Syncsort ZEN® EE Security Installation and Configuration

syncsort

ZEN EE Security - Installation and Configuration

Syncsort is a provider of enterprise software and the global leader in Big Iron to Big Data solutions. As organizations worldwide invest in analytical platforms to power new insights, Syncsort's innovative and high-performance software harnesses valuable data assets while dramatically reducing the cost of mainframe and legacy systems. Thousands of customers in more than 85 countries, including 87 of the Fortune 100, have trusted Syncsort to move and transform mission-critical data and workloads for nearly 50 years. Now these enterprises look to Syncsort to unleash the power of their most valuable data for advanced analytics. Whether on premise or in the cloud, Syncsort's solutions allow customers to chart a path from Big Iron to Big Data. Experience Syncsort at www.syncsort.com.

Version 3.11
Last Update: December 4, 2017

Contact Details

North America

Syncsort Mainframe Product Services
Syncsort Incorporated
2 Blue Hill Plaza, #1563
Pearl River, NY 10965
U.S.A

Tel: (877) 700-0970
Fax: (201) 882-8305

Email: info@syncsort.com
Web: http://www.syncsort.com

Europe, Middle East, and Africa

Syncsort EMEA Support Center
Stroombaan 4
1181 VX Amstelveen
The Netherlands

Tel: +31 20 219 9912

Email: supportemea@syncsort.com
Web: http://www.syncsort.com

Other Regions
Customers in other regions should contact their local Syncsort representative.

Contents

List of Panels

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter introduces the facilities available when you run ZEN EE SECURITY - APIAS on your system. The final section in this chapter contains a number of 'Frequently Asked Questions' with detailed responses.

ZEN EE SECURITY is referred to by its simple name 'APIAS' throughout this document.

Overview

APIAS comprises three primary functions:

- **SSL transport for Enterprise Extender (SSL EE)**

  Enables you to SSL secure all traffic passing across an Enterprise Extender link between systems. It works by converting the EE UDP packets to TCP packets and then SSL encrypting them prior to transmission. On receipt, APIAS decrypts the packets, and reconverts them to UDP packets which are then passed to the standard EE ports.

- **TCP transport for Enterprise Extender (TCP EE)**

  This function is provided for companies that do not permit UDP traffic on their networks which would therefore exclude the use of EE. In this mode, APIAS converts the EE UDP packets to TCP packets and it is these that flow across the EE link. Although SSL encryption is not done in this mode (the assumption being that either it is not required, or that it is being done at the VTAM level), APIAS is able to use digital certificates for authentication.

- **Multiple stack support for Enterprise Extender**

  This feature enables EE to work with multiple stacks in the same LPAR, a capability not normally allowed. As with the TCP transport feature, APIAS can optionally use digital certificates for authentication although this would require APIAS to be running at both ends of the EE link and therefore currently limits this authentication to z/OS only. If authentication is not required, the target EE system can be non-z/OS, for example an IBM 2216, PCOM, CISCO routers, CS for LINUX and so on. For z/OS systems, this facility can be combined with either the TCP or SSL features.

APIAS is simple and quick to install and configuration is through a standard PARMLIB member that can be edited using the standard ISPF editor.

SSL Transport for Enterprise Extender

Enterprise Extender enables SNA (HPR) traffic to be transmitted over an IP network. SNA data is carried in UDP packets on registered UDP ports 12000 to 12004. Port 12000 is used for XID and NLP exchanges, and ports 12001 to 12004 are mapped to the APPN transmission priorities.

Since EE relies on encapsulating SNA data in UDP packets which are not normally encrypted or authenticated, EE is usually not an acceptable method of transmitting sensitive corporate data between sites, especially where part of the network is known to be inherently insecure. With the imminent demise of IBM's 3745 communications controller, many companies are being forced to consider what alternative secure network options they have.

APIAS provides the means of using IBM's EE unchanged in a secure way, irrespective of the network routes used, including the Internet, between the source and target IP addresses.

It works by converting the EE UDP packets to TCP packets and then SSL encrypting them prior to transmission. On receipt, APIAS decrypts the packets, and reconverts them to UDP packets which are then passed to the standard EE ports. Prior to any transmission taking place, full digital certificate authentication can take place between the source and target systems so you are assured that the link is fully secure.

Neither the source nor the target system is aware of any of this conversion process happening. The target system simply sees the UDP packets on the standard ports 12000 to 12004 as if they had been received directly from the source system. Thus, APIAS provides full authentication and encryption security in a situation in which it would not normally be available, and without the disruption caused by major system changes.

TCP Transport for Enterprise Extender

For reasons of security, many Installations do not permit UDP traffic to be carried on their IP networks which can present a migration problem given the discontinuation of the 3745 SNA controller. However, APIAS can convert the EE UDP packets to TCP packets and transmit them across the network. At the remote end of the connection, APIAS converts the packets back to UDP and passes them to the EE ports. As far as the system is concerned, no change has taken place since it is still receiving UDP packets on the standard EE ports, however, the UDP protocol restriction has been honored.

For additional security APIAS can optionally authenticate the connection between the source and target systems using a digital certificate. Note, however, that in this mode of operation, APIAS is not encrypting the TCP packets that are sent across the network (although it can of course do this if required as described in the previous section).

Multiple Stack Support for Enterprise Extender

In a situation where an Installation, often for reasons of security, is running multiple stacks in a LPAR, EE is only allowed to be used on one of them. This restriction can cause difficulties if each stack is required to route traffic to different remote sites and EE is required for enabling SNA traffic to be carried by the IP network. Generally, this would require radical system changes to split the stacks across separate LPARs with all of the system redefinition and disruption this would cause.

APIAS can resolve this difficulty by enabling EE traffic to be routed through any of the stacks on the LPAR *without any major changes to the current LPAR definitions*. By running APIAS on each stack, it can pick up the EE packets and route them as required to the ‘other’ IP stacks or systems.

At the simplest level, APIAS simply performs UDP packet relaying between the IP stacks but you could combine this capability with the TCP Transport function which would enable you to authenticate the connection between the source and target systems using a digital certificate. If this is not a requirement, a ‘target’ EE system could be non-z/OS such as PCOM, or a variety of CISCO routers.

FAQ

***Q**. Does APIAS support the X509 Certificate standard and if so is PKI supported?*

The answer to this is “yes”. APIAS encryption and authentication is provided by an API to System SSL (part of z/OS). They key question is ‘Does System SSL support X509 and PKI and the answer again, is “yes”.

The following text can be found in: “Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security” - SG24 6840:

“In z/OS System SSL provides a common set of libraries and an API. System SSL is part of the System SSL Cryptographic Services Base element of z/OS. z/OS Communications Server uses the System SSL APIs to create and manage SSL connections. X.509 certificates are used by both the client and server when securing communications using System SSL.

System SSL supports the following two methods for managing PKI private keys and digital certificates:

- A z/OS shell-based program called gskkyman. gskkyman creates, fills in, and manages a z/OS HFS file that contains PKI private keys, certificate requests, and certificates. This z/OS HFS file is called a key database and, by convention, has a file extension of .kdb.
- The z/OS SecureWay Security Server (RACF) RACDCERT command. RACDCERT installs and maintains PKI private keys and certificates in RACF. RACF supports multiple PKI private keys and certificates to be managed as a group. These groups are called keyrings. RACF keyrings are the preferred method for managing PKI private keys and certificates for System SSL.”

APIAS implementation of authentication supports both gskkyman and RACF keyrings.

***Q**. Can APIAS be used to secure applications other than Enterprise Extender.*

William Data Systems has chosen to focus on Enterprise Extender security because IBM already offers alternative facilities for encrypting and authenticating TCP/IP applications:

"With z/OS 1.7 IBM introduced a new Application Transparent Transport Layer Security (AT-TLS) function in the TCP/IP stack to provide TLS for TCP/IP sockets applications that require secure connections. AT-TLS performs TLS on behalf of the application by invoking the z/OS System Secure Socket Layer (SSL) in the TCP transport layer of the stack. System SSL provides support for TLSv1, SSLv3, and SSLv2 protocols."

Further information about AT-TLS can be found in IBM Redbook SG24-7172: "Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 4 Policy-Based Network Security."

***Q**. Is it necessary to have a copy of APIAS at each end of the connection?*

The answer to this is "yes". APIAS removes the SNA data from the UDP packets and places it in TCP packets. Once it is TCP-encapsulated APIAS applies SSL authentication and encryption to the TCP packets (SSL does not support UDP). A copy of APIAS is required at the other end of the Enterprise Extender connection to decrypt the packets and place the data back into UDP packets.

***Q**. Does APIAS support SYSPLEX Distributor or Dynamic VIPA?*

APIAS works with Dynamic VIPAs and there is nothing in the APIAS code to prevent it working in a Sysplex Distributor environment.

Sysplex Distributor enables applications to be distributed across different LPARs with some element of dynamic re-routing. APIAS is not an application, it is a network service. Multiple APIAS instances can be active within a Sysplex Distributor environment but since APIAS only supports point-to-point secure connections (as does Enterprise Extender) to partner EE systems, it is not clear how customers would want to use Sysplex Distributor and APIAS/EE.

***Q**. Does APIAS support the SRCIP function?*

Source IP addressing enables an application to always have the same source IP address when transmitting packets to the outside world. However, this only applies to applications which BIND to a null IP address. Both Enterprise Extender and APIAS bind to a specific IP address, so outbound packets will (and must) contain this address.

***Q**. Does APIAS have a facility to establish connections (tunnels) on demand, as some firewall systems do?*

The major difference between APIAS and an IPSec-based solution is that APIAS is operating at the application layer and IPSec is a function of the connection layer. However, APIAS to APIAS connections through SSL could be viewed as a "tunnel". They are predefined in the APIAS configuration and can be dynamically activated and de-activated by command.

Some level of automation could be achieved using standard Automatic Operations software. APIAS will *not* dynamically allocate “tunnels” based on traffic flow as the tunnel needs to be open before traffic can flow and dynamically allocating tunnels based on traffic rather than configuration would be a security exposure.

CHAPTER 2

Installing APIAS

This chapter describes how APIAS is installed. It covers the prerequisites, getting the software on to your host and the automated installation job build process.

Once product installation is complete, you should see the chapter ‘Configuring APIAS’ for information on configuring the product specifically for your Installation.

System Prerequisites

APIAS requires z/OS to be at version 1.4 or higher.

Note also that for z/OS versions prior to 1.7 a key IBM PTF **must** be applied for APIAS to work in a single IP stack. The relevant APAR is PK06135 and the PTF numbers are:

z/OS 1.4 UK04942
z/OS 1.5 UK04943
z/OS 1.6 UK04944
z/OS 1.7 UK05347

APIAS may still be run without these PTFs but it will require a second IP stack to be used.

PC Prerequisites

By preference you require a networked PC capable of using FTP to transfer files to the target system where APIAS is to be installed.

If you cannot use FTP, the install can be done manually using IND£FILE. In this latter case you should invoke the setup.exe after unzipping the product to your PC and click the Help button. In the Help file you will find a section called 'Manually Installing a WDS Product' which contains a full description of the process.

No more than 5 Mb of temporary hard disk space is required for the install process. This space can be freed after the data is transferred to the target system.

