For example, if the user follows the transaction ID with the characters LOOKUP, then the data passed to the target program is:

```
DATA=' STACK=(LOGON SASAK;LOOKUP)'
```

If the escape character appears more than once within a given data string, then only the first occurrence is replaced with the input data. If no input data is provided, the escape character is simply removed from the string passed to the target program.

The default DATA parameter value is "@" (at sign), which means that the string passed to the target program is the data provided by the user, or by the program that invokes TPFXFER.

If the "@" sign itself needs to be embedded in a DATA string, the ESCCHAR parameter can be used to change the escape character. Note that the DATA string is actually evaluated while the USERDEF statement in which it appears is being processed, during the creation of the TPFXTAB subtable. This means that the ESCCHAR value is the escape character currently in effect.

The following example illustrates this:

```
USERDEF GLOBAL,DATA='A$B*C,ESCCHAR='$'
  data is 'A$B*C', ESCCHAR is $
MERC USERDEF TRAN,ESCCHAR='*'
  data is 'A$B*C', ESCCHAR is '*'
AREC USERDEF TRAN,DATA='D$E'
  data is 'D$E', ESCCHAR is '$'
```

In this example, the DATA parameter associated with the MERC transaction is "A\$B*C". In this case, the escape character is "\$", since this is the value of the ESCCHAR parameter when the DATA statement was specified. The DATA string for the AREC transaction is "D\$E", and the escape character is again "\$".

If no data is to be passed to the target program, specify DATA=NO.

The maximum length of the data which can be passed to the target program depends on the construct used to send it:

1. Using the construct:

```
TPFFPGM=(DCALL,pgmname),DATA=<data>
```

the maximum length of data is 240 bytes.

2. Using the construct:

```
TPFFPGM=pgmname,DATA=<data>
```

the maximum length of data is 160 bytes.

Overriding Information

A user or programmer can override the parameters set in the TPFXTAB subtable by overriding them with screen input. The purpose of these overrides is to simplify certain testing situations, for example when a test system exists on a separate node from a production system, or if certain user IDs are handled

differently on the target system.

Override authorization and for which transaction(s) it is valid is defined by the OVER parameter in the TPFXTAB subtable.

If some overriding information is to be specified as screen input, certain points must be noted and certain conventions must be followed:

- When transparency program TPFXF46 is invoked directly using a transaction associated with it - for CICS - in the PCT, any data following the transaction ID on the screen is processed by TPFXF46.
- The data processed by TPFXF46 when it is invoked via a NATCICS-type XCTL is the string of data specified as part of the NAT CICS parameter.
- Overrides can be specified by preceding any data with override information, surrounded by parentheses, listed in the following order:

```
tranid (nodeid,userid,password,systemdata)userdata or
tranid (nodeid,userid,password,systemdata) userdata
```

where:

systemdata is the name of the program to be invoked directly.
userdata is the data to be passed to the program to be invoked.

For example:

```
atpf (tpf46,admin46,admin,nat21) fuser=(9,81)
```

Note that if the direct-call option is specified in the table for the transaction for which you override the program name, the direct-call remains in effect (see explanation of the TPFPGM parameter above).

If any parameter you are allowed to override is omitted within the parentheses, it must be substituted by a comma as placeholder. The value for the omitted parameter is then taken from the TPFXTAB subtable.

Examples:

```
tranid (,userid,password,systemdata) userdata
tranid (,,password,systemdata) userdata
tranid (,,,systemdata) userdata
```

Exceptions:

1. If only *nodeid* is specified to be overridden (OVER=NODEID) and you override it, no placeholder commas are required, for example:

```
tranid (nodeid)
```

2. If everything can be overridden (OVER=ALL), but you do not wish to override the *systemdata*, no placeholder comma is required for the *systemdata* operand, for example:

