CICS Transaction Server for z/OS

A MVS BatchPipes Stage interface to CICS using EXCI

Robert Harris,
CICS Technical Strategist,
IBM Hursley.

Issued:	31 August 2001
Revision Date:	31 August 2001
Previous Revision Date:	None
Review Date:	As Required

Take Note!

Before using this document be sure to read the general information under "Notices".

First Edition, August 2001.

Notices:

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS-IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Trademarks:

The following are Trademarks of International Business Machines Corporation in the United States, in other countries, or both:

3090
ACF/VTAM
AD/Cycle
AFP
AIX
AnyNet
Application System/400
APPN
AS/400
AT
BookManager
C Set++
C/370
C/MVS
CBIPO
CBPDO
CICS
CICS/400
CICS/6000
CICS/ESA
CICS/MVS
CICS OS/2
CICS TS
CICS/VM
CICS/VSE
CICSPlex
CICSPlex SM
COBOL/370
COBOL/2
Common User Access
CUA
DATABASE 2
DB2
DFSMS
DFSMS/MVS
DFSMSdfp
DFSMSdss
DFSMShsm
DFSORT
DXT
eNetwork
Enterprise Systems Architecture/370
Enterprise Systems Architecture/390
ES/3090
ESA/370
ESA/390
ES/9000
ESCON
GDDM
HiperBatch
Hiperspace
InfoWindow
IBM
IBMLink
IMS
IMS/ESA
Language Environment
MQ
MQSeries
MVS
MVS/DFP
MVS/ESA
MVS Parallel Sysplex
MVS/SP
MVS/XA
Multiprise
NetView
OpenEdition
OS/2
OS/390
OS/400
Processor Resource/Systems Manager
Parallel Sysplex
PR/SM
Presentation Manager
RACF
Resource Measurement Facility
RETAIN
RISC System/6000
RMF
RT
S/370
S/390
SAA
SQL/DS
SP
System/36
System/38
System/360
System/370
System/390
SystemView
Systems Application Architecture
VisualAge
VSE/ESA
VTAM
WebExplorer
z/OS

UNIX is a registered Trademark in the United States and other countries licensed exclusively through X/Open Company Limited

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

INTEL is a registered trademark of Intel Corporation, in the United States, or other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, or other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Summary of amendments

Date Of Change	Change made to document
31/08/2001	Creation in SupportPac CA1J

Reference Material and Bibliography:

This document uses a short reference to the following documentation:

Short reference	Book Title
CICS RDO Book	CICS Resource Definition Guide SC34-5722
CICS SDG	CICS System Definition Guide SC34-5725
CICS CG	CICS Customization Guide SC34-5706
CICS EXT	CICS External Interfaces Guide SC33-1944
CICS APR	CICS Application Programming Reference SC34-5702
CICS SPR	CICS System Programming Reference SC34-5726
Pipe Guide	IBM BatchPipes OS/390 V1R2 BatchPipeWorks User Guide SA22-7457
Pipe UG	IBM BatchPipes OS/390 V1R2 Users Guide and Reference SA22-7458
Pipe Ref	IBM BatchPipes OS/390 V1R2 BatchPipeWorks Reference SA22-7456

Required Prerequisite SupportPacs:

The following SupportPacs are required to run the code distributed in this SupportPac:

SupportPac	Relevance
CA1I	The underlying 'Rexx/MVS Interface to CICS using EXCI' code which is used within this SupportPac

Preface:

This SupportPac consists of an utility that enables a MVS BatchPipe to access to CICS function. It does this by running an EXCI Connection to a CICS Transaction Server for z/OS region using the code supplied in SupportPac CA1I.

This SupportPac contains information on:

- Pipe Input and Output Streams
- How to incorporate information from CICS in a Pipe Stage

It can be used to:

- Incorporate CICS-sourced information into Batch Pipe processing

You need to understand RDO configuration for CICS and have a general familiarity with MVS Pipes to get the best out of this SupportPac.

The information and code in this SupportPac is **only** applicable to CICS Transaction Server for z/OS. It is not applicable to earlier CICS releases.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1: The Concept

This SupportPac provides a method for a MVS Batch Pipe to access CICS Transaction Server for z/OS facilities via the EXCI Interface.

This permits a Pipe Stage to issue a CICS Distributed Program link to a CICS region in the same MVS. A named program is executed within the CICS region with data sent from the Primary Input Stream in a Commarea. The results of the execution are returned from CICS in the Commarea and made available to the caller as the Primary Output Stream.

The name of the Pipe stage is **RXDPLP**.

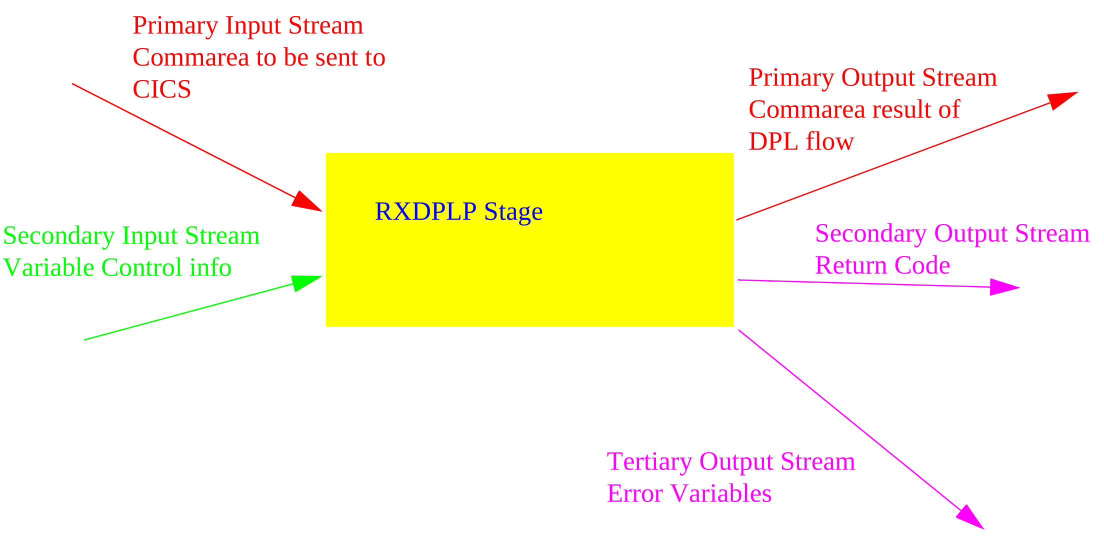


Figure 1: The RXDPLP Stage

The Secondary Input Stage is optional and can be used to supply communication settings in a variable fashion.

The Secondary and Tertiary Output Streams are optional and contain error information for the EXCI flow to CICS.

Chapter 2: Installation

2.1: SupportPac prerequisites

You must have installed SupportPac CA1I as this SupportPac uses the RXDPL interface supplied by CA1I. This document refers to CA1I documentation.

2.2: CICS prerequisites

The code in this SupportPac uses various standard CICS EXCI facilities. You must have configured the CICS region for this type of access. The things you have to do for this are contained in the *'External CICS Interface'* part of the *CICS EXT* book.

In particular, you must:

- Code the required DFHXCOPT EXCI Interface table
- Code up an optional DFHXCURM User-replaceable module
- Supply RDO definitions for EXCI Connections

2.2.1: Coding the DFHXCOPT Table

The DFHXCOPT table defines the connectional characteristics of the EXCI linkage to CICS. Although it controls tracing and other debugging facilities, the main usage to name the SVC being used for communication with CICS. The SVC number to use is that given for **CICSSVC** in the SIT.

See *'External CICS interface options table, DFHXCOPT'* in the *CICS EXT* for full information including how to compile the table.

The source for your DFHXCOPT should look something like:

```
      DFHXCO TYPE=CSECT,                                           |
            CICSSVC=245,                                            |
            CONFDATA=SHOW,                                          |
            DURETRY=30,                                             |
            GTF=ON,                                                 |
            MSGCASE=MIXED,                                          |
            SURROGCHK=NO,                                           |
            TIMEOUT=60000,                                          |
            TRACE=2,                                                |
            TRACESZE=16,                                            |
            TRAP=OFF
      END DFHXCOPT
```

where SVC 245 is being used for communication.

This table must be assembled and linked into a library which is in the `//STEPLIB` concatination of the Pipe job.

2.2.2: Coding the DFHXCURM Module

The DFHXCURM is a CICS user-replaceable module that can be used to set defaults used for EXCI Communication. You do not have to code a DFHXCURM, but if you do so, it must be placed in the `//STEPLIB` concatination for the Pipe Step.

Full details are in *'The EXCI user-replaceable module'* section of the *CICS EXT*.

The main use of DFHXCURM is to permit defaults to be specified for communication parameters or to police those that are supplied by the user. The `RXDPLP` Stage has facilities to require the use of DFHXCURM to set parameters.

However, DFHXCURM is not needed for normal usage of this SupportPac, and it is recommended that you do not supply one unless necessary.

2.2.3: Defining the EXCI RDO Connections

A RDO CONNECTION definition and its associated SESSIONS must be active in the region for Rexx usage of EXCI. You have to code these RDO items in a group and ensure that this Group in is a suitable LIST (or is manually installed).

The CONNECTION definition names the EXCI pathway between the Rexx/MVS Exec and the CICS region, whilst the SESSIONS define the number of active pathways. See the *CICS RDO* Book for full information on specifying CONNECTION and SESSIONS, and the *'Defining connections to CICS'* section of the *CICS EXT*.