Upgrading from Version 3.1 to a later 3.1

If you are installing a later version 3.1 on your system, no special migration is necessary. You should let the automated installation procedure build the job control, review it and only allow the creation and unloading of the new libraries to take place, along with the authorization of the new APIAS library if required. You can then start the product with your current configuration and APIAS procedure (suitably modified to pick up the newly unloaded libraries).

Installing APIAS

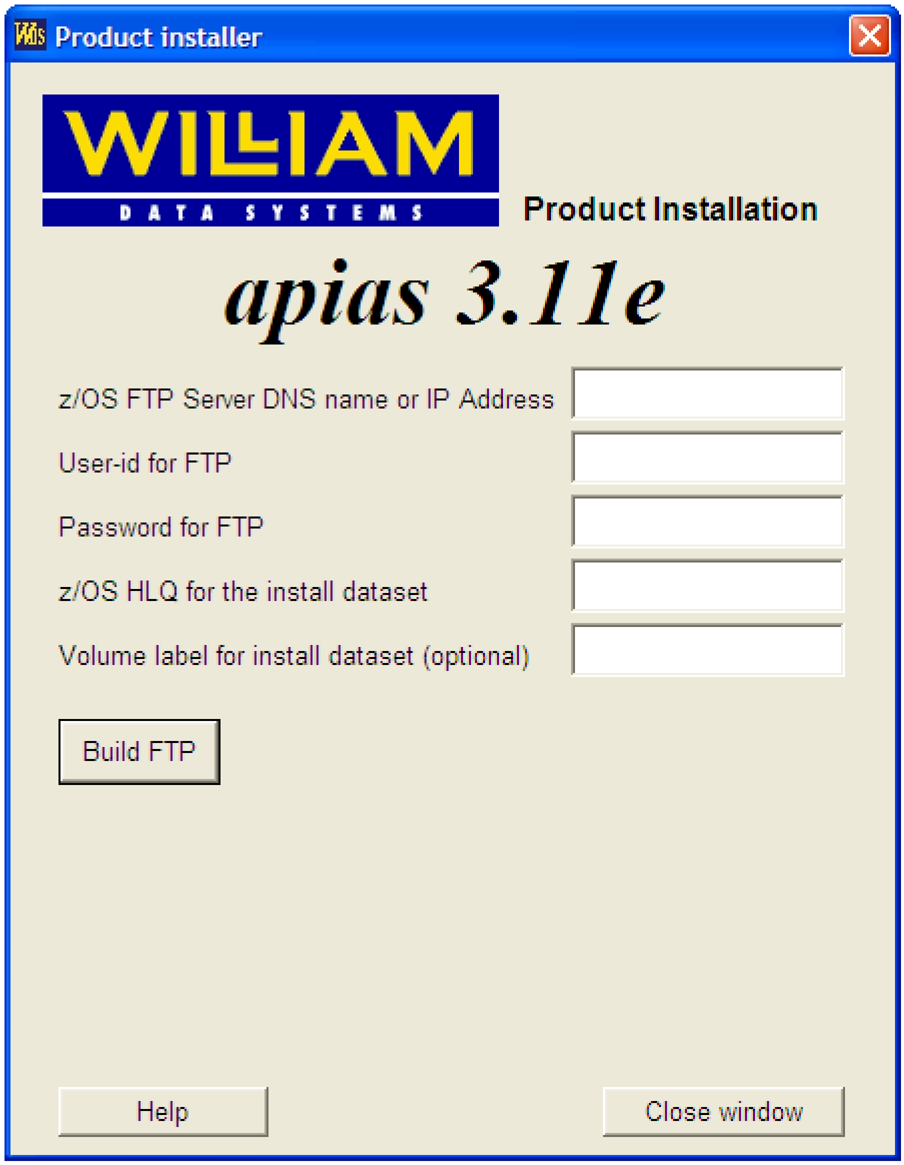
This section guides you through the installation of APIAS. On completion, you will have the product installed on your system and all necessary system updates made such as library authorization and copying the APIAS procedure to your nominated procedure library will have been completed.

APIAS is usually installed from a self-extracting EXE downloaded from the WDS website, provided via an email attachment, or shipped on a CD. WDS has provided Automatic Install Procedures for both the PC and the mainframe to make the process of installing the product as quick and easy as possible.

If your site is unable to download EXE files from our website, WDS will provide a ZIP which you should unzip to a c:\Apias (or some other chosen directory) and then invoke the setup.exe. After this follow the steps outlined below.

Run the PC Self-Extractor and Automatic Installer

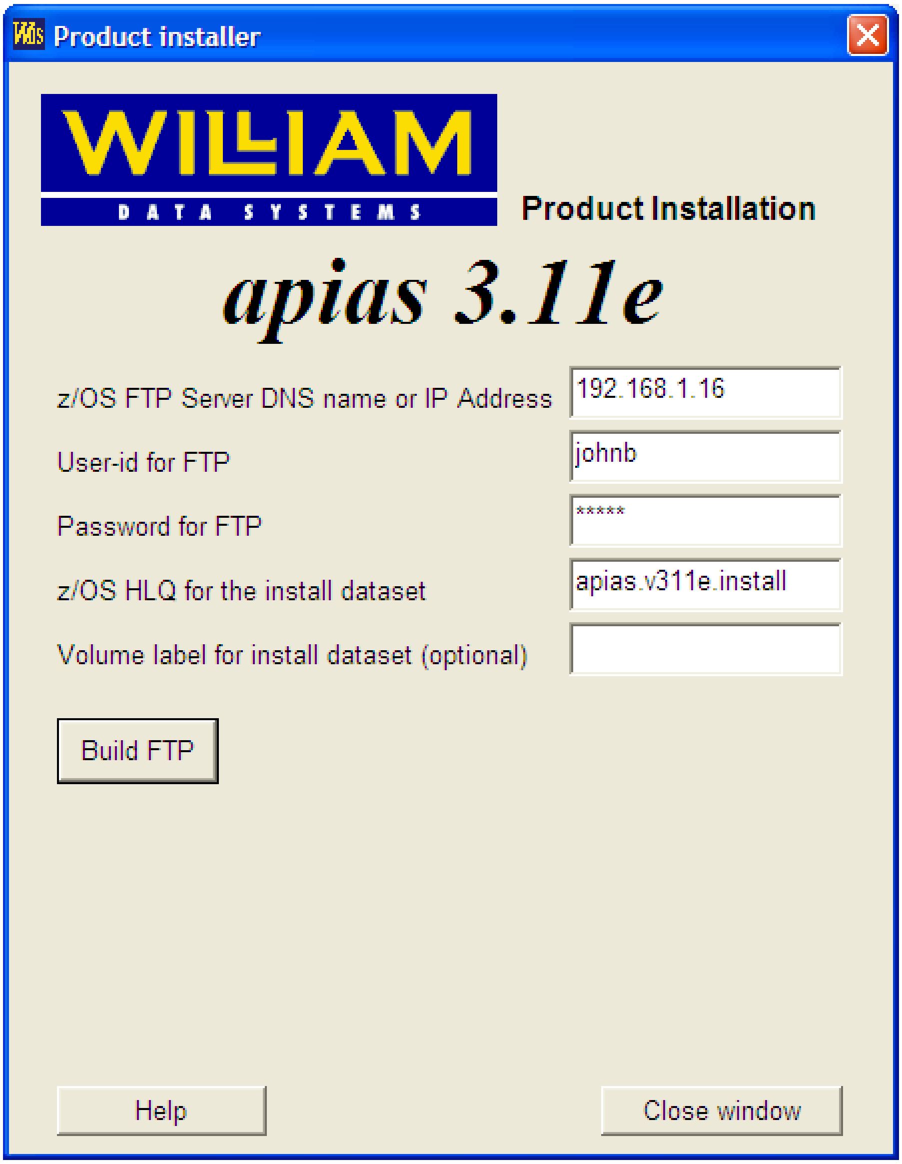
Ensure that the self-extracting EXE (APSVnnnc.EXE) is loaded into a PC folder of your choice. Run the self-extractor by double-clicking on it, or by right-mouse clicking and selecting the ‘Open’ option. When the extract has completed the Automatic Installer (setup.exe) will be run and you will see a window similar to this:



Panel 1. Automatic Installer Window

The Automatic Installer requires you to key the IP address or name of the target z/OS system, a userid and password for FTP, and a high-level qualifier for the MVS datasets that it builds and (if required) the target volume name. It then generates and runs an FTP to load the product to the chosen z/OS system and a job to TSO RECEIVE the product into an install TAPEIMGE library. Printed output from the RECEIVE step is transferred back to your PC for verification of success.

Here is an example with all the fields completed:



Panel 2. Automatic Installer Window with Fields Completed

Now click the 'Build FTP' button and two more buttons, 'View FTP Control Files' and 'Run FTP' will appear. It is recommended that you do view the FTP control files to verify that the Installer has built job control that will run successfully on your system. You can make any changes appropriate and *save the two control files* before clicking the 'Run FTP' button. After this the process of uploading the product to your mainframe is automatic.

After the upload and TSO RECEIVE are complete, the printed output is transferred back to your PC (again using FTP) and you can verify that the procedure has worked correctly.

Note that if you have any problems with the Automatic Installer you should always consult the Help first. The Help file contains a full description of the install processes - both manual and automatic - and a Troubleshooting Guide and FAQ. Often you will be able to resolve your own problems. However, if you do have unresolvable difficulties, you can always contact you local WDS office for technical support - a full list of offices is given at the beginning of this manual and on the WDS website at www.willdata.com.

Run the Mainframe Automatic Installer

All the job control necessary to install, authorize and run the product on your mainframe is generated when you run the SETUP program that now resides in the tape image library you created and loaded in the previous step.

However, it is left to you to decide on the extent to which you allow the process to happen automatically. For example, for some of the steps, such as RACF authorization, it may be necessary to rely on other groups in the site and therefore these steps may not be able to be done immediately.

From TSO option 6 enter:

EX 'hlq.TAPEIMGE(SETUP)'

The panel illustrated below will be displayed:

```
------------------- Apias Version 3.1 Installation - Step 1 ------------------
Command ===> ________________________________________________________________

HLQ for runtime datasets     : APIAS.V311_______________
HLQ for install datasets     : APIAS.V311.INSTALL_______
HLQ for TCP/IP  datasets     : TCPIP____________________
HLQ for L/E     datasets     : CEE______________________

Volume for runtime datasets  : _______     specify volser or SMS
Volume for install datasets  : _______     specify volser or SMS
Volume for VSAM datasets     : _______     specify volser or SMS
Unit for datasets            : SYSDA___
Unit for work datasets       : SYSALLDA

Apias procname               : APIAS___

Select (/) the steps for which you require automated.

/ Copy procedures into       : SYS1.PROCLIB_____________________________
/ APF Authorize Apias Load library
/ Create RACF profiles

Press Enter to continue.
Press PF3 to terminate
```

Panel 3. Installation Option Panel

Key into this panel the high level qualifier and volume for the runtime and install datasets and the TCP/IP and language environment datasets; key the volumes for the SMP/E VSAM datasets (overkeying the default values if required). The 'Install' datasets are required only during the installation and SMP/E maintenance of the product. The 'Runtime' datasets are required during normal running and are the only datasets required on other z/OS images.