`tranid (,,password) userdata`

3. If you can override systemdata and userdata, but you do not wish to, no placeholder commas are required, for example

`tranid (,,password)`

- If you have specified the parameter `OVER=ALL` in the `TPFXTAB` subtable, you can override the `TPFPGM` with your screen input. If you override the program name using screen input, it is treated as if it was in the table for this transaction as follows:
- If it was specified as a direct-call in the table, you are directed straight to the program, and when you finish with the program, you are returned to your host environment (CICS or TSO);
- If it was not specified as direct-call in the table, (simply `TPFPGM=prog1`) and is to be overridden with `prog2`, then `prog2` is called directly. However, on finishing, you will see the Complete mode menu of `USTACK`, from where you can select any listed program, or log off using the `LOGOFF` command.

CICS: Special Considerations

This chapter covers the following topics:

- Printing Under TPF
- Upper/Lower Case Considerations

Printing Under TPF

Printing using a local CICS printer is accomplished by programs ACSBEG46 and ACSSPL46, which run under the CICS host system. ACSBEG46 is related to the transaction AB46, and ACSSPL46 to the transaction AP46.

Any printer to be used must be defined in the TIBTAB of the Com-plete target system with the related 4 Byte CICS *termid* and the corresponding CICS pseudo-DBID (see the ACSDEF subtable). For example:

```
TIBT TIBSTART NOTIBS=100
*
* Define a standard ACCESS printer
TIB 2,ACCESS,3288L,NAME=P001,GROUP=(ALL=NO),NODEID=127
* Define an SCS-ACCESS printer
TIB 3,ACCESS,3288L,NAME=P002,GROUP=(ALL=NO),NODEID=127,OPT=(SCS)
* Define a Graphics ACCESS printer
TIB 4,ACCESS,3287L,NAME=P003,GROUP=(ALL=NO),NODEID=127,OPT=(SCS)
TIBEND
```

Note that only *printers* must be defined in the TIBTAB. Com-plete allocates TIB entries for terminal devices dynamically.

Program ACSBEG46 communicates with the Com-plete target system to determine whether any printout is queued for this local CICS printer. If printout is queued, it starts the program ACSSPL46 which actually performs the printing on the host side.

If ACSBEG46 is started via the PLT entry suffix=*in*, it initiates itself with interval control, using a delay of one minute.

If any error occurs, for example, if the printer is not in service or has no ATI specified, the transaction AB46 does not attach the AP46 transaction and an appropriate error message is logged to the temporary storage queue TPF E.

You can read all messages with the browse command CEBR TPF E from CICS in the temporary storage called TPF E. If AB46 is started from a terminal, the message:

```
ACS5008 - ACSBEG nnn task(s) attached
```

is displayed at the terminal, where *nnn* is the number of AP46 tasks started.

Messages are also written to the transient data queue called TPF E, and this is printed in the SYSLOG of CICS if any destination is given in the DCT table for the name TPF E. The destination can be specified directly or indirectly in the table, so that the messages can be routed together with other messages to a data set, or you can specify a special data set for your TPF printer messages.

Note that ACSSPL46 is a standard CICS command level program and assumes that the correct CICS definitions have been made for any output device it uses. This is especially true in an MRO environment when ACSSPL46 runs in the AOR and the printers are serviced in the TOR. It is the responsibility of the CICS definitions to ensure that the output is routed to the remote printers correctly.

Upper/Lower Case Considerations

Pre CICS 3.3

By default, TPF runs in pseudo-conversational mode under CICS. In this mode, the user may encounter a well-known CICS problem relating to upper-case translation..

Upper-case translation in CICS is fixed throughout a session on a terminal basis, rather than being dynamic on an individual transaction basis. When CICS upper-case translation is in effect, the first screen received by a task is always translated to upper-case. In pseudo-conversational mode, this means that ALL screens are translated to upper-case since the ASIS option of an EXEC CICS RECEIVE applies only to the second and subsequent messages received for a transaction.