This SupportPac uses a GENERIC EXCI connection to the CICS region. Consequently, the CONNECTION definition will look something like:

```
 CONnection     : PIPES
 Group          : RAHEXCI
 DEscription    : LINKAGE FROM PIPES
CONNECTION IDENTIFIERS
 Netname        :
 INDsys         :
REMOTE ATTRIBUTES
 REMOTESYSTem   :
 REMOTEName     :
 REMOTESYSNet   :
CONNECTION PROPERTIES
 ACcessmethod   : IRc                Vtam | IRc | INdirect | Xm
 PRotocol       : Exci               Appc | Lu61 | Exci
 Conntype       : Generic            Generic | Specific
 SInglesess     : No                 No | Yes
 DAtastream     : User               User | 3270 | SCs | STrfield | Lms
 RECordformat   : U                  U | Vb
 Queuelimit     : No                 No | 0-9999
 Maxqtime       : No                 No | 0-9999
OPERATIONAL PROPERTIES
 AUtoconnect    : No                 No | Yes | All
 INService      : Yes                Yes | No
SECURITY
 SEcurityname   :
 ATtachsec      : Local              Local | Identify | Verify | Persistent
                                     | Mixidpe
 BINDPassword   :                    PASSWORD NOT SPECIFIED
 BINDSecurity   : No                 No | Yes
 Usedfltuser    : No                 No | Yes
RECOVERY
 PSrecovery     :                    Sysdefault | None
 Xlnaction      : Keep               Keep | Force
```

The associated SESSIONS definition will look like:

```
 Sessions        : RAHEXCIP
 Group           : RAHEXCI
 DEscription     : LINK TO PIPES
SESSION IDENTIFIERS
 Connection      : PIPES
 SESSName        :
 NETnameq        :
 MOdename        :
SESSION PROPERTIES
 Protocol        : Exci                 Appc | Lu61 | Exci
 MAximum         : 000 , 000            0-999
 RECEIVEPfx      : RX
 RECEIVECount    : 005                  1-999
 SENDPfx         :
 SENDCount       :                      1-999
 SENDSize        : 04096                1-30720
 RECEIVESize     : 04096                1-30720
 SESSPriority    : 000                  0-255
 Transaction     :
OPERATOR DEFAULTS
 OPERId          :
 OPERPriority    : 000                  0-255
 OPERRsl         : 0
 OPERSecurity    : 1
PRESET SECURITY
 USERId          :
OPERATIONAL PROPERTIES
 Autoconnect     : All                  No | Yes | All
 INservice       : Yes                  No | Yes
 Buildchain      : Yes                  Yes | No
 USERArealen     : 000                  0-255
 IOarealen       : 04096 , 04096        0-32767
 RELreq          : No                   No | Yes
 DIscreq         : No                   No | Yes
 NEPclass        : 000                  0-255
RECOVERY
 RECOVOption     : Sysdefault           Sysdefault | Clearconv | Releasesess
                                        | Uncondrel | None
 RECOVNotify     : None                 None | Message | Transaction
```

Observe that this definition only has Receive-type terminals and 4k is used for buffering of the flows. 5 concurrent connections into CICS are allowed.

2.3: Installing the SupportPac

2.3.1: Unpacking the ZIP file

Unzip `the CA1J.ZIP` file. It contains:

- `CA1JCLIR.BIN` a Rexx Exec that provides the RXDPLP Pipe Stage for this SupportPac in an unloaded MVS PDS Exec format

The SupportPac contains a single Rexx/MVS Exec called `RXDPLP` which uses `RXDPL` from Supportpac CA1I. `RXDPL` must already be installed to use this SupportPac.

2.3.2: Copying the PDS to MVS

You should copy the unloaded MVS PDS **CA1JCLIR.BIN** to MVS.

You can do this operation via file transfer under TSO. Follow these instructions exactly or else the transfer will not work (It is assumed that IBM Personal Communications is being used as the 3270 emulator).

- Define (if not already present) a transfer type with the ASCII, CRLF and APPEND checkboxes unselected, the LRECL set to 80 with the transfer using FIXED length records; I call this the LOADLIB transfer type
- Transfer the `CA1JCLIR.BIN` file to MVS with a file name of CA1JCLIC using the LOADLIB transfer type. This will create a flat file called `CA1JCLIC` in MVS
- In TSO, issue a RECEIVE INDSN(CA1JCLIC); when prompted for a dataset name reply DSN(CA1JCLIS). A PDS called CA1JCLIS should be created in MVS containing the RXDPLP Exec.

Alternatively, you can use FTP to transfer the PDS.

- Use the `FTP` command to start a session to the MVS region
  - Issue a `BINARY` command to prevent corruption of the data
  - Issue a `QUOTE SITE RECFM=FB` to transfer in the correct layout
  - Issue a `QUOTE SITE LRECL=80` to transfer in the correct layout
  - Send the file via `PUT CA1JCLIR.BIN CA1JCLIC`
- In TSO, issue a RECEIVE INDSN(CA1JCLIC); when prompted for a dataset name reply DSN(CA1JCLIS). A PDS called CA1JCLIS should be created in MVS containing the RXDPLP Exec.

Once this `CA1JCLIS` PDS has been created, copy the RXDPLP member to a Library in the `//REXX` concatination for your Pipe Step.

2.3.3: Installing within Pipes

You must include in the //STEPLIB concatination the CICS-supplied SDFHEXCI library which contains the EXCI interface routines and the library that contains the DFHXCOPT table. In this concatination must be a library containing RXDPL obtained from SupportPac CA1I.

There is nothing special that has to be done within a Pipe Step to invoke the RXDPLP Stage that provides the function for this interface.

As long as RXDPLP is placed in a Library which is in the //REXX concatination all should be well.

The JCL for a Pipe Step should look like:

```
//RUNP     EXEC PGM=PIPE,
//              PARM='Pipe function'
//REXX     DD  DSN=<library containing RXDPLP>,DISP=SHR
//STEPLIB  DD  DSN=<library containing RXDPL>,DISP=SHR,DCB=BLKSIZE=32760
//         DD  DSN=<library containing DFHXCOPT>,DISP=SHR
//         DD  DSN=CICS.SDFHEXCI,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//PIPEPOUT DD  SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//PIPEOUT  DD  SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSTSPRT DD  SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
/*
```

Chapter 3: Operational Characteristics of the RXDPLP Stage

3.1: RXDPLP as a Stage Command

RXDPLP is a Stage Command for Pipe usage. The Commarea sent to CICS is taken from the Primary Input Stream and the results of the DPL flow returned as the Primary Output Stream.

The Secondary Input Stream is optional and can be used to supply settings dynamically to the RXDPLP Stage. This Stream can be processed on a record-by-record basis to permit each flow to CICS to have different characteristics, or a single record can apply for all usages of the Stage.

Secondary and Tertiary Output Streams are optional and are used to convey error information for the EXCI flow.

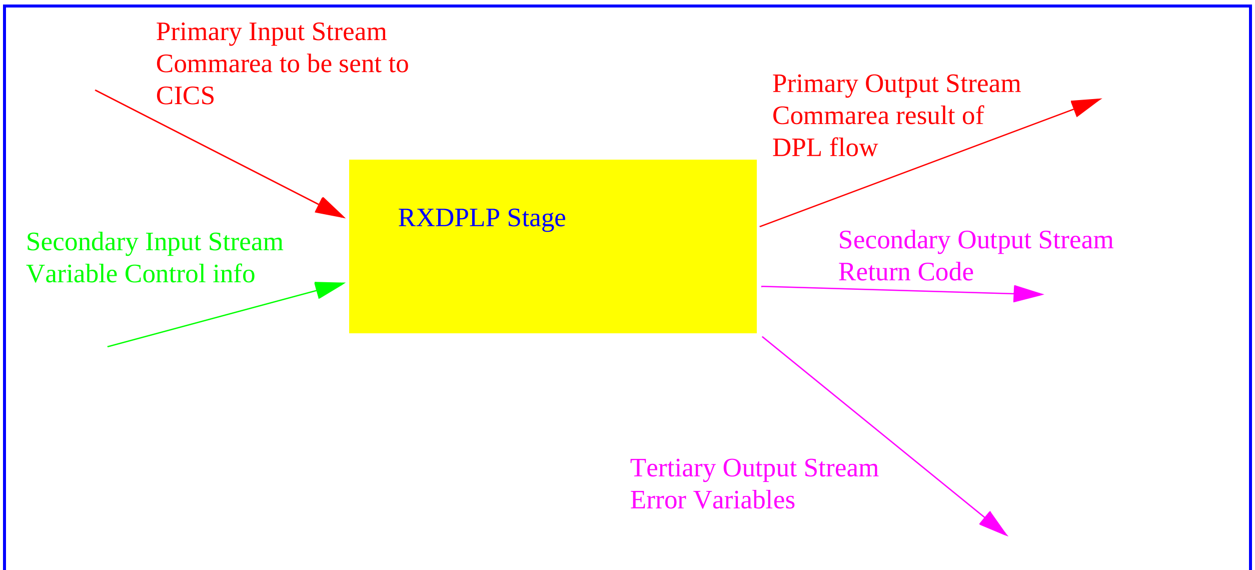


Figure 2: Streams and RXDPLP

3.2: RXDPL from SupportPac CA1I

The underlying mechanism for running the EXCI Communication with CICS uses the RXDPL module from SupportPac CA1I. Consult that documentation for full details of the operational characteristics of the interface.

3.3: Use of EXCI facilities via RXDPL

RXDPL uses the six native EXCI interfaces as defined in the *CICS EXT* book under the *The EXCI Call Interface* section.

A new EXCI Pipe is obtained for each usage of RXDPL. This means that each flow engendered by a record in the Primary Input Stream for the RXDPLP Stage will initiate a new Connection.

If one of the `Initialize_User`, `Allocate_Pipe` or `Open_Pipe` calls fail, then subsequent ones will not be called and the `DPL_Request` flow will not be made. In all cases, an attempt will be made to cleanly end EXCI communication by issuing the `Close_Pipe` and `Deallocate_Pipe` calls. A failure will cause the RXDPLP Stage to stop processing Input Stream records. Error information is available via the Secondary and Tertiary Output Streams.

3.4: Unit of Work Considerations

RXDPLP will never associate itself with the MVS (or any other) Recovery Manager. All flows to CICS are made with the `SYNCONRETURN` setting.

3.5: Connection and CICS Considerations

RXDPLP uses a Generic EXCI Connection to CICS (see Section 2.2.3 "Defining the EXCI RDO Connections" on page 5).

The relevant CICS region must:

- Have the required RDO definitions active (see Section 2.2.3 "Defining the EXCI RDO Connections" on page 5)
- Be accepting VTAM traffic (`CEMT SET VTAM OPEN`)
- Be accepting MRO Traffic (`CEMT SET IRC OPEN`)
- Have the Mirror Transaction and DPLed Program active and available
- Be authorised to accept function for the relevant Userid

3.6: Pipe Commit Level

RXDPLP does not explicitly use `COMMIT` level processing.