Key the procedure name you want to use and mark the additional steps you want to build job control for using the '/' character. Note that if you choose not to generate these steps automatically they will need to be done manually prior to starting the product for the first time.

Here is an example of the primary input panel with all the fields completed:

```
------------------ Apias Version 3.1 Installation - Step 1 ------------------
Command ===> ______________________________________________________________

HLQ for runtime datasets     : APIAS.V31E1.RUNTIME______
HLQ for install datasets     : APIAS.V311E.INSTALL______
HLQ for TCP/IP  datasets     : TCPIP____________________
HLQ for L/E     datasets     : CEE______________________

Volume for runtime datasets  : ZOSWDF_    specify volser or SMS
Volume for install datasets  : ZOSWDF_    specify volser or SMS
Volume for VSAM datasets     : ZOSWDF_    specify volser or SMS
Unit for datasets            : 3390 ___
Unit for work datasets       : 3390

Apias procname               : APIAS___

Select (/) the steps for which you require automated.

/ Copy procedures into       : WDS.PROCLIB_____________________________
/ APF Authorize Apias Load library
/ Create RACF profiles

Press Enter to continue.
Press PF3 to terminate
```

Panel 4. Installation Option Panel with all fields completed

When you have made all required changes, press the Enter key. The installation procedure then builds all the necessary job control to complete the installation of the product.

During this build process, a window is displayed on the panel to show the progress like this:

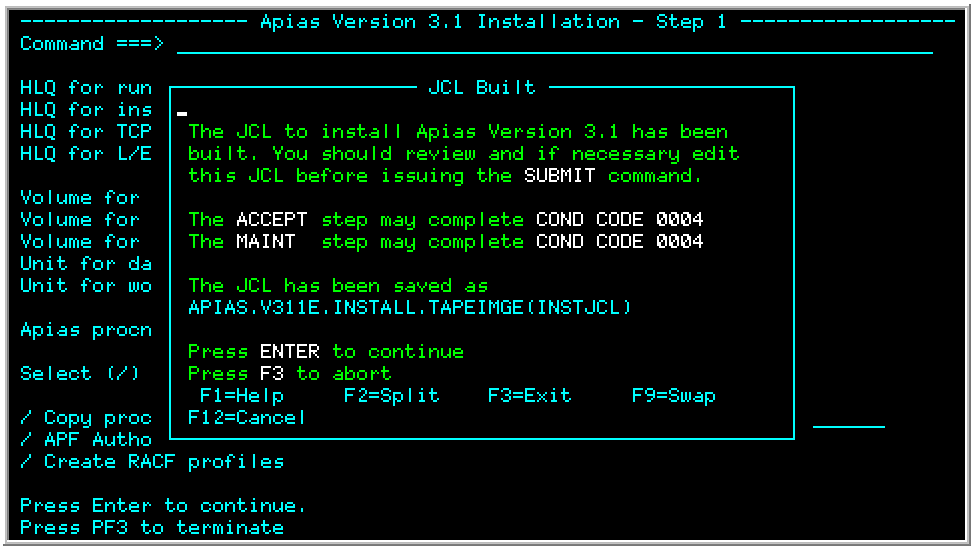
```
------------------ Apias Version 3.1 Installation - Step 1 ------------------
Command ===> ______________________________________________________________

HLQ for runtime datasets     : APIAS.V31E1.RUNTIME_ ---- Building JCL ----
HLQ for install datasets     : APIAS.V311E.INSTALL_ |_                    |
HLQ for TCP/IP  datasets     : TCPIP_______________ | Step 1: completed   |
HLQ for L/E     datasets     : CEE_________________ | Step 2: completed   |
                                                    | Step 3: completed   |
Volume for runtime datasets  : ZOSWDF_    specify v | Step 4: processing  |
Volume for install datasets  : ZOSWDF_    specify v | Step 5:             |
Volume for VSAM datasets     : ZOSWDF_    specify v | Step 6:             |
Unit for datasets            : 3390____             | Step 7:             |
Unit for work datasets       : 3390____             |                     |
                                                    |                     |
Apias procname               : APIAS___             |                     |
                                                    |                     |
Select (/) the steps for which you require automate |  F1=Help   F2=Split |
                                                    |  F3=Exit   F9=Swap  |
/ Copy procedures into       : WDS.PROCLIB_________  ---------------------
/ APF Authorize Apias Load library
/ Create RACF profiles

Press Enter to continue.
Press PF3 to terminate
```

Panel 5. Installation Job Control Build Progress Panel

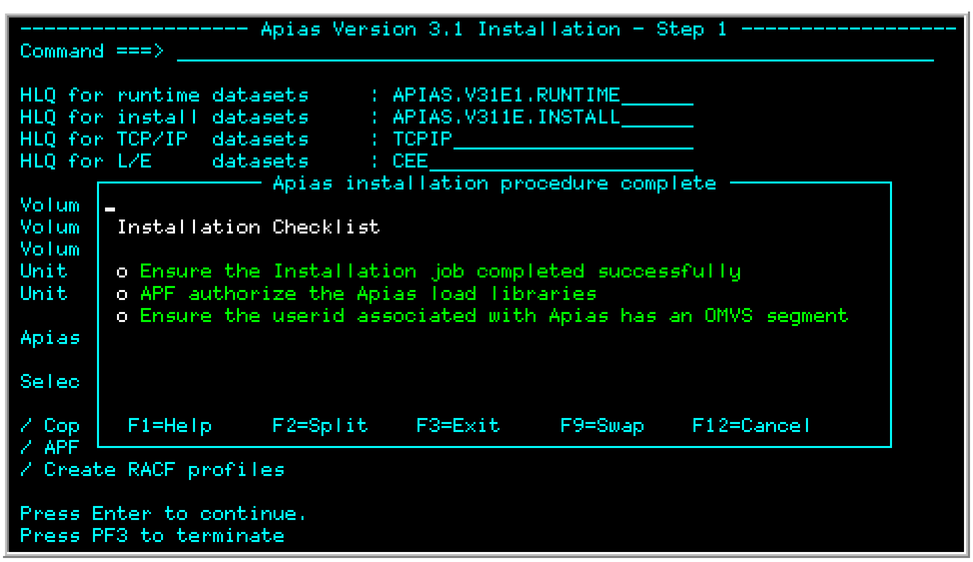
Once the build process is complete a window is displayed that indicates the member in which the job control has been saved:



Panel 6. Installation Job Control Build Completion

When you press Enter on this panel, the generated job control will be displayed. You can then make any additional changes or deletions required and file it away for later submission, or submit it immediately. The generated job control contains all the steps necessary to take the file you originally uploaded and build from it all the files APIAS needs to complete the installation process and to run in a production environment. See the member 'INSTJCL' in the TAPEIMGE dataset, as indicated in the completion window, to see the generated job control.

Once you have either submitted the job, or filed it away for submission later, the following panel is shown:



Panel 7. Installation Verification Panel

This panel provides a checklist of the activities that need to be completed prior to activating the product. Provided all of these steps have been done properly, the product should initiate without any problems.

You can now configure the product as described in the appropriate configuration chapter after pressing Enter on the final confirmation panel shown above. For further details and explanation of the steps involved in the generated job control, see the section 'APIAS Post-Install System Set-up' on page 16.

APIAS Start-up Procedure

For completeness, this section contains the procedure that is generated by the automated install procedure and which initiates the APIAS started task. You may alter this if required according to your own Installation's standards and naming conventions.

APIAS Startup Procedure

This job is written by the installation process into the *hlq*.TAPEIMGE library as member APIAS.

```
//APIAS    PROC PROFILE=PROFILE
//********************************************************
//*                                                      *
//* Apias Version 3.1                                    *
//*                                                      *
//********************************************************
//APIAS    EXEC PGM=APIAS,REGION=4M,TIME=1440,PARM='&PROFILE'
//STEPLIB  DD   DISP=SHR,DSN=APIAS.V311E.RUNTIME.LOADLIB
//         DD   DISP=SHR,DSN=GSK.SGSKLOAD
//PARMLIB  DD   DSN=APIAS.V31E1.RUNTIME.PARMLIB,DISP=SHR
//SYSOUT   DD   SYSOUT=*
//SYSAUDIT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSTRACE DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
```

Notes on the Start-up Procedure

'GSK.SGSKLOAD' is only required in the STEPLIB if you are planning to use RACF for your certification and it is not in your system's LINKLIST, but must be APF authorized. This library can be removed altogether if you are not going to use APIAS's SSL support, or if you store your digital certificates elsewhere.

For z/OS version 1.6 and above, the SSL modules have moved from GSK.SGSKLOAD to a PDS called '*your-hlq*.SIEALNKE' (for example, 'SYS1.SIEALNKE'). The start-up procedure should therefore be changed appropriately if required.

APIAS Post-Install System Set-up

This section describes the steps that are required once the base product install has been completed. *All these steps can be performed by the job control that is automatically generated by the install process, or they can be performed manually.*

Note that if you have run the job control generated by the automated install procedure, you will have already completed all the tasks described in this section. You can now proceed directly to the chapter 'Configuring APIAS' on page 39. Otherwise you may need to complete one or more of the jobs listed below.

Step 1 *Define RACF (or equivalent Security System) Resources*

APIAS uses RACF, or your system's equivalent security system, to perform authorization checking for the APIAS started task. There is a step in the job control deck generated by the automated install procedure that contains the necessary RACF statements. You should review this job prior to submitting it, or issue the commands *or their equivalents in your security system* prior to attempting to activate APIAS.

Failure to define the correct RACF authorities may mean the product will not function correctly.

To use APIAS, a profile must be created in the RACF STARTED class with the name 'APIAS.*'. Running the generated job step will dynamically add the required resource and will activate it using the SETROPTS ... REFRESH parameter. Below is an example of the RACF job step from the generated job control deck:

```
//*****************************************************
//* Define RACF resources                            *
//*****************************************************
//RACF     EXEC PGM=IKJEFT01,DYNAMNBR=75,REGION=6M
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTERM  DD DUMMY
//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSIN  DD *
  RDEFINE STARTED APIAS.* STDATA(USER(JOHNB))
  SETROPTS CLASSACT(STARTED)
  SETROPTS RACLIST(STARTED) REFRESH
/*
```

Note also that the userid under which APIAS runs must have an OMVS Segment defined. The UID parameter for the user must provide enough authority to use sockets, including opening a TCP and UDP socket for listening. Failure to ensure that this is the case will render APIAS unusable.