If, on the other hand, the user's terminal is defined to CICS without upper-case translation, then commands to initiate transactions (the *transid*) must be entered with the terminal shift key held down, since CICS does not recognize transaction IDs entered in lower-case.

The problem arises when the user's terminal has CICS upper-case translation enabled (FEATURE=UCTRAN in the TCT macro), but an individual transaction requires text to be entered and saved "as is".

ACSUCTR Transaction Level Program

TPF solves the problem of upper-case translation under CICS by calling a user-defined program called ACSUCTR to enable/reset upper-case translation.

A sample program ACSUCTR is provided on the TPF source data set. This example determines dynamically whether CICS upper-case translation is in effect for a terminal. If this is the case, the program deactivates upper-case translation for the terminal by modifying the terminal's TCTTE. The deactivation is in effect for the duration of the transaction.

The program works only if the terminal's TCTTE specifies UCTRAN. If this is not the case, ACSUCTR returns control immediately.

If ACSUCTR is in effect and a TPF program terminates abnormally, upper-case translation remains deactivated. ACSUCTR can be started manually by entering transid ACSU at the affected terminal.

Note:

You will know when this situation arises, as CICS will not recognize transactions typed in lower-case.

Important:

The program ACSUCTR and the transaction ACSU must be defined in the PPT and PCT respectively (see the installation instructions for the CICS environment in this documentation). If these definitions are not made, the TPF upper-case translation facility is disabled.

Post CICS 3.3

With CICS releases after version 3.3 the problem with upper-case translation has been resolved in CICS itself. It is possible to define upper-case translation for the terminal and transaction separately. Using Resource Definition Online (CEDA) you should define TERMTYPE UCTRAN (TRANID) and PROFILE UCTRAN (NO) for the TPF transactions. For more information about these parameters, please consult the appropriate CICS documentations.

The program ACSUCTR and transaction ACSU are redundant under CICS 3.3 and later, and should not be installed.

Transparency (CICS and TSO) and Reverse Access (CICS)

This chapter covers the following topics:

- Transparency Under CICS and TSO
 - Reverse Access (CICS only)
-

Transparency Under CICS and TSO

Transparency provides an automatic method of integrating the TPF component in an existing Natural CICS or TSO installation, as well as an easy way of implementing new transactions that invoke TPF.

- In the CICS environment, two programs are primarily responsible for controlling TPF interaction: the ACCESS46 program and the Transfer program.
- In the TSO environment, module TPFXFER includes the ACCESS program and the Transfer program.

In both environments, the function of these programs is the same.

The ACCESS Program

The ACCESS46 program is responsible for all of the actual interaction with the TPF system. In order to startup a TPF session, four items must be defined to the ACCESS program:

1. The node ID of the Com-plete system to be accessed; this name is defined in the ACSTBL subtable;
2. The user ID with which to sign on to Com-plete;
3. A password.
4. Data to be interpreted by the target system.

Note:

See also the section *User ID Considerations* for more information on user ID and password.

The Transfer Program

The Transfer program (named TPFXF46) is responsible only for simplifying the startup of a Com-plete session. The Transfer program uses a table (the Transfer table, TPFXTAB subtable), defined by the system administrator to automatically determine the proper node ID and user ID, and invokes ACCESS, passing over these items, including a dummy password. This means that:

1. The user does not need to perform an extra logon to Com-plete (since a CICS or external security user ID may have been used);

2. Programs that invoke Com-plete need not be modified to determine these items.

Programs written in current 3GL (COBOL, Assembler, PL/1) that pass control directly to Natural under CICS or TSO need not be changed to pass control to Natural under Com-plete. The 3 GL programs continue to run in the CICS address space/partition, but instead of transferring control to Natural, they can pass control to the Transfer program. This can be implemented by installing the Transfer program with the same name as that used for Natural under CICS or TSO. Based on the data passed to it under these circumstances, the transfer control program can determine the following items:

- The Natural program to be invoked;
- The data to be passed to that program;
- The name of a CICS or TSO program to which control is to be transferred when the Natural program has run. The node ID and user ID are determined using the definitions in the Transfer table. Thus, after the 3GL program transfers control to the Transfer program, Natural is invoked automatically under Com-plete, and the process is invisible (transparent) to the user and to the programmer.