Chapter 4: Stage Streams

4.1: Primary Input Stream

The Primary Input Stream for the RXDPLP Stage contains the Commarea that is to be sent to CICS. The physical size of the commarea used in EXCI processing is always 32500 bytes, but the length of the current Primary Input Stream record is used within the EXCI flow. Zero-length Primary Input Stream records or those greater than 32500 bytes will raise an error.

4.2: Primary Output Stream

The Primary Output Stream for the RXDPLP Stage contains the Commarea sent from CICS as a result of running the DPL processing. Primary Output Stream records will always be 32500 bytes long. If processing results in an error the Primary Output Stream record will be a copy of the Primary Input Stream record extended to 32500 bytes.

4.3: Secondary Input Stream

The Secondary Input Stream is optional. It is used to convey Control information in a dynamic fashion. This Control information is usually specified on the RXDPLP Stage command, but can also be specified at runtime through the Secondary Input Stream.

If the Secondary Input Stream is being used, there will usually be one Secondary Input Stream record for the RXDPLP Stage as the Control Parameters will usually be fixed for the Stage. However, each Primary Input Stream record (and so each flow to CICS) can have an associated Secondary Input Stream record to supply Connectional details.

4.4: Secondary Output Stream

The Secondary Output Stream is optional. If defined, there will be one record per Primary Input Stream record. The Secondary Output Stream conveys the Return Code String from the underlying RXDPL that runs the EXCI communication to CICS.

4.5: Tertiary Output Stream

The Tertiary Output Stream is optional, but if required the Secondary Output Stream must also be defined. The Tertiary Output Stream contains further diagnostic information for the EXCI flow garnered from the OutputCommareaStem. returned components from the underlying RXDPL.

Chapter 5: The RXDPLP Stage

The RXDPLP Stage Command has both positional and optional parameters

```
streams: RXDPLP applid prog tran userid
                / NOABEND|ABEND RETAIN|NORETAIN NOTRACE|TRACE /
```

The four positional parameters can be specified

Statically — The setting is hard coded on the Stage command

Dynamically — The setting can be taken from a Secondary Input Stream record on either a positional or word basis

Defaultly — The setting is assumed to be fixed up by DFHXCURM processing

An example stage is:

```
RXDPLP IYCKRAH6 UTPROGB * *
```

which runs communication to the `IYCKRAH6` CICS region passing as a Commarea a Primary Input Stream record to program `UTPROGB` which is run under the default Mirror Transaction (`CSMI`) with a Userid of that being used by the current job to produce a returned Commarea which is available in the Primary Output Stream.

`RXDPLP` cannot be the first Stage of a Pipeline (you cannot send a null-commarea to CICS) nor does it operate `STAGE` processing.

5.1: Specifying the Streams

Streams are specified on Pipe Stages via Labels in conjunction with the `endchar` setting. Section *2.7 Defining Multistream Pipelines through Labels* in the *Pipe Guide* shows how this is done.

The Label on a Pipe Stage shows that additional Input or Output Streams are active for the Stage. These additional streams are specified after the `endchar` delimiter.

Each occurrence of the Label defines another Input or Output Stream (the first occurrence is the Secondary Input or Output Stream, the second the Tertiary Input or Output Stream etc.). If there is something before the Label, that portion defines an Input Stream: if there is something after the Label, an Output Stream is defined. Usual Stage processing applies before the Label to manipulate the Input Stream or after the Label to process the Output Stream.

Here is an example of Streams definition:

```
PIPE (SEP ; end ?)
       literal Primary Input Stream record
 ; ss: PipeStage
 ;     specs /Primary Output Stream :/ 1 1-* next ; cons
 ?
   literal Secondary Input Stream record
 ; ss:
 ; specs /Secondary Output Stream :/ 1 1-* next ; cons
 ?
   ss:
 ; specs /Tertiary  Output Stream :/ 1 1-* next ; cons
```

The Pipe Stage has a Primary Input Stream generated from a `LITERAL` Stage. As it has a Label of `ss:` other Input and Output Streams are provided.

After the ? `endchar` delimiter the first `ss:` defines Secondary Streams. As there is a `LITERAL` stage in front of the Label, there is a Secondary Input Stream. As there is a Stage after the Label, there is a Secondary Output Stream which is processed (appending the text in this case).

The same Label occurs after the next ? delimiter, so defining Tertiary Items. This time the Label does not have any Stages in front of it, so a Tertiary Input Stream is not active. There is processing after the Label so a Tertiary Output Stream is present and processed.

5.2: Specifying the CICS region

The VTAM applid of the CICS region into which the EXCI request will be passed is specified as the first parameter on the stage.

```
streams: RXDPLP applid prog tran userid
                / NOABEND|ABEND RETAIN|NORETAIN NOTRACE|TRACE /
```

The Applid can be specified via:

8 byte hard coded name	The applid is supplied on the Stage and applies to all flows through the Stage
(Cn)	The applid is to be taken from a Secondary Input Stream record starting in column n for 8 bytes. The start of a Secondary Input Stream record is at column 1. If the RETAIN option is active (the default) then the setting will be taken from the first Secondary Input Stream record for all flows through the Stage. If NORETAIN is active, each flow through the Stage will cause a new record to be read from the Secondary Input Stream to supply the setting.
(Wn)	The applid is to be taken from a Secondary Input Stream record from word n. The start of a Secondary Input Stream record is word 1. If the RETAIN option is active (the default) then the setting will be taken from the first Secondary Input Stream record for all flows through the Stage. If NORETAIN is active, each flow through the Stage will cause a new record to be read from the Secondary Input Stream to supply the setting.
.	Supply this parameter as blanks on the assumption that DFHXCURM will fix it up to point to the required CICS region (this is a dot)

If the CICS region is not contactable, an error will result.

If this parameter is omitted (which means that `prog`, `tran` and `userid` are also omitted), then `applid`, `prog`, `tran` and `userid` are passed as spaces, so implying that DFHXCURM will fixup the settings.

5.3: Specifying the Program

The program to be executed within the CICS region is specified as the second parameter on the stage.

```
streams: RXDPLP applid prog tran userid
                / NOABEND|ABEND RETAIN|NORETAIN NOTRACE|TRACE /
```

The Program can be specified via:

8 byte hard coded name	The program is supplied on the Stage and applies to all flows through the Stage
(Cn)	The program is to be taken from a Secondary Input Stream record starting in column n for 8 bytes. The start of a Secondary Input Stream record is at column 1. If the RETAIN option is active (the default) then the setting will be taken from the first Secondary Input Stream record for all flows through the Stage. If NORETAIN is active, each flow through the Stage will cause a new record to be read from the Secondary Input Stream to supply the setting.
(Wn)	The program is to be taken from a Secondary Input Stream record from word n. The start of a Secondary Input Stream record is word 1. If the RETAIN option is active (the default) then the setting will be taken from the first Secondary Input Stream record for all flows through the Stage. If NORETAIN is active, each flow through the Stage will cause a new record to be read from the Secondary Input Stream to supply the setting.
.	Supply this parameter as blanks on the assumption that DFHXCURM will fix it up to point to the required program (this is a dot)

If the program is not available within the CICS region, an error will be raised.

If this parameter is omitted (which means that `tran` and `userid` are also omitted), then `prog`, `tran` and `userid` are passed as spaces, so implying that DFHXCURM will fixup the settings.

5.4: Specifying the Mirror Transaction

The Mirror Transaction to run the program in the CICS is specified as the third parameter on the stage.

```
streams: RXDPLP applid prog tran userid
                / NOABEND|ABEND RETAIN|NORETAIN NOTRACE|TRACE /
```

The Mirror Transaction can be specified via:

*	Use the default Mirror Transaction of CSMI (this is a star)
4 byte hard coded name	The Mirror Transaction is supplied on the Stage and applies to all flows through the Stage
(Cn)	The Mirror Transaction is to be taken from a Secondary Input Stream record starting in column n for 4 bytes. The start of a Secondary Input Stream record is at column 1. If the RETAIN option is active (the default) then the setting will be taken from the first Secondary Input Stream record for all flows through the Stage. If NORETAIN is active, each flow through the Stage will cause a new record to be read from the Secondary Input Stream to supply the setting.
(Wn)	The Mirror Transaction is to be taken from a Secondary Input Stream record from word n. The start of a Secondary Input Stream record is word 1. If the RETAIN option is active (the default) then the setting will be taken from the first Secondary Input Stream record for all flows through the Stage. If NORETAIN is active, each flow through the Stage will cause a new record to be read from the Secondary Input Stream to supply the setting.
.	Supply this parameter as blanks on the assumption that DFHXCURM will fix it up to point to the required Mirror Transaction (this is a dot)

This parameter should usually be specified as `*` unless there is a good reason for changing the Mirror Transaction.

If this parameter is omitted (which means that `userid` is also omitted), then `tran` and `userid` are passed as spaces, so implying that DFHXCURM will fixup the settings.

5.5: Specifying the Userid

The Userid to specify the Identity under which the Mirror Transaction is to run the program under in CICS is specified as the forth parameter on the stage.

```
streams: RXDPLP applid prog tran userid
                / NOABEND|ABEND RETAIN|NORETAIN NOTRACE|TRACE /
```

The Userid can be specified via:

*	Use the value returned from the Rexx userid() function. This will normally be the Userid specified on the Job Card (this is a star)
8 byte hard coded name	The Userid is supplied on the Stage and applies to all flows through the Stage
(Cn)	The Userid is to be taken from a Secondary Input Stream record starting in column n for 8 bytes. The start of a Secondary Input Stream record is at column 1. If the RETAIN option is active (the default) then the setting will be taken from the first Secondary Input Stream record for all flows through the Stage. If NORETAIN is active, each flow through the Stage will cause a new record to be read from the Secondary Input Stream to supply the setting.
(Wn)	The Userid is to be taken from a Secondary Input Stream record starting in column n for 8 bytes. The start of a Secondary Input Stream record is word 1. If the RETAIN option is active (the default) then the setting will be taken from the first Secondary Input Stream record for all flows through the Stage. If NORETAIN is active, each flow through the Stage will cause a new record to be read from the Secondary Input Stream to supply the setting.
.	Supply this parameter as blanks on the assumption that DFHXCURM will fix it up to supply the required Identity (this is a dot)

This parameter should usually be specified as `*` unless there is a good reason for forcing a defined Userid.