Users running z/OS 1.14 with ICSF configured and the CSFSERV RACF class defined should ensure use of the hardware RNG to avoid ICH408I messages by adding the following statement to the RACF job above:

```
  PERMIT CSFRNG ACCESS(READ) CLASS(CSFSERV)ID(your_userid)
```

Step 2 *Authorize the APIAS Load Library*

To activate successfully, the libraries that contains the product load modules *must* be APF authorized. There are two methods of doing this: adding the libraries to the APF list, (member IEAAPFnn/PROGxx in SYS1.PARMLIB or its equivalent), or issuing the SETPROG command. Note that the SETPROGs will not hold across an IPL so will need to be reissued in this event. The generated job control authorizes the libraries using a SETPROG and the library name will be as you specified during the automated install procedure:

```
//******************************************************
//* APF Authorize LOADLIB                              *
//******************************************************
//APFAUTH EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//SYSIN  DD DUMMY
// SETPROG APF,ADD,DSN=APIASEE.V311A.RUNTIME.LOADLIB,VOLUME=your_volid
```

Step 3 *Create the APIAS Started Task Procedure*

The APIAS startup member is created in the TAPEIMGE dataset during the install process. It contains the statements necessary to initiate APIAS. Make any alterations you require and then copy it to your chosen procedure library. An example of the relevant step from the generated job control deck is shown below:

```
//******************************************************
//* Copy PROCLIB Members                               *
//******************************************************
//PROCLIB  EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT3   DD   UNIT=SYSDA,SPACE=(CYL,(2,2))
//SYSUT4   DD   UNIT=SYSDA,SPACE=(CYL,(2,2))
//IN       DD DISP=SHR,DSN=APIASEE.V311A.INSTALL.TAPEIMGE
//OUT      DD DISP=SHR,DSN=WDS.PROCLIB
//SYSIN    DD *
 COPY INDD=((IN,R)),OUTDD=OUT
 SELECT MEMBER=((PROCLIB,APIAS,R))
```

Very Important Notes

Since APIAS is handling Enterprise Extender traffic, it should be given the same dispatching priority as TCP/IP. Failure to do this when APIAS is active could cause intermittent and unexpected EE link terminations due to timeouts.

Also, if your TCP/IP stack should be terminated, whether intentionally or inadvertently, you must restart APIAS. Failure to do this may lead to connection failures on one or more of the EE ports.

Job Control Parameter Options

For the APIAS started task a single parameter is available to define the name of the APIAS profile member:

TABLE 1. APIAS Procedure Parameter Options

Parameter	**Purpose**
PROFILE=*profile-name*	Defines the name of the profile member that contains the APIAS configuration statements. The default name provided in the sample procedure is ‘PROFILE’ which would normally be read from the APIAS PARMLIB built during the install procedure.

Troubleshooting

Startup Problems

Problems do sometimes occur when attempting to start APIAS and these are almost invariably due to RACF (or your alternative security system) authority problems. In particular ensure the following:

- That the job to build the RACF definition has been run, or the correct authorities have been defined manually.
- That the userid under which APIAS runs has an OMVS segment. Also, the UID parameter for the user must provide enough authority to use sockets, including opening a TCP and UDP socket for listening.
- That the APIAS load library has been authorized.

Connection Problems

These can be due to problems in the APIAS profile, so the first place to look is the SYSOUT dataset where any errors in the configuration statement will be logged. You should also check that the processing and post-processing stages have ended cleanly.

If there are no processing or post-processing errors, check the Audit Log for any connectivity problems such as handshake failures, 'bind failed' or 'link failed' messages. Also, look for messages in the APIAS SYSOUT dataset such as 'ssl_listen(udpbind): EDC8115I Address already in use.'

Sometimes, when restarting APIAS after only a very short period of inactivity, it is possible for one or more of the previous socket listeners to still be active giving rise to bind failures or connection inactive messages at initialization time. If this is the case you can stop APIAS and manually drop the residual listeners, or use the procedure given in "Appendix 2 APIAS PROC Clean-up Step" on page 103.

Connection problems can also occur if APIAS is activated when the EE links are already active. To be consistently successful when initializing APIAS, the resources should be activated in the following sequence:

1. Activate the EE XCA using a V NET, ACT,ID=*your-xca* command
2. Activate APIAS
3. Activate the EE IP link using a V NET,ACT,ID=*your-netee* command

It is also *always* worth checking the system log where some APIAS messages are written, as well as RACF security related messages.

Also, remember that if you restart your TCP/IP stack for any reason, you *must* restart APIAS as well. Failure to do this will almost certainly lead to connection failures on one or more of the EE ports.

Users on z/OS 1.12 and Above

IBM introduced a new feature at z/OS 1.12 which causes VTAM to send an LDLC probe to the remote partner during the activation of the EE connection to check whether all the EE ports are accessible. Since it is APIAS at the remote end of the connection rather than native EE and APIAS is not currently able to handle it, the LDLC probe fails with the IST1430I message. Here is an (partial) example:

```
V NET,ACT,ID=ETPN19A
IST097I VARY ACCEPTED
...
IST2330I EE HEALTH VERIFICATION FAILED FOR ETPVPU6 AT 14:29:08
...
IST1411I INOP GENERATED FOR L1060000
IST1430I REASON FOR INOP IS EE HEALTH VERIFICATION FAILURE
```

Until this difficulty can be overcome in APIAS' code, there is a simple work-around since the EE connection verification process can be deactivated using the command:

```
F VTAM,VTAMOPTS,EEVERIFY=NEVER
```

on ***all systems*** running z/OS 1.12 and above; if you only issue the command on one system, the connection will fail. (The command is not valid on pre z/OS 1.12 systems and will be rejected).

CHAPTER 3

Configuring for SSL EE

APIAS is able to SSL secure your Enterprise Extender traffic by converting it to encrypted TCP packets. This chapter gives an overview of how the process works, and how to set up the appropriate definitions in VTAM, TCP/IP and APIAS to take advantage of this product feature.

Overview

Enterprise Extender (EE) is an architecture that enables SNA (HPR) traffic to be transmitted over an IP backbone. SNA data is carried in UDP packets on registered UDP ports 12000 to 12004. Port 12000 is used for XID and NLP exchanges, and ports 12001 to 12004 are mapped to the APPN transmission priorities.

Since UDP is a connectionless protocol, it is not possible to use SSL/TLS encryption and digital certification to secure the data. This can lead to organizations being reluctant to use EE for their data, especially when the connections pass through the internet. Indeed, some organizations will not use UDP protocol at all, other than on an internal LAN. This position is becoming untenable with the expected demise of the 3745 SNA FEP and the organizational requirement to convert networks from SNA to IP.

APIAS can resolve this issue by SSL-enabling your EE links. This has a number of effects:

- It eliminates the transmission of EE UDP packets across the IP network
- EE packets can be encrypted and digitally certified
- It is more secure than usual since packets are encrypted even on the internal LAN

SecureEE Profile Statement

The parameters of the SecureEE APIAS profile statement define the secure Enterprise Extender support. The SecureEE section specifies the EE IP address of the system on which APIAS is running and an EEPath parameter for each path to other systems in the network that are communicating with this system using EE.

APIAS will use the Keyring and CertLabel parameter values specified in the Defaults section as well as a number of other defaults, however, these may be over-ridden by identical parameters on each EEPath if required. For the full description, see “SecureEE Section” on page 51.

System-Related Changes

The following system-related changes are required for APIAS’s secure EE feature to work:

- Alter the VTAM Resources

  The VTAM resources used to connect the source EE system to the target EE system need to be changed so that they cause the EE packets to be sent instead to APIAS. The PATH statement in the VTAMLST member specifies the virtual device address defined in the stack. It is recommended that a new VTAMLST member is created to hold the new definitions so that switching back to ‘standard’ EE can easily be done.

- Define a second TCP/IP stack for APIAS's use.

  This step is *only* required when the z/OS system does not permit APIAS to 'listen' on the EE port numbers on a different (VIPA) IP address on the same stack that runs EE - see "System Prerequisites" on page 8. In this situation, APIAS must run on a different IP stack from the stack running the EE it is supporting. The second stack needs to have connectivity with the primary IP stack, typically using XCF or CTCs.

Note that there is no restriction on running 'normal' EE connections as well as APIAS Secure EE connections. They would, however, need to run across different links.

An Illustrated Example

This section contains simplified examples of the way a two systems might be configured to use Enterprise Extender without and with APIAS.

First, here are the systems using standard Enterprise Extender without APIAS, just looking at it from the point of view of the IP addresses involved:

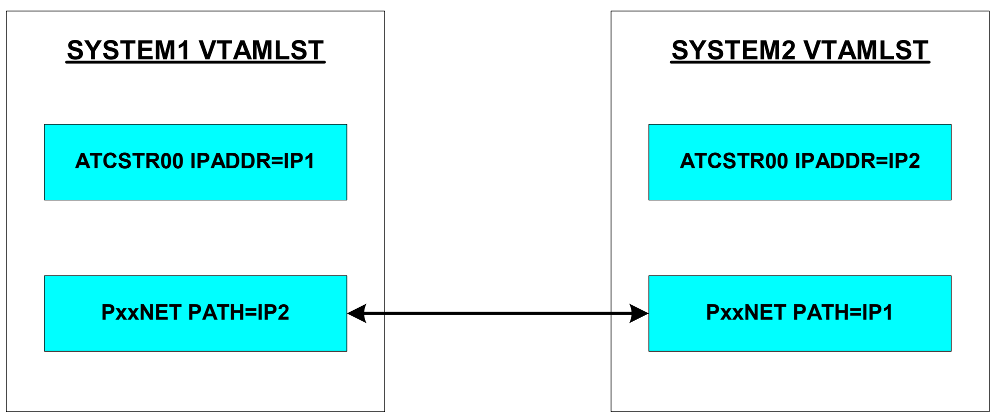


FIGURE 1. **Simple Illustration of EE without APIAS**

Here, the definitions are effectively a mirror image of one another on the two systems involved and the EE traffic passes between the two virtual IP addresses IP1 and IP2 on the primary IP stack.

Here is the situation when APIAS is SSL-securing the EE data between these two systems:

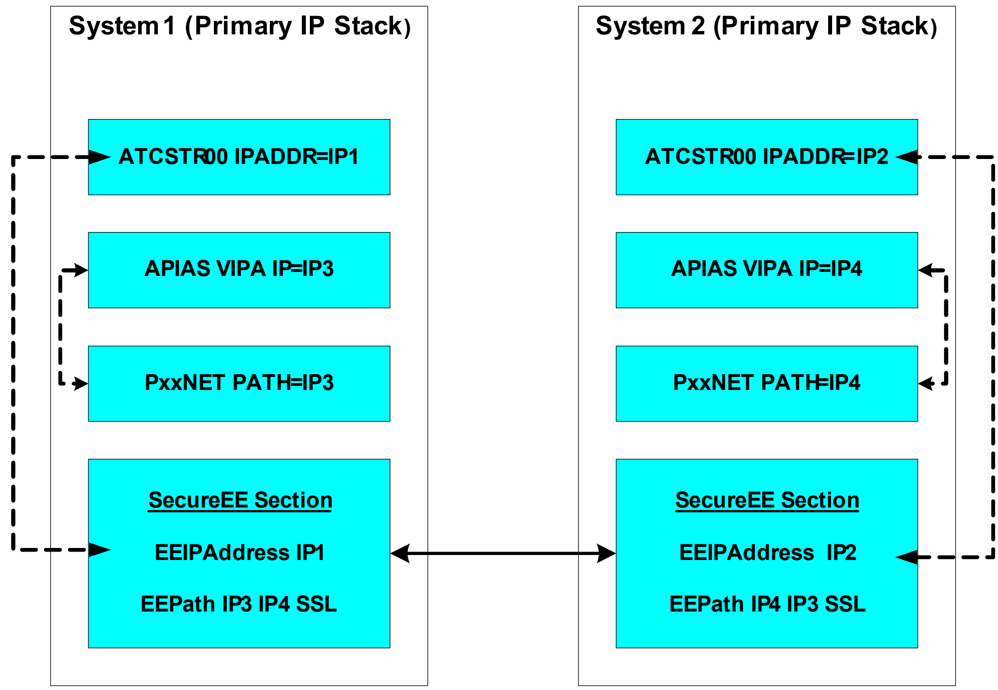


FIGURE 2. **Illustration of SSL-Secured EE with APIAS (Using Single Stack)**

In this situation defining a VIPA address for APIAS to use and the VTAM path change are the only system changes needed.