A second function of the Transfer program is to simplify definition of new CICS transactions. If a transaction is defined in the PCT as invoking TPFXF46, and the proper information is placed into the Transfer table, a user can invoke TPFXF46 directly using that transaction code; the Transfer program determines all of the information needed to start up a Natural program under TPF. This eliminates the need for many 3GL front-ends.

Reverse Access (CICS only)

Reverse Access is another feature of TPF transparency and is available for CICS sites only. Many Natural programs currently under CICS call 3GL subroutines to perform certain functions. If these subroutines do not directly use any CICS facilities, then they can simply be installed to run in the TPF address space/partition. However, if a 3GL subroutine uses CICS for any reason, then it must run in the CICS address space/partition.

A facility is required to arbitrate between the Natural program, running in the target system, and the subprogram, running in the host system. The Subprogram Transparency feature of TPF performs this function and thus simplifies the implementation of TPF in a Natural CICS implementation.

There are two variants of Reverse Access:

- **Reverse Access with Immediate Return**

If a subroutine that returns immediately needs to be invoked using Reverse Access, then a "stub program" with the same name as the subroutine is installed in the TPF system. This stub program consists of two lines of Assembler language and is used mainly to pass control.

When a Natural program calls the stub program, TPF passes information to the ACCESS program (running under CICS), which in turn invokes the actual subroutine. When the subroutine has completed, it returns control to ACCESS, which in turn will return control to TPF and the Natural program.

- **Reverse Access with Delayed Return**

If a subroutine that performs non-conversational functions needs to be invoked using Reverse Access, then a user-written "stub program" must be called. This program must issue a pseudo screen I/O which is interpreted by ACCESS to invoke the required subroutine. Before linking to the

user-subroutine, ACCESS saves its CICS environment in temporary storage and passes to the user-subroutine the name of a program to be invoked by the subroutine on termination. On being called, this program restores the ACCESS environment and continues the Reverse Access calls, returning to the original caller of the Reverse Access function.

Calling Reverse Access with Immediate Return

Installing TPFSTUBM

TPFSTUBM calls interface routines to perform TPF-specific functions. These subroutines were installed during the installation of the Com-plete target system. An example job to link TPFSTUBM is provided in the source library (member JCLLSTUB).

The linked program TPFSTUBM must exist in the residentpage space of Com-plete. This can be accomplished by including the following startup parameter in the sysparms:

```
RESIDENTPAGE=TPFSTUBM
```

CICS Subprogram Preparation

Transparency can be invoked for a given subprogram only if that subprogram exists in the CICS load library, and has an associated PPT entry.

Stub Program

The source for the stub program consists of two lines of Assembler code:

- The first line invokes the TPFSTUBC macro;
- The second line is the END statement.

The label of the TPFSTUBC statement must be the name of the program. The syntax for the entire stub program is illustrated below: `progname TPFSTUBC END`

where *progname* is the program name.

This program must be assembled and linked into a library contained in the Com-plete COMPLIB concatenation.

Determining Parameter Lengths

Using information passed by Natural, the stub program can determine parameter lengths.

However, sometimes the CALL statement may designate only the first field of a contiguous area which is being passed as a single parameter. Natural interprets the parameter length as that of the field specified on a CALL statement, and thus passes an incorrect length to the stub program.

When coding the TPFSTUBC statement for such a stub program, you can override the length information being passed by Natural by listing the various parameter lengths (enclosed by parentheses) after the TPFSTUBC designation.