If this parameter is omitted, `userid` is passed as spaces, so implying that DFHXCURM will fixup the settings.

5.6: Specifying the Optional parameters

The optional parameters control various processing choices for the RXDPLP Stage. They are specified between / delimiters:

```
streams: RXDPLP applid prog tran userid
                / NOABEND|ABEND RETAIN|NORETAIN NOTRACE|TRACE /
```

5.6.1: Abend Processing

The NOABEND|ABEND optional parameter defaults to NOABEND. It controls whether or not the returned Commarea is to be sent to the Primary Output Stream if the DPL program Abended in CICS. After detecting an Abend, the Stage will stop processing

When the DPL program Abends, the ABEND setting will flow the Commarea to the Primary Output Stream, but it may be an exact copy of the Input Stream record expanded to 32500 bytes if the Program did not change it before Abending. The advantage of this setting is that the number of Primary Output Stream records will be the same as the number of Primary Input Stream records processed up to this point; but the disadvantage is that the content of the last Primary Output Stream record may not be what is expected or required.

The default setting of NOABEND says that if the DPL program abended the physical Commarea used for DPL communication will not be passed to the Primary Output Stream. In this case, as no Commarea is written when the Abend occurred, the Primary Output Stream will always contain valid Commareas. However, if the abend occurs this has the disadvantage that the last chunk of (potentially valid) data will be absent from the pipeline.

5.6.2: Secondary Input Stream Retention

The Secondary Input Stream is used to supply applid, prog, tran and userid settings dynamically via the (Cn) and (Wn) settings in the Pipe Stage.

The default setting is RETAIN. This means that a single Secondary Input Stream record is read to obtain the relevant settings. Once read, the settings apply for all flows through the Pipe Stage. Once read, the Secondary Input Stream record is discarded.

The NORETAIN option says that for each Primary Input Stream record (representing the Commarea to be sent to CICS) there will be an associated Secondary Input Stream record which contains the settings. Consequently, this option allows each flow to CICS to have different settings. This option is useful if different programs are to service the DPL flow (although all of the four settings can be controlled in this fashion, only prog is generally useful).

5.6.3: Tracing

The underlying RXDPL mechanism obtained from SupportPac CA1I permits the tracing of internal logic to assist in diagnosis. This is controlled by the `NOTRACE|TRACE` option. Tracing should only be used in an emergency to discover why something is failing.

NOTRACE is the default and does not trace activity.

`TRACE` turns on all traceing for RXDPL. Output is sent to the current destination for Rexx `say` verbs, which will usually be `//SYSTSPRT` .

5.7: The Input and Output Streams

5.7.1: Primary Input Stream

This contains the Commarea to be sent to CICS. There are no formatting constraints, but the maximum record size is 32500.

5.7.2: Secondary Input Stream

If parameters are specified dynamically, the Secondary Input Stream contains the settings. There are no formatting restrictions placed on this stream by RXDPLP as long as the information is accessible.

5.7.3: Primary Output Stream

This contains the Commareas sent by CICS as a result of EXCI processing. Primary Output Stream records are always 32500 bytes long, so they may have to be put though the `CHOP` Stage to get the correct record length.

5.7.4: Secondary Output Stream

The Secondary Output Stream contains the Return Code string from RXDPL for the EXCI Operation. This is fully documented in SupportPac CA1I. If the Secondary Output Stream is defined, there will be one record per Primary Input Stream record.

There will be the same number of records in the Secondary Output Stream as in the Primary Output Stream unless the DPL program has abended. In this case with the NOABEND option active, there will be an extra record in the Secondary Output Stream over the Primary Output Stream (as the returned Commarea is not written to the Primary Output Stream in this circumstance).

The Secondary Output Stream has the following blank delimited layout and is best processed using the word-based facilities of the SPECS Stage

word 1	The Overall Return Code for the RXDPLP Stage: >0 from EXCI related processing 0 function, from EXCI viewpoint, worked OK -1 > -99 from RXDPL processing -100 > from RXDPLP processing
word 2	The EIBRESP code from the DPL processing (0 if no error occurred)
word 3	the EIBRESP2 code from DPL processing (0 if no error occurred)
word 4	function that detected an error
further words	The following words contain a description of an error. A word consisting of just ':' (a single colon) will proceed a message sent from CICS

The EXCI related errors are documented in *'The EXCI CALL Interface'* section of the *CICS EXT* book.

Errors upto -99 are documented in SupportPac CA1I (and summarised in Section 9.2 "Errors generated by RXDPL" on page 40), but should not occur as these should be avoided by code within the RXDPLP Stage.

Errors > -100 are generated by RXDPLP and are documented in Section 9.1 "Errors generated by the RXDPLP Stage" on page 37.

However, the mere presence of a zero Return Code does not always mean that the DPL flow has succeeded. See the documentation in CA1I as to how some circumstances raise a zero Return Code and a non-zero EIBRESP. In normal circumstances, a test on the first word of the Secondary Output Stream is good enough to detect whether or not the DPL flow succeeded. For a full test, you must consult the Tertiary Output Stream.

The Secondary Output Stream record for a successful flow is

```
0 0 0 RXDPL OK
```

5.7.5: Tertiary Output Stream

The Tertiary Output Stream contains the variables returned in the `OutputCommareaStem.` of RXDPL to show full diagnostics about the EXCI flow. Consult the Supportpac CA1I documentation for full details.

The layout of the Tertiary Output Stream consists of blank delimited words and is best processed using the word functions of the `SPECS` stage. Each set starts with the name of the field in word n and the value in word n+1:

word 1	DIDFLOW	Shows whether or not an EXCI flow was actually sent to CICS and whether it succeeded or not.
word 2		Y A Flow made it to CICS and this was successfully processed F A Flow made it to CICS but this resulted in an Error (or an Abend) N A Flow did not make it to CICS either due to an error or CICS not being contactable (bad APPLID)
word 3	AC	The EXCI Reason Code
word 4		This is taken from Field 2 of the dpl_retarea and shows the Exec RESP2 for the DPL program execution. This will be set to '....' (four dots) if the value is not available
word 5	RESP	The EXEC Return Code
word 6		This is taken from Field 1 of the dpl_retarea and shows the value of EIBRESP for the DPL operation of the program. This will be set to '....' (four dots) if the value is not available
word 7	RESP2	The EXEC Reason Code
word 8		This is taken from Field 2 of the dpl_retarea and shows the value of EIBRESP2 for the DPL operation of the program. This will be set to '....' (four dots) if the value is not available
word 9	ABEND	The EXEC Abend Code (blanks if no abend occurred)
word 10		This is taken from Field 3 of the dpl_retarea and shows the abend code that the DPL operation of the program possibly engendered. If the program did not abend, it will be set to '....' (four dots)
word 11	MSG	Any message that CICS returned as a result of the DPL flow.
words 12 and upwards		This is taken from the five word Return Area for the dpl_request call. It will be set '.' (a dot) if CICS did not send any message. If CICS sent a message, it will start in the usual fashion with the word 12 word being the Message Number and words 13/14 being the date and time: the actual message starts at word 15.

If the Tertiary Output Stream is defined, the Secondary Output Stream will also be present. There will be the same number of records in the Tertiary Output Stream as in the Secondary Output Stream. Consequently, there will usually be the same number of records in the Tertiary Output Stream as in the Primary Output Stream unless the DPL program has abended. In this case, and if `NOABEND` is active, there will be an extra record in the Tertiary Output Stream over the Primary Output Stream (as the returned Commarea is not written to the Primary Output Stream in this circumstance).

The `DIDFLOW` setting - the second word of the Tertiary Output Stream - is the easiest way of determining whether or not the corresponding Primary Input Stream record was correctly processed by DPL facilities.

The Tertiary Output Stream record for a successful flow is

```
DIDFLOW Y AC 0 RESP 0 RESP2 0 ABEND .... MSG .
```

5.8: Input Streams

The Primary Input Stream contains the Commarea to be sent to CICS. This is placed in a Commarea of 32500 bytes with the actual data length being the length of the record.

If the `NORETAIN` option is applicable and active, there must be a corresponding Secondary Input Stream record containing the variable parameters.

If none of the RXDPLP Stage parameters is dynamic, a Secondary Input Stream is not required.

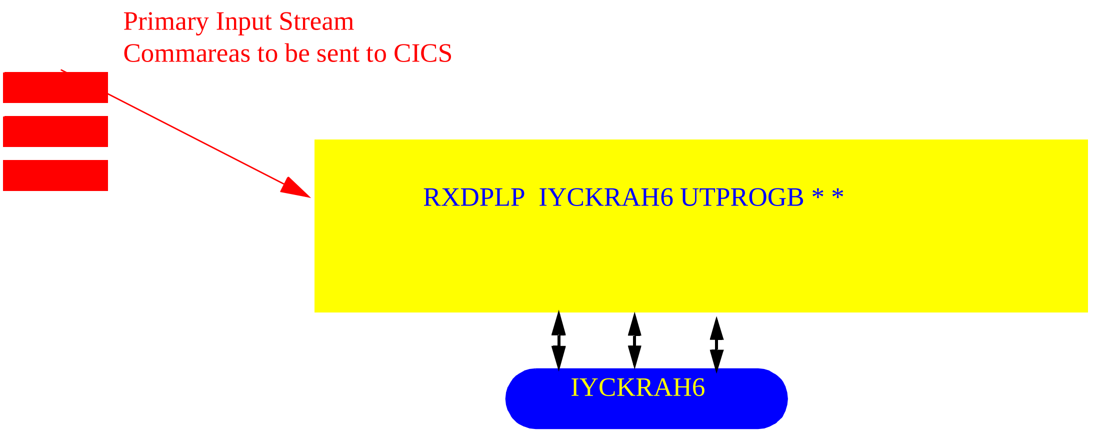


Figure 3: Input Stages without Dynamic settings

If at least one parameter is specified dynamically, a Secondary Input Stream is required. If the `RETAIN` option (default) is active, there is only one Secondary Input Stream record.