If your z/OS system does not have the appropriate PTF applied (see “System Prerequisites” on page 8), to enable both APIAS and EE to use the same ports on the primary stack, the setup is slightly more complicated as a second ‘secure stack’ has to be used for APIAS like this:

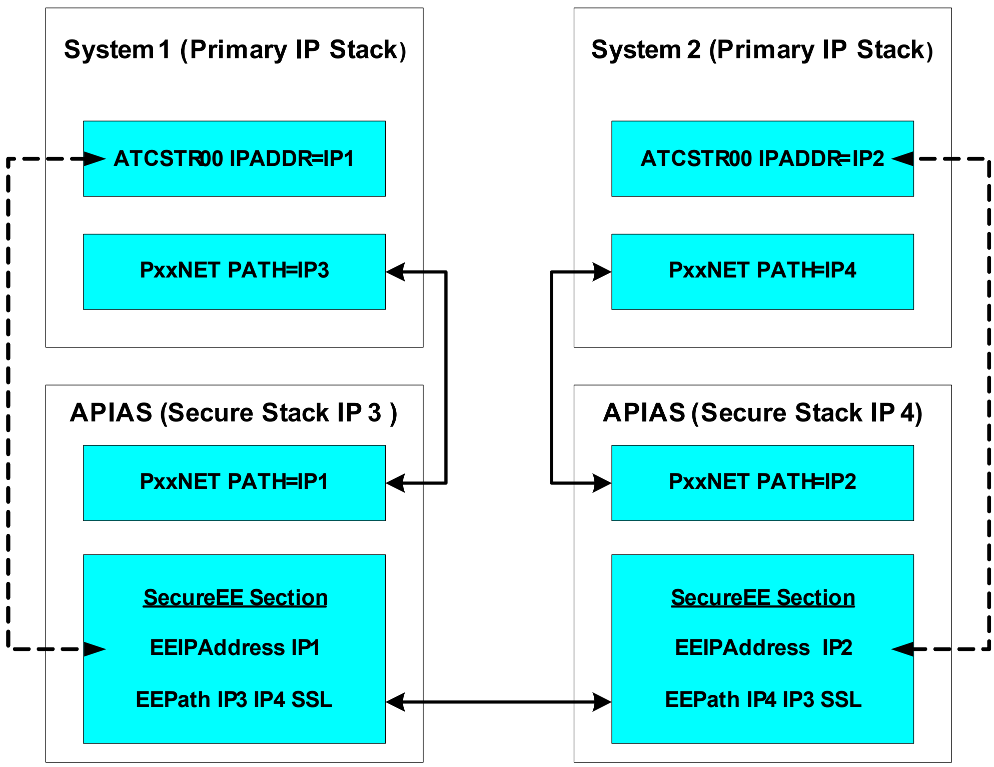


FIGURE 3. **Illustration of SSL-Secured EE with APIAS (Using Secure Stack)**

Here, the EE traffic is redirected by making an appropriate change to the VTAM resource definitions, from the UDP ports on the primary IP stack to APIAS on the secure IP stack. The secure EE feature, using the path between the secure IP stacks at IP addresses IP3 and IP4, SSL encrypts outbound EE packets, and decrypts inbound EE packets.

The APIAS SecureEE section on System1 would contain at least the statements:

```
EEIPAddress  IP1
EEPath System2  IP3  IP4
```

while the SecureEE section on System 2 would contain:

```
EEIPAddress  IP2
EEPath System1 IP4 IP3
```

Other relevant statements such as the Keyring, CertLabel and ClientAuth can also be defined but would most often be picked up from the Defaults section.

CHAPTER 4

Configuring for TCP EE

APIAS is able to allow you to use Enterprise Extender in your network even when the use of UDP protocol is disallowed. This chapter gives an overview of how the process works, and how to set up the appropriate definitions in VTAM, TCP/IP and APIAS to take advantage of this product feature.

Overview

If you are not allowed to have UDP traffic on your network the use of Enterprise Extender would normally be excluded. APIAS can remove this restriction by converting the EE packets to TCP packets which it then sends to the target system where APIAS converts them back to UDP prior to passing them on to the standard EE ports.

Although SSL encryption is not done in this mode (the assumption being that either it is not required, or that it is being done at the VTAM level), APIAS is optionally able to use digital certificates for authentication.

SecureEE Profile Statement

The parameters of the SecureEE APIAS profile statement define the EE TCP support. The SecureEE section specifies the EE IP address of the system on which APIAS is running and an EEPath parameter for each path to other systems in the network that are communicating with this system using EE. Parameters on the EEPath statement specify that the link is to use TCP protocol, and whether authentication using digital certificates is required.

If authentication is required, APIAS will use the Keyring and CertLabel parameter values specified in the Defaults section, however, these may be over-ridden by identical parameters on each EEPath if required. For the full description, see “SecureEE Section” on page 51.

System-Related Changes

The following system-related changes are required for APIAS’s secure EE feature to work:

- Alter the VTAM Resources

  The VTAM resources used to connect the source EE system to the target EE system need to be changed so that they cause the EE packets to be sent instead to the APIAS secure IP stack. The PATH statement in the VTAMLST member specifies the virtual device address defined in the secure IP stack (see above). It is recommended that a new VTAMLST member is created to hold the new definitions so that switching back to ‘standard’ EE can easily be done.
- Define a second TCP/IP stack for APIAS’s use.

  This step is *only* required when the z/OS system does not permit APIAS to ‘listen’ on the EE port numbers on a different (VIPA) IP address on the same stack that runs EE, see “System Prerequisites” on page 8. In this situation, APIAS must run on a different IP stack from the stack running the EE it is supporting. The second stack needs to have connectivity with the primary IP stack, typically using XCF or CTCs.

Note that there is no restriction on running 'normal' EE connections as well as APIAS TCP EE connections. They would, however, need to run across different links.

An Illustrated Example

Here is an illustration of the way two systems are connected using standard Enterprise Extender without APIAS, simply from the point of view of the IP addresses involved:

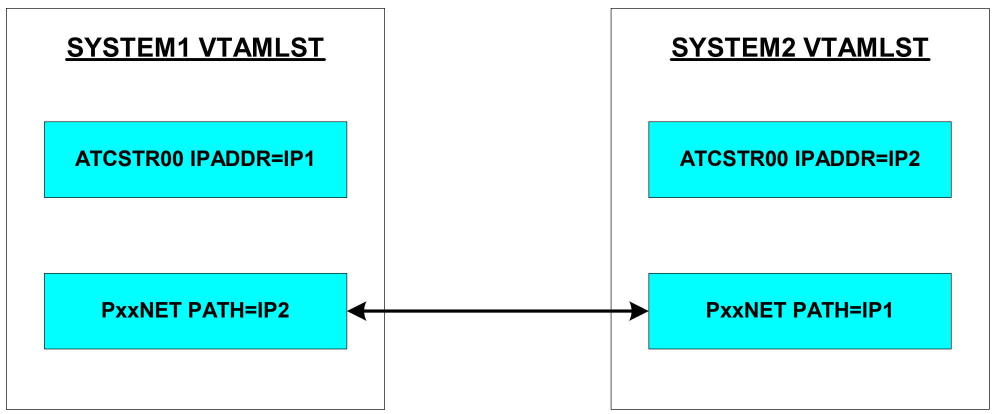


FIGURE 4. **Simple Illustration of EE without APIAS**

Here, the definitions are effectively a 'mirror image' of one another on the two systems involved and the EE traffic passes between the two virtual IP addresses IP1 and IP2 on the primary IP stack.

Here is the situation when APIAS is picking up the UDP packets from EE, converting them to TCP and sending them to the target system where the reverse process takes place:

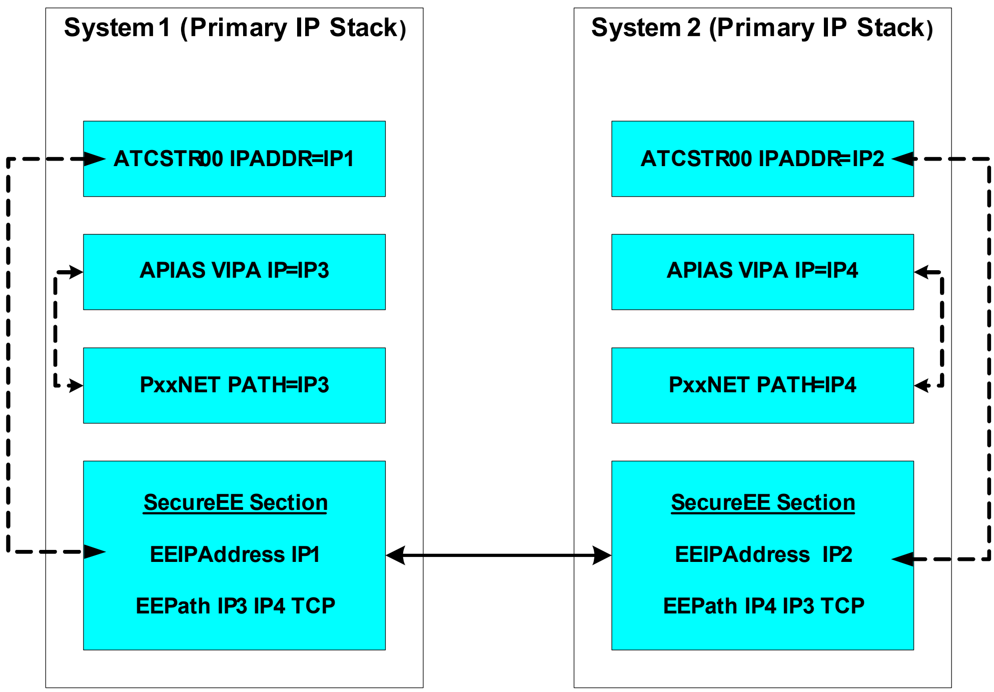


FIGURE 5. **Simple Illustration of TCP EE with APIAS (Using Single Stack)**

In this situation defining a VIPA address for APIAS to use and the VTAM path change are the only system changes needed.