For example:

```
Natural program call
  01 PARM1 (A80)
  01 PARM2 (A80)
  01 PARM3
      02 FLD1 (A2)
      02 FLD2 (A5)
      02 FLD3 (N7)
      02 FLD4 (B4/1:2000)
  .
  .
  .
  CALL 'CICSPGM' PARM1 PARM2 PARM3
  .
  .
  .
```

The corresponding stub program definition is:

```
CICSPGM TPFSTUBC (80,80,2,5,7,8000)
```

Calling the Subprogram from Natural

Once the stub program is installed, Subprogram Transparency can be used to invoke the subprogram from Natural running under Com-plete. When Natural issues the CALL for the subprogram, the stub program is invoked:

```
CALL 'progname'
```

The stub program, along with the TPFSTUBM program, determines the parameters passed by Natural and passes the appropriate data over to CICS, where the actual subprogram is invoked. When the subprogram is finished, control returns to Com-plete, and the Natural program continues.

Data can be passed in either direction.

Sample programs (TESTRE and TESTREV) are included on the source dataset. A sample Natural program is provided below:

```
0010 DEFINE DATA LOCAL
0020   01 ALLPARMS
0030   02 A(A250)
0040   02 REDEFINE A
0050     03 AA(A20)
0060   02 B(A250)
0070   02 REDEFINE B
0080     03 BB(A20)
0090   02 C(A250)
0100   02 REDEFINE C
0110     03 CC(A20)
0120 END-DEFINE
0130 MOVE ALL '1' TO A
0140 MOVE ALL '2' TO B
0150 MOVE ALL '3' TO C
0160 DISPLAY 'BEFORE CALL TO REVERSE ACCESS'
0170 DISPLAY AA BB CC
```

```

0180 CALL 'TESTRE' ALLPARMS
0190 DISPLAY 'AFTER CALL TO REVERSE ACCESS'
0200 DISPLAY AA BB CC
0210 END

```

Calling Reverse Access with Delayed Return

If you need to call CICS subroutines (via Reverse Access) that use non-conversational functions under CICS, you must code and install an interface subroutine (a so-called "stub program").

An example Assembler program is supplied on the source data set (member ACSHXC). This program calls and passes data to a user program called USERPGM. The source data set also contains an example program with this name.

You must assemble this sample program and link it into a data set contained in the Complete COMPLIB concatenation.

Program USERPGM runs under CICS and can perform non-conversational screen I/Os. It ends with an EXEC CICS RETURN TRANSID(XTPF) or the transaction name you specified in the ACSDEF subtable for the internal continuation transaction.

The CICS application program USERPGM must be linked with the CICS stub programs DFHEAI and DFHEAI0 from the CICS loadlib. You must define the resulting module to CICS with CEDA, or include it in the PPT.

CICS program USERPGM obtains control with the following data:

```

4 byte ID      'HXCC'      IDENTIFIER
8 byte pgmname 'ACCESS46' RETURN PROGRAM
      user data

```

Exits for the TPF Component

This chapter covers the following topics:

- USERID exit
 - NODEID exit
 - TPFXFEX exit
 - Exits in ACCESS46
-

USERID exit

This exit is used to determine the node ID to be used to logon to the Complete target. It is invoked if EXIT is specified for the NODEID parameter in the TPFXTAB subtable, (USERDEF) in ACSTAB.

How to use the USERID exit

Entry is via standard linkage conventions.

Upon entry to the USERID exit, a set of parameters is received in the form of fullword addresses pointed to by R1.

All registers must be restored to their original values before control is returned from the exit to the calling program.

R1	The address of the parameter list.
R13	The address of an 18-fullword savearea.
R14	The return address.
R15	The entry point address of the exit.

Parameters

The first address is the address of a 40-fullword work area that can be used as working storage by the exit. It is recommended that the first 18-fullwords of this area be used as a savearea.

The second address is the address of the appropriate entry in the transparency subtable (TPFXTAB). The layout of this entry is defined in the DSECT in the copy book CCACSTAB.