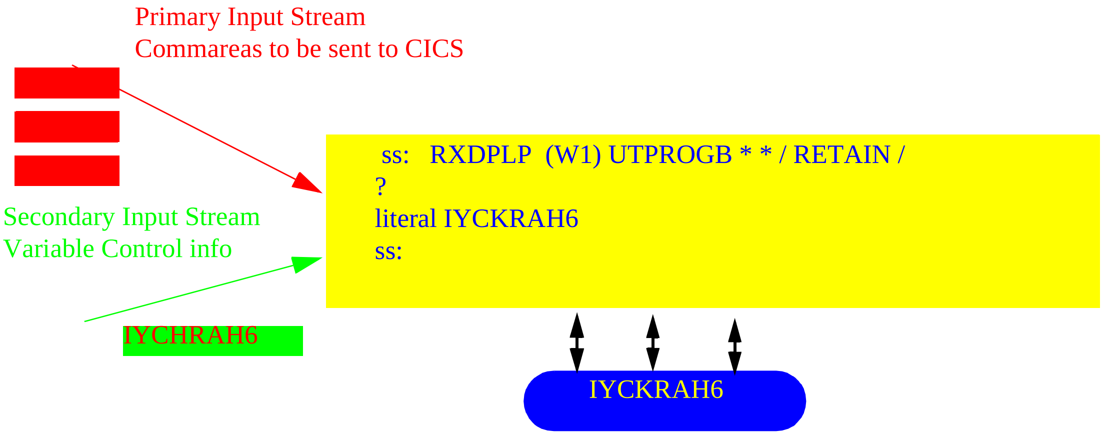
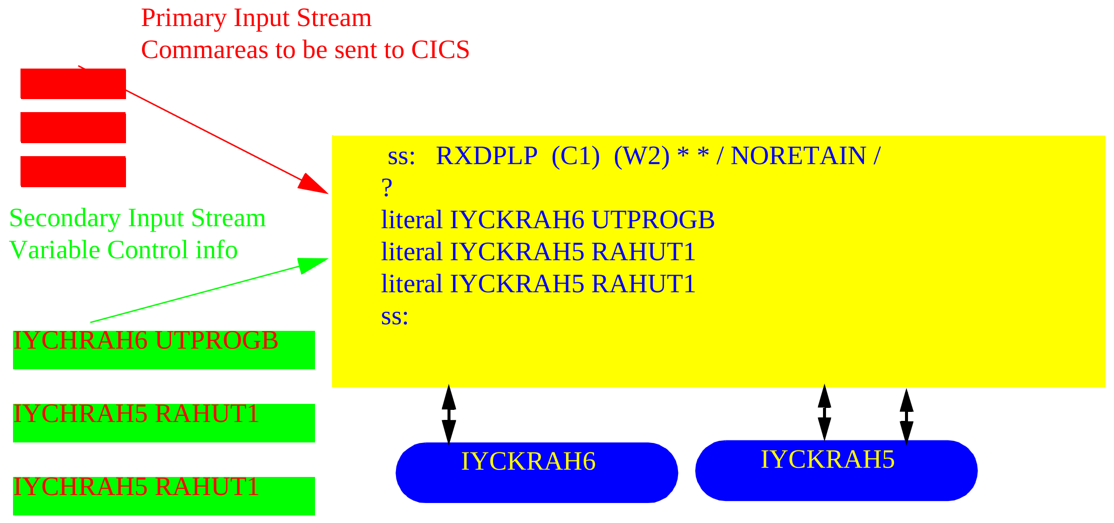


Figure 4: Input Stages with RETAIN

However, the whole flow can be made dynamic with the `NORETAIN` option active.



```
 ss:  RXDPLP  (C1)  (W2) * * / NORETAIN /
?
literal IYCKRAH6 UTPROGB
literal IYCKRAH5 RAHUT1
literal IYCKRAH5 RAHUT1
ss:
```

Figure 5: Input Stages with NORETAIN

5.9: Primary Output Stream

The Primary Output Stream consists of the Commarea sent back from CICS. The record length will always be 32500, so records may need to be put through the CHOP Stage to set the correct length for subsequent processing.

Each record in the Primary Input Stream will, if all is well, generate a record in the Primary Output Stream.

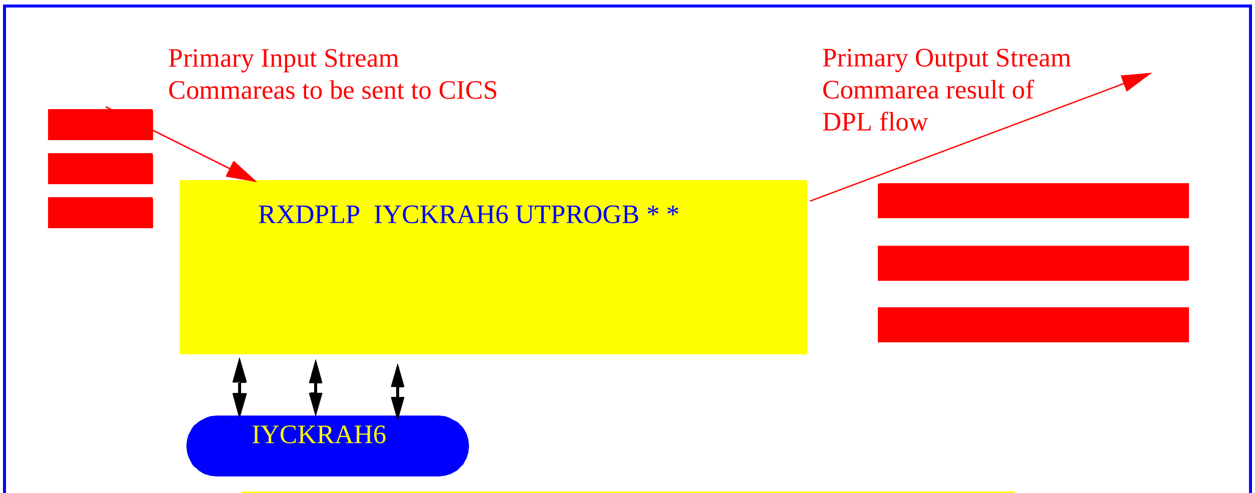


Figure 6: Normal Primary Output Stream records

If ABEND is active, then an abending DPL program will still result in a Commarea being generated to the Primary Output Stream, but it may be exactly the same as the Input Stream Record expanded to 32500 bytes. The Stage stops processing when an Abend is detected.

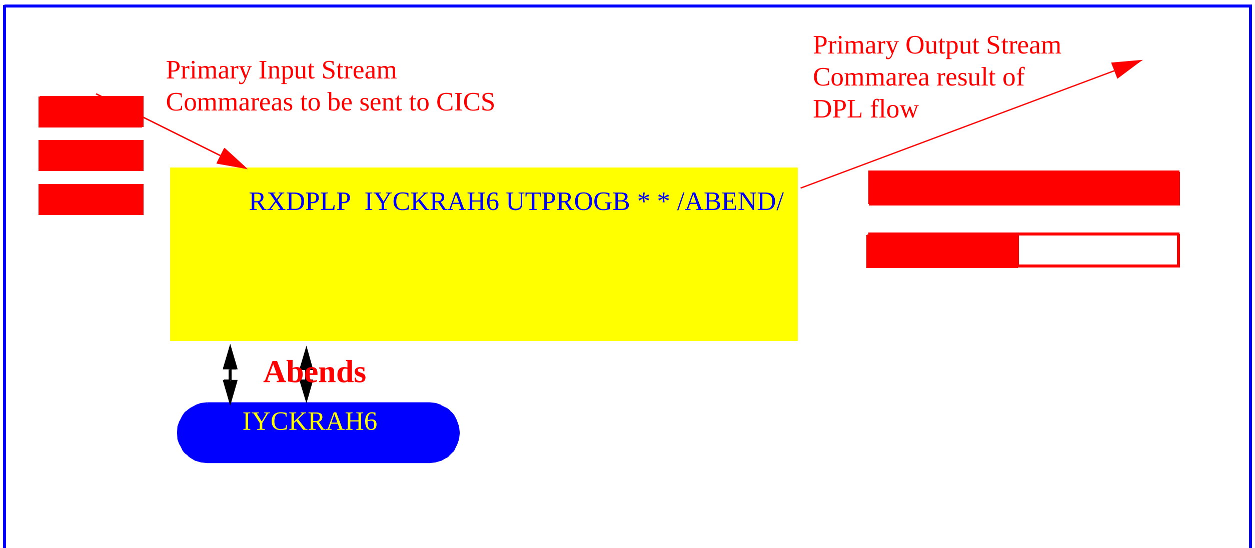


Figure 7: Primary Output Stream and ABEND

With NOABEND active, the failing Commarea is not written to the Primary Output Stream.