If your z/OS system does not have the appropriate PTF applied (see "System Prerequisites" on page 8), to enable both APIAS and EE to use the same ports on the primary stack, the setup is slightly more complicated as a second 'secure stack' has to be used for APIAS like this:

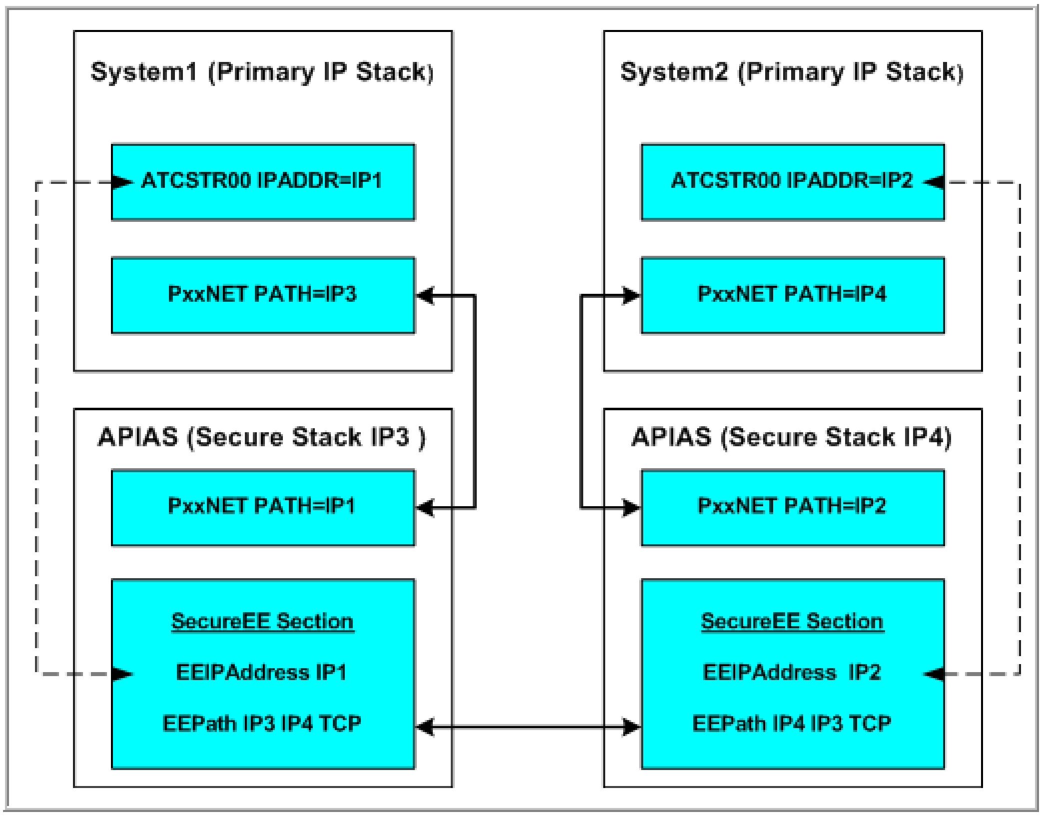


FIGURE 6. **Simple Illustration of TCP EE with APIAS (Using Secure Stack)**

Here, the EE traffic is redirected by making an appropriate change to the VTAM resource definitions, from the UDP ports on the primary IP stack to APIAS on the secure IP stack. The TCP EE feature, using the path between the IP stacks at IP addresses IP3 and IP4, converts outbound EE packets into TCP, and converts inbound TCP EE packets back into UDP.

The APIAS SecureEE section on System1 would contain at least the statements:

```
EEIPAddress  IP1
EEPath System2  IP3  IP4  TCP
```

and the SecureEE section on System 2 would contain:

```
EEIPAddress  IP2
EEPath System1 IP4 IP3  TCP
```

Other optional statements such as the Keyring, CertLabel and ClientAuth can also be defined but would most often be picked up from the Defaults section.

CHAPTER 5

Configuring for Multi-Stack EE

APIAS can allow you to bypass the restriction on running EE via a single stack and allow you to run it on as many IP stacks as you have active in an LPAR. This chapter gives an overview of how the process works, and how to set up the appropriate definitions in VTAM, TCP/IP and APIAS to take advantage of this product feature.

It should be noted that this feature uses standard UDP packets so as a side benefit enables sites that want to use standard EE but require digital certification to authenticate the two end-points to do so.

Overview

Standard EE does not have the capability of running through more than one IP stack in an LPAR. However, many companies do run multiple stacks, often so that communications paths are segregated for security reasons. This restriction means that if a site decides to adopt the EE approach, serious system changes need to be planned and implemented in order to enable this to occur.

APIAS is able to overcome this restriction so that EE can be used through all the IP stacks running in an LPAR. In this mode it is simply acting as a relayer of the UDP packets, passing them to an APIAS running in each stack from which they are routed to the appropriate target system. In this mode APIAS does not need to be running in the target system.

This capability can be combined with the other APIAS features. For example, if UDP packets are not allowed to flow on the external network, the APIAS EE TCP feature can be used to convert the packets from UDP to TCP, and again APIAS can optionally use digital certificates for authentication. If these additional facilities are used APIAS would need to be running at both the local and remote ends of the EE link so that the authentication can take place and so that the packets can be converted back from TCP to UDP.

Configuration Options

Configuring APIAS for multi-stack operation can be done in several ways. For example, it might be easiest (and in some cases required) to define a new IP stack just for APIAS and route all the EE traffic to that new stack. APIAS would need to be started on any other stacks that need to route EE packets. APIAS receives the UDP EE packets, performs any conversion if required, and then routes the packets on to the required target systems.

Alternatively, you could implement APIAS in the IP stack that is currently configured for EE. APIAS will pick up the packets, again perform any conversion required and relay the packets appropriately.

The decision as to how best to configure the APIAS environment is largely dependent upon the system environment, including both the IP stack configuration, software and and hardware definitions, and a company's requirements for EE processing. In any event, APIAS is flexible enough to cope with all situations, no matter how simple, or how complex.

SecureEE Profile Statement

The parameters of the SecureEE APIAS profile statement define the Multi-Stack EE support. The SecureEE section specifies the EE IP address to which EE packets are being routed and an EEPath parameter for each path to other systems in the network that are communicating with this system using EE. In this mode of operation, APIAS is normally configured to use UDP protocol, but the other features of the product may also be used both to authenticate the link and to do TCP and SSL conversion if required.

System-Related Changes

The following system-related changes are required for APIAS's secure EE feature to work:

- Optionally define an additional TCP/IP stack for EE.

  This step is *only* required when the z/OS system does not permit APIAS to 'listen' on the EE port numbers on a different (VIPA) IP address on the same stack that runs EE, see "System Prerequisites" on page 8. In this situation, APIAS must run on a different IP stack from the stack running the EE it is supporting. The additional stack needs to have connectivity with the other IP stacks, typically using XCF or CTCs.

- Alter the VTAM Resources

  The VTAM resources need to be changed so that EE packets are sent to the IP stack running APIAS. The PATH statement in the VTAMLST member specifies the virtual device address defined in the APIAS IP stack. It is recommended that a new VTAMLST member is created to hold the new definitions so that switching back to 'standard' EE can easily be done.

Note that there is no restriction on running 'normal' EE connections as well as APIAS EE connections. They would, however, need to run across different links.

An Illustrated Example

This section contains a simplified example of the way an LPAR running three IP stacks can be configured using APIAS so that all of them can use Enterprise Extender. Here is the situation without APIAS:

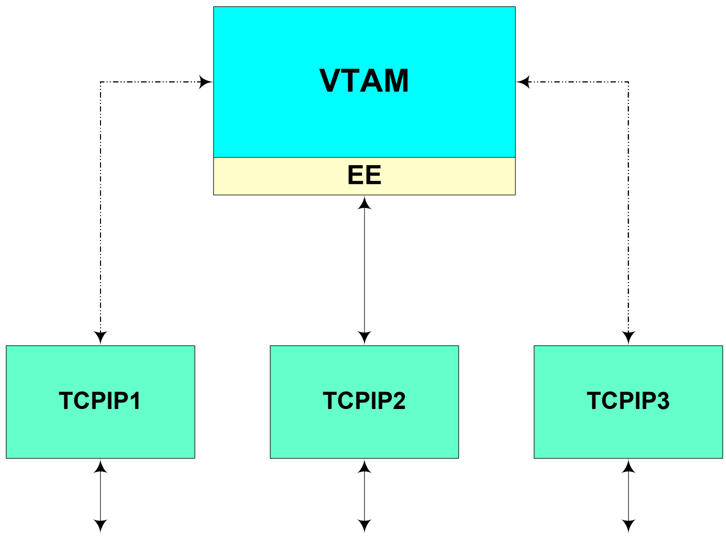


FIGURE 7. A Multi-Stack LPAR running EE without APIAS

Here we have three stacks with stack TCPIP2 designated as the EE stack, while TCPIP1 and TCPIP3 cannot support EE but may have inbound telnet connections and other services running.

For this example, we are going to define a fourth IP stack to be used for EE from which APIAS in the other stacks can route the packets appropriately. Note that this is not a mandatory requirement, APIAS could be activated for stack TCPIP2 and route the packets from there via the other stacks.

We need to activate an instance of APIAS for each of the IP stacks TCPIP1, TCPIP2 and TCPIP3 so that it can route the EE packets to the required destinations. For this example, we are going to assume that the eventual destinations for the EE packets from IP stacks TCPIP1, TCPIP2 and TCPIP3 are systems ZOSIP10 connected to using (VIPA) IP address IP10, ZOSIP11 using IP11 and ZOSIP12 using IP12 respectively. This will become clearer later where we look at the EEPath statements that need to be defined in each APIAS system to make this configuration work.

Here is the simplified diagram showing APIAS running in this LPAR:

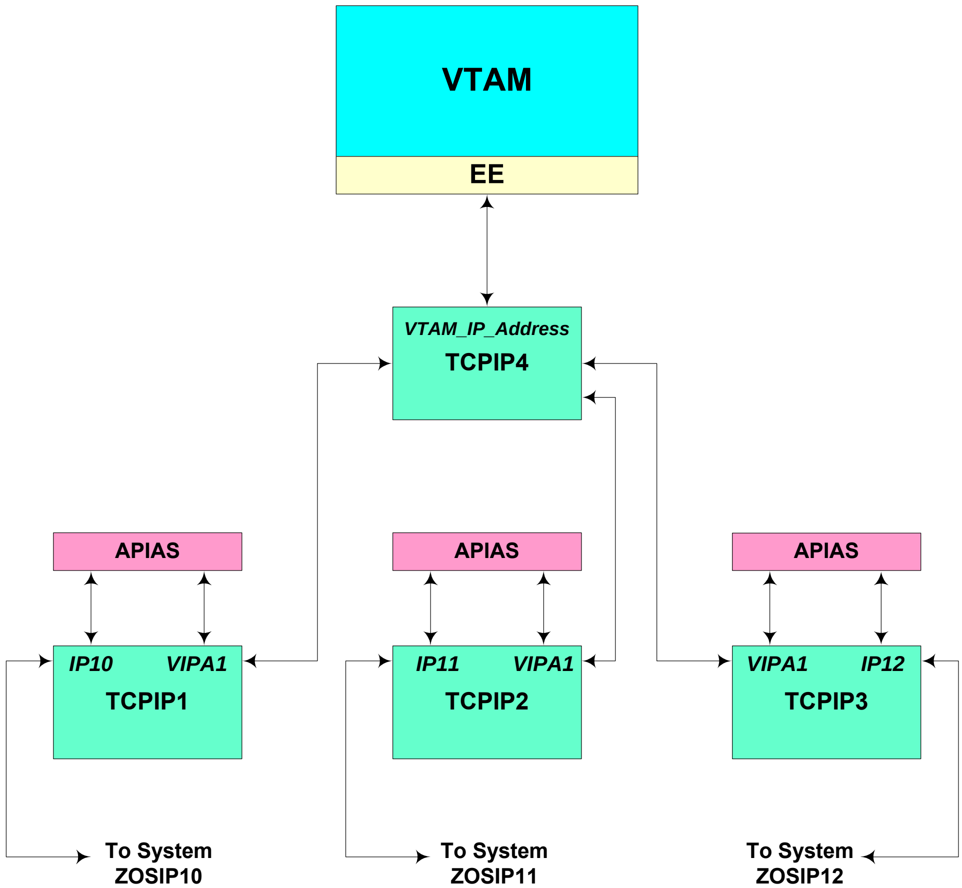


FIGURE 8. **An LPAR with APIAS Multi-Stack Support**

Here, the EE traffic is routed to our 'EE stack' TCPIP4 instead of its original TCPIP2 stack by making an appropriate change to the VTAM ATCSTR*nn* member. APIAS's Multi-Stack EE feature routes the EE UDP packets to the appropriate target systems.