The third address is the address of an 8-byte field where the exit should place the node ID to be used for communication with the target Complete system.

Linkage

Change the supplied linkage for TPFXF46 so that the pre-defined exit name TPFXNTCM is overridden by the name of your choice. Insert the following lines between the INCLUDE SYSLIB(TPFXCICS or TPFXTSO) and INCLUDE SYSLIB(TPFX01):

```
CHANGE userexit(TPFXNODE)
INCLUDE USERLIB(userexit)
```

The resulting linkage will return a condition code of 4.

NODEID exit

This exit is used to determine the node ID to be used to logon to the Com-plete target. It is invoked if EXIT is specified for the NODEID parameter in the TPFXTAB subtable, (USERDEF) in ACSTAB.

How to use the NODEID exit

Entry is via standard linkage conventions.

Upon entry to the NODEID exit, a set of parameters is received in the form of fullword addresses pointed to by R1.

All registers must be restored to their original values before control is returned from the exit to the calling program.

R1	The address of the parameter list.
R13	The address of an 18-fullword savearea.
R14	The return address.
R15	The entry point address of the exit.

Parameters

The first address is the address of a 40-fullword work area that can be used as working storage by the exit. It is recommended that the first 18-fullwords of this area be used as a savearea.

The second address is the address of the appropriate entry in the transparency subtable (TPFXTAB). The layout of this entry is defined in the DSECT in the copy book CCACSTAB.

The third address is the address of an 8-byte field where the exit should place the node ID to be used for communication with the target Com-plete system.

Linkage

Change the supplied linkage for TPFXF46 so that the pre-defined exit name TPFXNODE is overridden by the name of your choice. Insert the following lines between the INCLUDE SYSLIB(TPFXCICS or TPFXTSO) and INCLUDE SYSLIB(TPFX01):

```
CHANGE userexit(TPFXNODE)
INCLUDE USERLIB(userexit)
```

The resulting linkage will return a condition code of 4.

An example of TPFXNODE is delivered in the source library.

TPFXFEX exit

This exit is invoked after CICS/TPFXTSO has built the data to be passed to the ACCESS program.

How to use TPFXFEX exit

Entry is via standard linkage conventions.

Upon entry to TPFXFEX, a set of parameters is received in the form of fullword addresses pointed to by R1.

All registers must be restored to their original values before control is returned from the exit to the calling program.

R1	The address of the parameter list.
R13	The address of an 18-fullword savearea.
R14	The return address.
R15	The entry point address of the exit.

Parameters

The first address is the address of a 40-fullword work area that can be used as working storage by the exit. It is recommended that the first 18-fullwords of this area be used as a savearea.

The second address is the address of the appropriate entry in the transparency subtable (TPFXTAB). The layout of this entry is defined in the DSECT in the copy book CCACSTAB.

The third address is the address of the area which will be passed to the ACCESS program.

Linkage

Link the TPFXFEX exit with CICS/TPFXTSO directly. Insert the following line after the INCLUDE SYSLIB(TPFXF01):

```
INCLUDE USERLIB(TPFXFEX)
```

Exits in ACCESS46

Please refer to the description of the ACCESS exits ACSUUEX1 and ACSUUEX2 in the Complete System Programming documentation.

Starting Access from a User Program

If for some reason you do not wish to use the normal transparency procedure (TPFXF46) to initialize to TPF, you may call the ACCESS program directly from a user written routine. A sample program to do this is provided on the source dataset (ATPFINIT). The ACCESS program expects parameters to be passed in a data area (mapped by CCACSCMN).

The first init call to TPF must always contain the string ITPF in the access common area to indicate that it is the init call to TPF.

To activate the ATPFINIT program, assemble and link the ATPFINIT program (see sample link job JCLLNKEX in the source library). The module must be linked to the TPF.CICSLOAD. The transaction calling the ATPFINIT program must be defined as normal with CEDA or in the PCT.