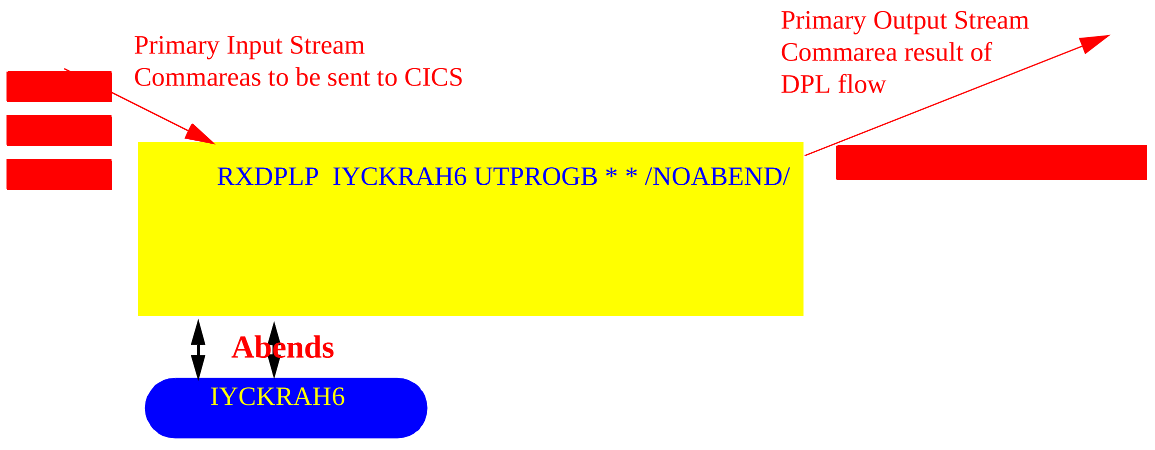
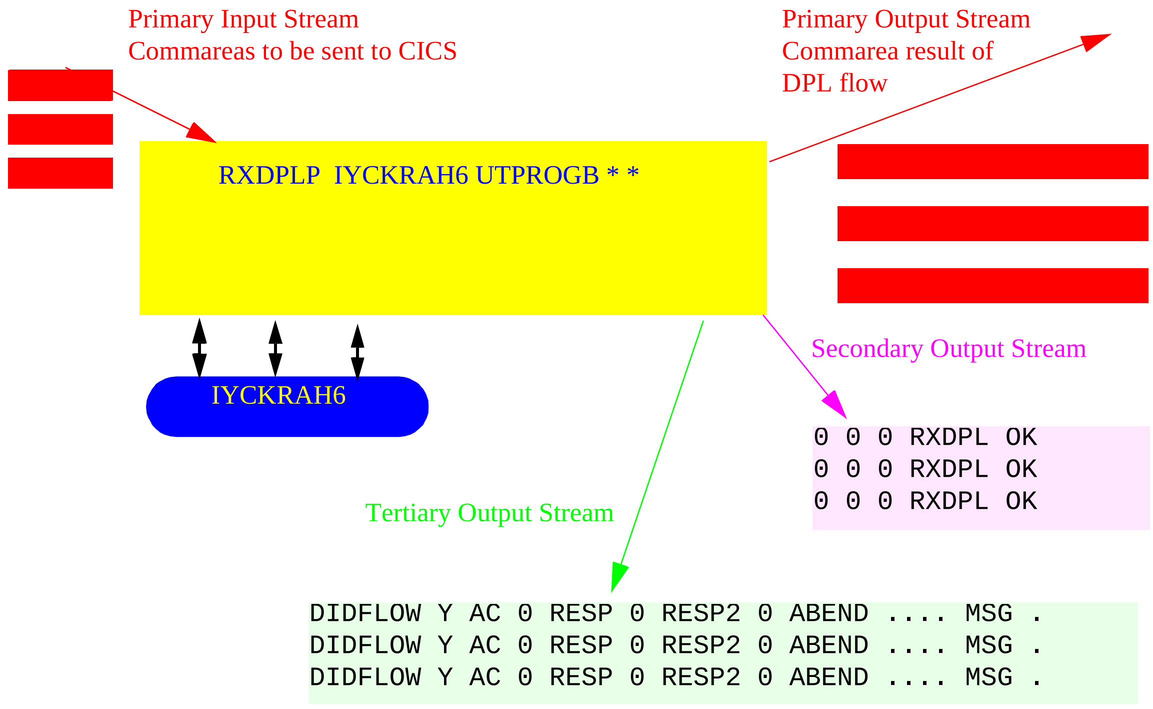


Figure 8: Primary Output Stream and NOABEND

5.10: Secondary and Tertiary Output Streams

The Secondary and Tertiary Potato Streams are optional and contain the Return Code and other associated error information from RXDPL. See the documentation in SupportPac CA1I for full information on these items.

If all is well, there will be the same number of records in the Secondary and Tertiary Output Streams as there are in the Primary Output Stream (and, accordingly, the Primary Input Stream).

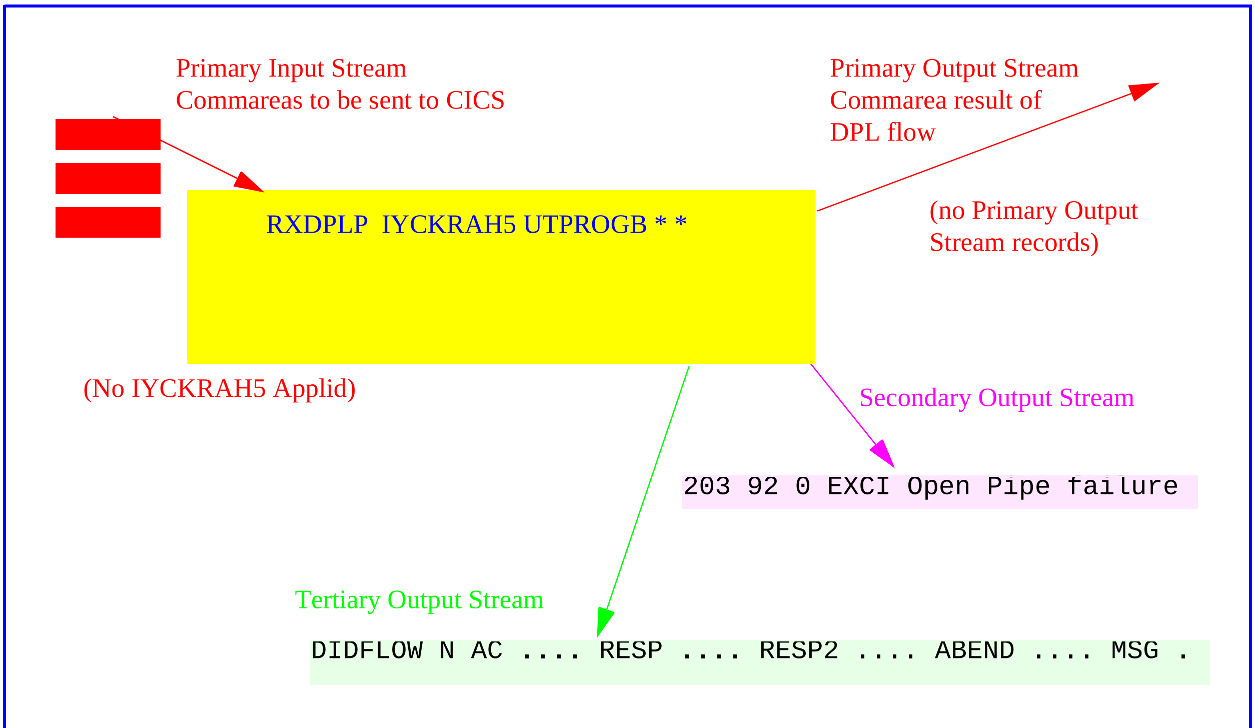


```
0 0 0 RXDPL OK
0 0 0 RXDPL OK
0 0 0 RXDPL OK
```

```
DIDFLOW Y AC 0 RESP 0 RESP2 0 ABEND .... MSG .
DIDFLOW Y AC 0 RESP 0 RESP2 0 ABEND .... MSG .
DIDFLOW Y AC 0 RESP 0 RESP2 0 ABEND .... MSG .
```

Figure 9: Successful Secondary & Tertiary Output Streams

A common error situation is where CICS is not contactable. When such an error occurs, no Primary Output Stream record is generated. The second word of the Tertiary Output Stream shows that the error did not happen within the CICS environment and the first word of the Secondary Output Stream record records the Connection failure.



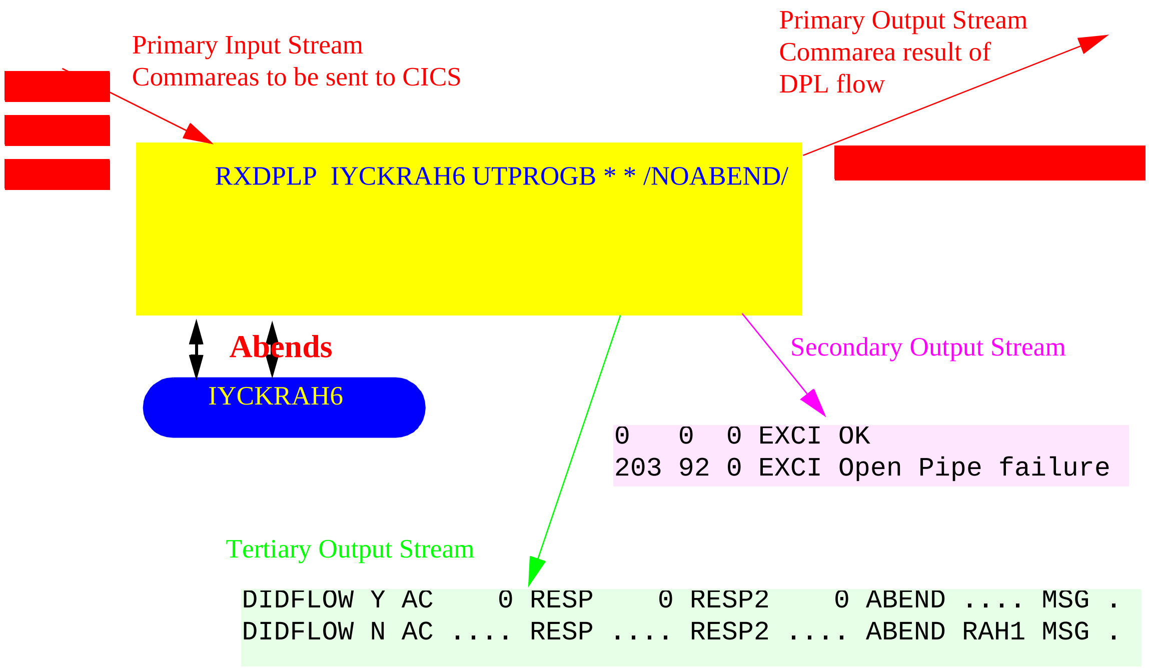
```
203 92 0 EXCI Open Pipe failure
```

```
DIDFLOW N AC .... RESP .... RESP2 .... ABEND .... MSG .
```

Figure 10: Output Streams on connection failure

The Output Streams have different numbers of records when an Abend occurs under the control of the ABEND or NOABEND setting.

In the case of NOABEND being active (which is the default) the Commarea sent back from CICS is not sent to the Primary Output Stream before the Stage stops processing.