As we said earlier, the TCPIP1, TCPIP2 and TCPIP3 stacks are required to route EE traffic to systems ZOSIP10 on IP address IP10, ZOSIP11 on IP address IP11 and ZOSIP12 on IP address IP12 respectively. To complete the example, we have assumed that there are two VIPA addresses on each stack, *VIPA1* that connects 'internally' to process the EE packets and *IP10*, *IP11* and *IP12* that connect to the remote systems. The APIAS configurations to support this arrangement would be as follows:

For stack TCPIP1:

```
Global Stackname TCPIP1
SecureEE
    EEIPAddr VTAM_IP_Address
    EEPath  ZOSIP10  TCPIP1_VIPA1  IP10  UDP
```

For APIAS on stack TCPIP2:

```
Global Stackname TCPIP2
SecureEE
    EEIPAddr VTAM_IP_Address
    EEPath ZOSIP11  TCPIP2_VIPA1  IP11  UDP
```

For APIAS on stack TCPIP3:

```
Global Stackname TCPIP3
SecureEE
    EEIPAddr VTAM_IP_Address
    EEPath ZOSIP12  TCPIP3_VIPA1  IP12  UDP
```

It also follows that in the PnnNET VTAMLST member for this LPAR, there are three PATH statements required to route the EE packets to the required stack:

```
VPU1   PU     PUTYPE=2,CPNAME=ZOSIP10,NETID=WDS,CPCP=YES,
              CONNTYPE=APPN,DWACT=YES,HPR=YES,TGN=3
VPA1   PATH   IPADDR=TCPIP1_VIPA1,GRPNM=GRP1

VPU2   PU     PUTYPE=2,CPNAME=ZOSIP11,NETID=WDS,CPCP=YES,
              CONNTYPE=APPN,DWACT=YES,HPR=YES,TGN=3
VPA2   PATH   IPADDR=TCPIP2_VIPA1,GRPNM=GRP2

VPU3   PU     PUTYPE=2,CPNAME=ZOSIP12,NETID=WDS,CPCP=YES,
              CONNTYPE=APPN,DWACT=YES,HPR=YES,TGN=3
VPA3   PATH   IPADDR=TCPIP3_VIPA1,GRPNM=GRP3
```

CHAPTER 6

Configuration Reference

This chapter describes how APIAS is configured once the product installation has been completed. It describes the APIAS control statements and parameters in full.

The APIAS Profile

The APIAS profile defines the configuration of an APIAS instance on your system. You can have a single APIAS running, or run multiple instances of APIAS on the same IP stack. For example, you might choose to have several APIAS started tasks running, each controlling the EE connection to a specific location. Alternatively, a single APIAS can manage all EE paths if required. The profile is defined in any PARMLIB you choose, but will usually be held for convenience in the APIAS PARMLIB created during the automated product install procedure.

When you activate APIAS, the name of the PARMLIB member that contains the control statements which configure the product is passed using the EXEC PARM. For example:

S APIAS,PROFILE=MYAPSPRF

This starts APIAS using the profile data saved in member ‘MYAPSPRF’ in the PDS that is pointed to by the PARMLIB DD statement in the APIAS procedure.

You can dynamically load a new or updated profile into APIAS at any time while the product is active, using the OBEYFILE command, for example:

F APIAS,OBEYFILE MYAPSPRF

The following messages will be written to the console to confirm processing:

APS0005I Apias processing profile member MYAPSPRF
APSW003I Command completed OK

APIAS Profile Syntax

The APIAS profile is a standard PDS member comprising fixed length 80 byte records. An asterisk (*) or semi-colon (;) in column one causes the line to be treated as a comment.

The profile contains a number of sections, each section defined using the following control statements:

TABLE 2. **APIAS Profile Sections**

Section	Number	Brief Description
Global	1	Defines Global setting for the APIAS system. Examples are the product password and TCP/IP stack name. For the full definition see page 43.
Defaults	0 or1	This optional statement defines default values used for EE paths (see SecureEE below). Parameters are available for example to specify the default buffer size for TCP/UDP/SSL communication, the name to be registered with the Workload Manager. For the full definition see page 45.
SecureEE	0+	This statement is used to configure APIAS's Enterprise Extender paths.
%Include	0+	The APIAS configuration file supports the inclusion of other members from the same PARMLIB in which the configuration file resides. This optional statement defines the name of the member to be included and can be specified as many times as required.

Each section is delimited by a section header and trailer statement:

section-name
        ...section parameters...
END*section-name*

So, the Global section starts with 'Global' and ends with 'EndGlobal' and so on. The only exception is the **%Include** statement for which there is no corresponding %EndInclude.

For example, the APIAS global section could look something like this:

Global
        global parameter 1
        global parameter 2
        ...
EndGlobal

All section delimiters and parameters can be in mixed case, although care should be taken when specifying HFS path names where the name is likely to be case sensitive.

Finally, the section parameters are positional and the asterisk ‘*’ can be used as a place holder if you wish to omit a default parameter.

Global Section

The Global section defines parameters which relate to the APIAS system as a whole, such as the product password and IP stack name APIAS is to use.

The full statement definition is as follows:

Global
LicenceKey ***password***
StackName ***TCP/IP stack name***
SSLProcess ***count***
UDPProcess ***count***
NONSWAPABLE **YES | NO**
SilentAbend **OFF**
EndGlobal

Global Section Parameters

LicenceKey ***password***

This mandatory parameter provides the fifteen character APIAS password. Without a valid and current password the product will not work.

StackName ***TCP/IP-stack-name***

Optional parameter to specify the TCP/IP stack name. This is the name defined on the TCPIPJOBNAME parameter in TCPIP.DATA. When omitted the default is 'TCPIP'.

SSLProcess ***count***

Optionally specify the number of APIAS (OE) processes started to handle SSL traffic. This must be a number in the range 1 to 16. When this parameter is omitted APIAS calculates the value dynamically based on the number of EEPATHs specified. APIAS can handle connections up to the value specified on the MAXFILEPROC statement in BPXPRMxx for each SSL process started, so when defining this value you need to consider the number of users you will have connected to applications via APIAS's SSL feature.

UDPProcess ***count***

Optionally specify the number of APIAS (OE) processes started to handle UDP traffic. This must be a number in the range 1 to 16. When this parameter is omitted APIAS calculates the value dynamically based on the number of EEPATHs specified. APIAS can handle connections up to the value specified on the MAXFILEPROC statement in BPXPRMxx for each UDP process started.

NONSWAPABLE <u>YES</u> | NO

By default and if YES is specified for this parameter, APIAS runs non-swapable. Specifying NO for this parameter enables APIAS to run in swapable mode.

SilentAbend OFF

This is a special debug option and should only be used at the request of William Data Systems.

If APIAS experiences an error in its SSL routines it will ‘silently’ abend with no information that the error has occurred, but the SSL support will no longer function.

To cause USS to take a SYSABEND dump in the event of an error in the SSL routines specify ‘SilentAbend off’.

Global Section Example

This section contains an example of a Global Section. As a minimum, the Global Section could contain just the LicenceKey parameter which would still enable the number of connections defined by the MAXFILEPROC parameter in BPXPRMxx for stack TCPIP to be managed by the product.

```
Global
      LicenceKey    JYJWHNCNYRVNGCF
      StackName     TCPIP1
      SSLProcess    1
EndGlobal
```

Defaults Section

This section defines global defaults that are applied to any SecureEE statements on which they are not overridden. It follows that this complete statement and all parameters on it are optional. However, if a Defaults statement appears and a parameter value is defined for it, then that value will be applied on every SecureEE statement on which that same parameter is not specified.

For example, specifying 'Buffer 2048' on the Defaults statement will cause that value to be applied to every SecureEE statement unless the Buffer parameter is used on the SecureEE statement to override the default value.

The full statement definition is as follows:

```
Defaults
    Buffer          size
    WLMName         name
    ClientAuth      YES | NO   AUDIT | NOAUDIT
    EPHEMERAL       ON | OFF
    KeyRing         HFS   HFS_name   HFS_password |
                    SAF   userid/SAF_keyring_name
    CertLabel       labelname
    LDAPServer      IP-address | Server-DNS-Name
    LDAPUserid      User-parameters
    LDAPPassword  Password
    INBOUNDPACING  ON | OFF
    OUTBOUNDPACING  ON | OFF
    MODE            TCB | SRB
    V2CIPHER        SSL V2 Cipher Specifications
    V3CIPHER        SSL V3 Cipher Specifications
    V3CIPHERX       nnnn
    V2TIMEOUT       SSL V2 Session Timeout Value
    V3TIMEOUT       SSL V3 Session Timeout Value
EndDefaults
```

Defaults Section Parameters

Buffer *size*

This parameter defines the buffer size in bytes to be used for TCP, UDP and SSL communications. When omitted the default is 8160 which it uses for UDP packets; the buffer size for TCP packets is calculated based on the UDP buffersize and the pacing parameter settings.

This buffer is used by APIAS to receive incoming data. There is always one buffer allocated to each connection on an APIAS port. Setting the buffer size inappropriately large will not cause any harm, other than wasting a little storage. If you decide to alter this value from the default, it would be reasonable to set the buffer size to the maximum SNA RU size.

Warning*: If you set this value too small, EE packets passed to APIAS will be truncated with consequent failure of data transfer sessions.*

WLMName **name**

Specify a name to be registered with the Workload Manager (WLM). If omitted, the default is for no registration to occur.

You can use WLM to set performance improvement parameters for APIAS. Registering APIAS with WLM is also useful since if an application is moved across LPARs WLM can be used to automatically update the DNS.

ClientAuth **YES | <u>NO</u> AUDIT | <u>NOAUDIT</u>**

EE connections always have a Server and a Client end. The Server end is the one with the 'higher' IP address. You can use APIAS to authenticate an EE connection even when using the TCP transport or Multi-Stack features (i.e. no SSL encryption).

Specifying 'YES' for this parameter enables you to force an EE client connection through APIAS to provide a valid digital certificate for which a copy exists in the z/OS (HFS or RACF) keyring. The default is 'ClientAuth NO'.

If you also specify the 'AUDIT' parameter, only valid if you have specified 'ClientAuth YES', the name and issuer associated with the client certificate are recorded in the APIAS audit log.

