

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 chapter 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 Complete 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 specified 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 Complete.

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 Complete 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 Overriding Information later in this chapter).
(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 Exits for TPF in this chapter.

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 Overriding Information later in this chapter).
EXIT	Use user exit to determine user ID (CICS only). For more information, see the section Exits for TPF in this chapter.
(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 TERMD 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 the section **User ID Considerations** in the installation chapter.

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.
	Notes: 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** below.

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 ENDCCHAR 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, unknwn 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=user id

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 in this chapter.

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 "at" 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:

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

the maximum length of data is 240 bytes.

2. Using the construct:

```
TPFPGM=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.