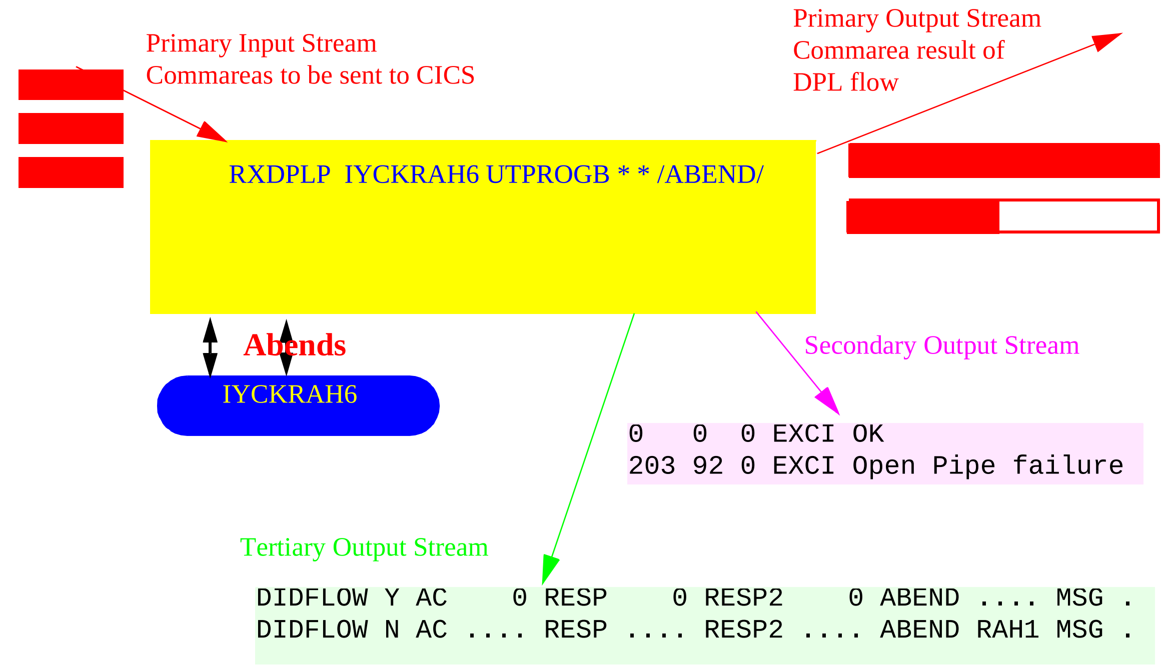


```
0   0  0 EXCI OK
203 92 0 EXCI Open Pipe failure
```

```
DIDFLOW Y AC    0 RESP    0 RESP2    0 ABEND .... MSG .
DIDFLOW N AC .... RESP .... RESP2 .... ABEND RAH1 MSG .
```

Figure 11: Output Streams on an Abend with NOABEND active

When ABEND is active the Commarea is returned, but it may contain the unchanged contents of the Primary Input Record.



```
0   0  0 EXCI OK
203 92 0 EXCI Open Pipe failure
```

```
DIDFLOW Y AC    0 RESP    0 RESP2    0 ABEND .... MSG .
DIDFLOW N AC .... RESP .... RESP2 .... ABEND RAH1 MSG .
```

Figure 12: Output Streams on an Abend with ABEND active

Chapter 6: Errors from RXDPLP

6.1: Return Code Strings from RXDPL

The underlying RXDPL function sends a Return Code String depending upon how far function got before the error or unexpected circumstance arose. These get passed from RXDPLP in the Secondary Output Stream.

The wholly successful function will return

```
0 0 0 RXDPLLINK OK
```

If an error arose before the EXCI function is attempted, the format will be

```
Error_Code 0 0 RXDPLLINK description_of_error
```

If the error arose due to EXCI processing, the format will be dependant upon where the error arose. However, it will start

```
Overall_Return_Code EIBRESP_code EIBRESP2_code EXCI
```

The fifth and subsequent words will be a description of the error. If any one of these subsequent words (5 and upwards) is a colon, then after the colon will be a message sent from CICS. This message will be in the usual format of MessageId, Date, Time and the actual text of the message.

```
... EXCI <more words> : MessageId Date Time the_message_itself
```

Chapter 7: Common Runtime Errors

This section shows records in the Secondary and Tertiary Output Streams for common runtime errors. Observe that CICS traps some of these and treats them as EXCI-errors whilst others are deemed to be Application errors and so treated differently.

7.1: Unknown Mirror Transaction

This circumstance is treated as a correct EXCI flow which results in a system error.

Secondary Output Stream	`0414 0000 0000 EXCI Flow DPL failure : DFHAC2001 31/08/2001 11:31:53 IYCKRAH6 Transaction 'CSMX' is not recognized. Check that the transaction name is correct.`
Tertiary Output Stream	`DIDFLOW F AC 0414 RESP 0000 RESP2 0000 ABEND .... MSG DFHAC2001 31/08/2001 11:31:53 IYCKRAH6 Transaction 'CSMX' is not recognized. Check that the transaction name is correct.`

7.2: Unknown Program

This circumstance occurs in the Mirror transaction so the failure is treated as an application error.

Secondary Output Stream	`0 0 0 RXDPLLINK OK`
Tertiary Output Stream	`DIDFLOW F AC 0000 RESP 0027 RESP2 0000 ABEND .... MSG .`

7.3: CICS not contactable

This is detected on the EXCI `Open_Pipe` call.

Secondary Output Stream	`203   92   0   EXCI Open Pipe failure`
Tertiary Output Stream	`DIDFLOW N AC .... RESP .... RESP2 .... ABEND .... MSG .`

7.4: Program Abended

This specific failure is trapped by EXCI and so results in an EXCI type of error.

Secondary Output Stream	0422 0000 0000 EXCI Flow DPL failure :
Tertiary Output Stream	DIDFLOW F AC 0422 RESP 0000 RESP2 0000 ABEND <abend_code> MSG .

7.5: Bad Userid

This circumstance is treated as a correct EXCI flow which results in a system error.

Secondary Output Stream	0414 0000 0000 EXCI Flow DPL failure : DFHAC2047 31/08/2001 12:21:47 IYCKRAH6 While performing an attach for node DFHGEN a security violation was detected.
Tertiary Output Stream	DIDFLOW F AC 0414 RESP 0000 RESP2 0000 ABEND . MSG DFHAC2047 31/08/2001 12:21:47 IYCKRAH6 While performing an attach for node DFHGEN a security violation was detected.

Chapter 8: Running the RXDPLP Stage

8.1: Within JCL

When running a Batch Pipes Step, the Pipe specification is usually too long to fit within the `// PARM='......'` statement. One usually uses a `RUNPIPE` Stage to permit the specification:

```
//RUN4     EXEC PGM=PIPE,
//         PARM='(SEP ; end ? ) < dd=PIPEIN ;  join * ;
//             runpipe ; > dd=PIPEPOUT coerce'
//REXX     DD  DSN=<library containing RXDPLP>,DISP=SHR
//STEPLIB  DD  DSN=<library containing RXDPL>,DISP=SHR,DCB=BLKSIZE=32760
//         DD  DSN=<library containing DFHXCOPT>,DISP=SHR
//         DD  DSN=CICS.SDFHEXCI,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//PIPEPOUT DD  SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//PIPEOUT  DD  SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSTSPRX DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//PIPEIN   DD  *
(SEP ; end ?)
  literal cemt i task       ;
  literal RAH Rules         ;
 ss: RXDPLP (W2) (C1) * *  / notrace retain abend/;
            specs /1->/ 1 1-* next ; > dd=PIPEOUT coerce
  ?
  literal utprogb iyckrah6 word3 word4 word5 ; ss:   ;
            specs /2->/ 1 1-* next ; > dd=PIPEOUT coerce
  ?
   ss: ;    specs /3->/ 1 1-* next ; > dd=PIPEOUT coerce

/*
```

A negative Return Code for the Job Step is converted in the usual fashion for JCL. These are shown in the second (green) column of Section 9.1 "Errors generated by the RXDPLP Stage" on page 37.

8.2: Within a MVS Exec

RXDPLP can run as a normal Pipe Stage within a Pipe command in a Rexx/MVS Exec. In this case, the Return Code from the Pipe command is available in the reserved Rexx RC variable. Values in RC will be in the -nnn format as shown in the first (red) column of Section 9.1 "Errors generated by the RXDPLP Stage" on page 37.

```
//LIB      EXEC PGM=IEBGENER
//*
//*            Create the exec library
//*
//SYSUT2   DD  DSN=&&TSO(T0),DISP=(NEW,PASS),
//             UNIT=SYSDA,
//             SPACE=(CYL,(1,1,10)),
//             DCB=(DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSPRINT DD  DUMMY
//SYSIN    DD  DUMMY
//SYSUT1   DD  DATA,DLM='##'
/* T0       REXX EXEC   */
/* ==================   */
/*                      */

 'pipe (SEP ; end ?) ',
  'literal OK cemt i task ; ',
  'literal OK RAH Rules   ; ',
  'ss: RXDPLP (W2) (C1) * *  / notrace retain abend/; ',
  'specs /1*</ 1 1-* next ; > dd=PIPEOUT coerce ',
  '?',
  'literal utprogb iyckrah6 word3 word4 word5 ; ss: ; ',
  'specs /2*</ 1 1-* next ; > dd=PIPEOUT coerce ',
  '?',
  'ss: ; specs /3*</ 1 1-* next ; > dd=PIPEOUT coerce '

 piperc = rc
 exit

##
/*
//RUN      EXEC PGM=IKJEFT01
//REXX     DD  DSN=<library containing RXDPLP>,DISP=SHR
//STEPLIB  DD  DSN=<library containing RXDPL>,DISP=SHR,DCB=BLKSIZE=32760
//         DD  DSN=<library containing DFHXCOPT>,DISP=SHR
//         DD  DSN=CICS.SDFHEXCI,DISP=SHR
//SYSPROC  DD  DSN=&&TSO,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//PIPEOUT  DD  SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSTSIN  DD  *
T0
/*
```

When running within a Exec, you are better off using the native RXDPL interface provided within SupportPac CA1I rather than operating via a Pipe.

Chapter 9: Return Codes

9.1: Errors generated by the RXDPLP Stage

The first column (red) shows the contents of a Secondary Output Stream record for an error. The second column (green) shows the Return Code as returned on a Pipe JCL Step.

`-100 0 0 RXDPLP APPLID (p1) must be supplied`	3996	This parameter cannot be specified as '*'
`-101 0 0 RXDPLP APPLID (p1) Invalid () specification`	3995	This parameter can only be specified dynamically via (Cn) or (Wn)
`-102 0 0 RXDPLP APPLID (p1) too long`	3994	The maximum length of this parameter is 8 bytes
`-110 0 0 RXDPLP PROGRAM (p2) must be supplied`	3986	This parameter cannot be specified as '*'
`-111 0 0 RXDPLP PROGRAM (p2) Invalid () specification`	3985	This parameter can only be specified dynamically via (Cn) or (Wn)
`-112 0 0 RXDPLP PROGRAM (p2) too long`	3984	The maximum length of this parameter is 8 bytes
`-121 0 0 RXDPLP TRAN (p3) Invalid () specification`	3975	This parameter can only be specified dynamically via (Cn) or (Wn)
`-122 0 0 RXDPLP TRAN (p3) too long`	3974	The maximum length of this parameter is 4 bytes
`-131 0 0 RXDPLP USERID (p4) Invalid () specification`	3965	This parameter can only be specified dynamically via (Cn) or (Wn)
`-132 0 0 RXDPLP USERID (p4) too long`	3964	The maximum length of this parameter is 8 bytes
`-140 0 0 RXDPLP Could not determine the number of Output Streams rc <rc>`	3956	The OUTPUT subcommand failed to return the number of Streams defined for the Stage
`-141 0 0 RXDPLP Secondary Input Stream required but not defined rc <rc>`	3955	Dynamic parameter settings were requested, but a Secondary Input Stream was not defined to convey this information

`-142 0 0 RXDPLP Secondary Input Stream required but no record currently available rc <rc>`	3954	A Secondary Input Stream is supplying dynamic parameters, but when a Primary Input Stream record was read, there is no corresponding Secondary Input Stream record available to supply the dynamic settings. If `RETAIN` is active, this condition will appear for the first Primary Input Stream record. If `NORETAIN` is active, this will error will occur if there are more Primary Input Stream records than Secondary Input Stream records.
`-143 0 0 RXDPLP Secondary Input Stream was empty rc <rc>`	3953	A Secondary Input Stream record was found to be Zero bytes long (a null-record) which is not allowed
`-144 0 0 RXDPLP Could not obtain current Stage Number rc <rc>`	3952	An error was returned on the `STAGENUM` subcommand
`-145 0 0 RXDPLP Can not run at 1st Stage`	3951	RXDPLP cannot run as the first Stage of a Pipeline
`-150 0 0 RXDPLP Secondary Input Stream was read in error rc <rc>`	3946	The `INPUT` subcommand failed to read a record from the Secondary Input Stream to obtain dynamic parameter settings
`-151 0 0 RXDPLP Secondary Input Stream contained no data rc <rc>`	3945	A Secondary Input Stream record was found to be Zero bytes long (a null-record) which is not allowed
`-152 0 0 RXDPLP Secondary Input Stream was too short for Applid`	3944	The (Cn) specification is bigger than the Secondary Input Stream record length
`-153 0 0 RXDPLP Secondary Input Stream has too few words for Applid`	3943	The (Wn) specification is for a word which is not present in the Secondary Input Stream record
`-154 0 0 RXDPLP Secondary Input Stream was too short for Program`	3942	The (Cn) specification is bigger than the Secondary Input Stream record length
`-155 0 0 RXDPLP Secondary Input Stream has too few words for Program`	3941	The (Wn) specification is for a word which is not present in the Secondary Input Stream record
`-156 0 0 RXDPLP Secondary Input Stream was too short for Tran`	3940	The (Cn) specification is bigger than the Secondary Input Stream record length
`-157 0 0 RXDPLP Secondary Input Stream has too few words for Tran`	3939	The (Wn) specification is for a word which is not present in the Secondary Input Stream record
`-158 0 0 RXDPLP Secondary Input Stream was too short for Userid`	3938	The (Cn) specification is bigger than the Secondary Input Stream record length

`-159 0 0 RXDPLP Secondary Input Stream has too few words for Userid`	3937	The (Wn) specification is for a word which is not present in the Secondary Input Stream record
`-170 0 0 RXDPLP Primary Input Stream was read in error rc <rc>`	3926	The `INPUT` subcommand failed to read a record from the Primary Input Stream to flow to CICS in a Commarea
`-171 0 0 RXDPLP Primary Input Stream contained no data rc <rc>`	3925	A Primary Input Stream record was found to be Zero bytes long (a null-record) which is not allowed for a Commarea to flow to CICS
`-172 0 0 RXDPLP Primary Input Stream record is longer than 32500 len: <length>`	3924	A Primary Input Stream record has to fit into a physical Commarea for conveyance to CICS and the maximum size of this physical commarea is 32500 bytes
`-180 0 0 RXDPLP Could not write to the Primary Output Stream rc: <rc>`	3916	The `OUTPUT` subcommand used to write the returned Commarea as a record in the Primary Output Stream failed
`-181 0 0 RXDPLP Could not write to the Secondary Output Stream rc: <rc>`	3915	The `OUTPUT` subcommand used to write the RXDPL Return Code String as a Secondary Output Stream record failed
`-182 0 0 RXDPLP Could not write to the Tertiary Output Stream rc: <rc>`	3914	The `OUTPUT` subcommand used to write the RXDPL Return Variables as a Tertiary Output Stream record failed

Return Codes originate from REXX Pipeline subcommands and are documented in '*9.5 Pipeline Subcommands*' in the *Pipe UG*.

9.2: Errors generated by RXDPL

-1 0 0 RXDPLLINK	Bad number of Parms	4 parms must be specified for the EXCI linkage call
-2 0 0 RXDPLLINK	Control Stem Variable not supplied	The 2nd parameter must be supplied
-3 0 0 RXDPLLINK	Input Commarea variable not supplied	The 3rd parameter must be specified
-4 0 0 RXDPLLINK	Output Commarea Variable not supplied	The 4th parameter must be supplied
-5 0 0 RXDPLLINK	PROG component not supplied	The Control Stem. variable must have a PROG component supplied which must not start with X'00' or X'40’
-6 0 0 RXDPLLINK	USERID component not supplied	The Control Stem. variable must have an USERID component supplied which must not start with X'00' or X'40’
-7 0 0 RXDPLLINK	Input Commarea not supplied	The length of the Input Commarea (in `.0`) must not be 0
-8 0 0 RXDPLLINK	Input Commarea data not supplied	A `.0` component must be specified for the Input Commarea Stem
-9 0 0 RXDPLLINK	Input Commarea too big	The maximum Commarea size is 32500
-10 0 0 RXDPLLINK	Commarea zero length	An attempt is being made to use a zero length physical commarea
-11 0 0 RXDPLLINK	Commarea area Getmain failure	Internal GETMAIN error in obtaining storage for the physical Commarea to be used for the EXCI flow
-98 0 0 RXDPL	Unknown Request	The first parameter is the function required and must be `'INIT'`, `'TERM'` or `'LINK'`
-99 0 0 RXDPL	Incorrect Number of parms supplied	At least some parameters must be supplied to DFHDPL

See SupportPac CA1I for a full listing of these errors. These errors should not appear as the RXDPLP Stage should have trapped them.

9.3: Errors generated by EXCI

8	ERROR
16	INVREQ
22	LENGERR
27	PGMIDERR
53	SYSIDERR
70	NOTAUTH
81	TERMERR
82	ROLLEDBACK
88	LINKERR
201	NO_CICS_IRC_STARTED
202	NO_PIPE
203	NO_CICS
204	WRONG_MVS_FOR_RRMS
205	RRMS_NOT_AVAILABLE
401	INVALID_CALL_TYPE
402	INVALID_VERSION_NO
403	INVALID_APPL_NAME
404	INVALID_USER_TOKEN
405	PIPE_NOT_CLOSED
406	PIPE_NOT_OPEN
407	INVALID_USERID
408	INVALID_UOWID
409	INVALID_TRANSID
410	DFHMEBM_LOAD_FAILED
411	DFHMET4E_LOAD_FAILED
412	DFHXCURM_LOAD_FAILED
413	DFHXCTRA_LOAD_FAILED
414	IRP_ABORT_RECEIVED
415	INVALID_CONNECTION_DEFN
416	INVALID_CICS_RELEASE
417	PIPE_MUST_CLOSE
418	INVALID_PIPE_TOKEN
419	CICS_AFCB_PRESENT
420	DFHXCOPT_LOAD_FAILED
421	RUNNING_UNDER_AN_IRB
422	SERVER_ABENDED
423	SURROGATE_CHECK_FAILED
424	RRMS_NOT_SUPPORTED
425	UOWID_NOT_ALLOWED
426	INVALID_TRANSID2
427	INVALID_CCSID
428	INVALID_ENDIAN
601	WS_GETMAIN_ERROR
602	XCGLOBAL_GETMAIN_ERROR
603	XCUSER_GETMAIN_ERROR
604	XCPIPE_GETMAIN_ERROR
605	VERIFY_BLOCK_GM_ERROR
606	SSI_VERIFY_FAILED
607	CICS_SVC_CALL_FAILURE
608	IRC_LOGON_FAILURE
609	IRC_CONNECT_FAILURE
610	IRC_DISCONNECT_FAILURE
611	IRC_LOGOFF_FAILURE
612	TRANSFORM_1_ERROR
613	TRANSFORM_4_ERROR
614	IRP_NULL_DATA_RECEIVED
615	IRP_NEGATIVE_RESPONSE
616	IRP_SWITCH_PULL_FAILURE
617	IRP_IOAREA_GM_FAILURE
619	IRP_BAD_IOAREA
620	IRP_PROTOCOL_ERROR
621	PIPE_RECOVERY_FAILURE
622	ESTAE_SETUP_FAILURE
623	ESTAE_INVOKED
624	SERVER_TIMEDOUT
625	STIMER_SETUP_FAILURE
626	STIMER_CANCEL_FAILURE
627	INCORRECT_SVC_LEVEL
628	IRP_LEVEL_CHECK_FAILURE
629	SERVER_PROTOCOL_ERROR
630	RRMS_ERROR
631	RRMS_SEVERE_ERROR
632	XCGUR_GETMAIN_ERROR
903	ESTAE_SETUP_ERROR
904	ESTAE_INVOKED

>>>>>>>>>>>>>>>END of document<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<