EPHEMERAL **<u>ON</u> | OFF**

By default, and when 'EPHEMERAL ON' is specified, APIAS uses ephemeral (system-assigned) ports at the client end of the connection (the 'lower' IP address) to communicate with the APIAS server. When EPHEMERAL OFF is specified, APIAS uses ports 12000-12004 for the connection.

All EE paths will use the value specified or defaulted here. If you require only selected EE paths to use system assigned ports, you can use the Ephemeral parameter after each EEPath (see page 52) to which it applies.

Note: In the event of a link failure, it can take some time for TCP to 'free' the ports it was using. In the default case this is not an issue since when the link is reactivated the system assigns five new ports for APIAS to use at the client end. However, if you are using EPHEMERAL OFF you must ensure that ports 12000-12004 have been freed prior to reactivating the link.

KeyRing ***type name password***

This parameter is relevant for all connections using certification.

type specifies the keyring type and must be SAF or HFS.

name identifies the keyring as follows:

For HFS keyrings, it is the full path of the HFS key database. Note that for HFS keyrings the *password* parameter is mandatory.

For SAF keyrings, it is the name of the SAF keyring. The name may be qualified by the addition of the userid owning the keyring in which case the format is: *userid/name*. This format is only mandatory if the keyring is not owned by the same userid as that under which APIAS runs.

password is only valid for *type* HFS and specifies the password associated with the key database, or the full path of the encrypted stash file associated with the key database.

CertLabel ***labelname***

This parameter is relevant to APIAS's SSL support and authenticated TCP support only.

Specify the label of the certificate to be used by APIAS. If this parameter is omitted, APIAS will use the default label in the keyring.

*Note that this parameter is **case sensitive** and must **exactly** match the label as it was originally defined irrespective of the keyring type.*

For HFS keyrings, if only a single certificate exists it is assumed to be the default certificate. SAF keyrings have no default certificate unless one of the certificates was added with the 'DEFAULT' keyword. This may mean that for SAF keyrings, the CertLabel will be a mandatory parameter.

LDAPServer ***IP-address* | *Server-DNS-Name***

If you are using an LDAP server to store and retrieve Certification Revocation Lists, specify the IP address or name of the server using this parameter. When you specify a name APIAS will require access to your DNS server to resolve the name to its IP address.

LDAPUserid ***User-parameters***

If you are using an LDAP server to store and retrieve Certification Revocation Lists, specify the user parameters required to logon to the server. These will be in LDIF format and be entirely dependent on how your LDAP structure is defined. An example might be:

cn=wds,ou=wdsstaff,dc=willdata,dc=com

where cn is the common name, ou is the organisational unit and dc is a domain component.

LDAPPassword ***Password***

Specify the password required to logon to the LDAP server. This parameter needs to be defined if you have specified the LDAPServer and LDAPUserid parameters above.

INBOUNDPACING <u>OFF</u> | ON

This parameter only applies to TCP transport.

When ON is specified for this parameter, APIAS can receive up to a maximum of two EE messages in a single TCP receive operation which can have a beneficial effect on CPU usage in APIAS and TCP/IP. When OFF is specified, which is also the default, APIAS receives a single EE message in each TCP receive.

OUTBOUNDPACING <u>OFF</u> | ON

This parameter only applies to TCP transport.

When ON is specified for this parameter, APIAS buffers a maximum of two outbound UDP packets into a single TCP stream which can have a beneficial effect on CPU usage in APIAS and TCP/IP. When OFF is specified, which is also the default, no buffering of UDP packets takes place so there is a one-to-one relationship between UDP packets received by APIAS and TCP packets sent on by APIAS.

MODE **<u>TCB</u> | SRB**

This parameter only applies to TCP transport.

Specifies the mode in which APIAS is scheduled by TCP for inbound packets. When TCB is specified, which is also the default, TCP/IP schedules an IRB; when SRB is specified TCP/IP schedules an SRB.

Running in SRB mode may improve throughput and reduce CPU consumption.

If MODE SRB is specified, you may not use SSL transport in the same APIAS address space.

V2CIPHER ***SSL V2 Cipher Specifications***

This parameter enables you to override the default cipher specifications if you are running SSL version 2. The SSL connection made by APIAS between the two ends of the EE link will then use the overrides provided here.

V3CIPHER ***SSL V3 Cipher Specifications***

This parameter enables you to override the default cipher specifications if you are running SSL version 3. The SSL connection made by APIAS between the two ends of the EE link will then use the overrides provided here.

V3CIPHERX ***nnnn***

This parameter enables you to override the default cipher specifications with a two-character cipher.

V2TIMEOUT ***SSL V2 Session Timeout Value***

This parameter may be used to override the SSL version 2 session cache timeout value. The value is a number of seconds between 0 and 100. A short SSL handshake is performed when a cached session exists since the session parameters have already been negotiated.

V3TIMEOUT ***SSL V3 Session Timeout Value***

This parameter may be used to override the SSL version 3 session cache timeout value. The value is a number of seconds between 0 and 86400. A short SSL handshake is performed when a cached session exists since the session parameters have already been negotiated.

Defaults Section Example

Here is a completed Defaults Section. Remember that all the specifications here can be overridden by the individual SecureEE statements. For this Defaults statement, any client accessing APIAS using SSL will be forced to provide a valid SSL certificate which will be verified against that held in the HFS key.kdb. APIAS will access the password for the key database from the encrypted stash file ‘key.sth’.

```
Defaults
        Buffer          1024
        WLMName         WLMAPIAS
        ClientAuth      Yes
        KeyRing         HFS /u/apias/key.kdb  /u/apias/key.sth
        CertLabel       WDS
EndDefaults
```

SecureEE Section

The SecureEE section defines the Enterprise Extender feature of APIAS.

The full statement definition is as follows:

```
SecureEE
    EEIPAddress   name  ipaddr
    EEPath        pathname  local_ip  remote_ip  protocol
                  <CERT> <TRACE> <STOP> <SERVER | CLIENT>
    EEPath        pathname  local_ip  remote_ip  protocol
                  <CERT> <TRACE> <STOP> <SERVER | CLIENT>
    ...
    Buffer        size
    WLMName       name
    ClientAuth    YES | NO   AUDIT | NOAUDIT
    EPHEMERAL     ON | OFF[1]
    KeyRing       type  name  password
    CertLabel     labelname
    LDAPServer    IP-address | Server-DNS-Name
    LDAPUserid    User-parameters
    LDAPPassword  Password
    INBOUNDPACING  ON | OFF
    OUTBOUNDPACING  ON | OFF
    MODE          TCB | SRB
    V2CIPHER      SSL V2 Cipher Specifications
    V3CIPHER      SSL V3 Cipher Specifications
    V2TIMEOUT     SSL V2 Session Timeout Value
    V3TIMEOUT     SSL V3 Session Timeout Value
EndSecureEE
```

The full parameter definitions apart from EEIPAddress and EEPath can be found under “Defaults Section” on page 45.

SecureEE Section Parameters

EEIPAddress *name ipaddr*

This mandatory parameter only needs to be coded once for any instance of APIAS that is to provide SSL support for Enterprise Extender.

‘*name*’ is any unique 8 character name for the ‘local’ host system on which this instance of APIAS is running.

1. If specified here, the EPHEMERAL parameter *must* appear after each EEPATH parameter to which it applies. EPHEMERAL is ignored for APIAS SERVERs.

The ‘*ipaddr*’ parameter specifies the IP address of the ‘local’ host and will be the same as that specified on the IPADDR parameter of the ATCSTR00 member in the VTAMLST library.

EEPath ***pathname local_ip remote_ip protocol*** **<CErt> <TRace> <STOP> <SErver | CLient>**

An EEPATH parameter is required for each EE path to a remote system, or to an APIAS running in another stack on the same LPAR. Any of the optional CErt, TRace and STOP keywords may be specified and in any order.

‘***pathname***’ is any unique 8 character name for the path.

‘***local_ip***’ is the IP address the local APIAS will listen on.

‘***remote_ip***’ is the IP address of the remote system. This can be the IP address that a remote APIAS is using to communicate with ‘this’ system. In a Multi-Stack configuration it could be a VIPA address in the stack in which APIAS is receiving the EE packets from VTAM.

It follows that the EEPath definitions on the ‘other’ systems will normally mirror those on the ‘local’ system, see the example in the following section.

Each EEPath parameter specified must specify a *unique* IP address for the local APIAS; this would normally be a VIPA address pre-defined, or an address within a dynamic VIPA IP Address range. If you specify the same *local-ip* for multiple EEPath parameters, an error condition will be raised and the configuration load will fail.

All remaining SSL-related parameters are optional. They are duplicates of the parameters available in the Defaults section, have identical meanings, and need only be specified if you want to override the value specified in the Defaults section. See page 45 for a full description of these parameters.

‘***protocol***’ is the transport protocol to be used for this path and can be set to ‘SSL’ (the default), TCP or UDP.

‘**CErt**’ specifies whether digital certificate authentication is required for TCP and UDP transport modes. Authentication is always performed for SSL mode. The default if this parameter is omitted is for no certificate checking for TCP and UDP modes.

‘**TRace**’ optionally activates APIAS tracing and is equivalent to issuing the ‘TRACE START PATH *path-name*’ command - see “TRACE Command” on page 60.

Note that this causes every packet passing across the link to be traced to the spool so will potentially generate huge amounts of output. It should therefore only be used after consultation with WDS. The default if this parameter omitted is for no tracing to be performed.

All defined EE paths are automatically started at APIAS initialization time unless the optional '**STOP**' parameter is specified. You can start such a path later using the START PATH command (see page 59).

APIAS TCP and SSL connections always have a Server and a Client end and by default APIAS sets the Server end to be the one with the 'higher' IP address. You can override this default using the optional **SErver** and **CLient** parameters. For example, if you want to force 'this' end of the path to be the SERVER, even if it has the lower IP address, add the 'SE' parameter to the EEPATH statement. Note that in this example, the 'other' end of the path must be defined as the Client otherwise no data exchange will take place.

SecureEE Section Example

In this simple example, EE paths are to be secured between three systems, SYSA, SYSB and SYSC. These three systems have VTAM IP addresses 192.168.1.20, 192.168.1.30 and 192.168.1.40. VIPA addresses are available in the ranges 192.168.11.200-232 on SYSA, 192.168.12.200-232 on SYSB and 192.168.13.200-232 on SYSC. For this example, we will have SSL EE paths from A to B and force SYSA to be the server even though it has the lower IP address, and an authenticated TCP path from A to C and from B to C. Here are the three SecureEE definitions:

SYSA

```
EEIPAddress  SYSA 192.168.1.20
EEPath SYSB 192.168.11.200 192.168.12.200 SSL SE
EEPath SYSC 192.168.11.201 192.168.13.201 TCP CERT
```

SYSB

```
EEIPAddress  SYSB 192.168.1.30
EEPath SYSA 192.168.12.200 192.168.11.200 SSL CL
EEPath SYSC 192.168.12.202 192.168.13.202 TCP CERT
```

SYSC

```
EEIPAddress  SYSC 192.168.1.40
EEPath SYSA 192.168.13.201 192.168.11.201 TCP CERT
EEPath SYSB 192.168.13.202 192.168.12.202 TCP CERT
```

A quick examination of the definitions will reveal that the definitions on each system for the other system have the same IP addresses but in reverse.

APIAS will activate all defined paths at initialization time by default, however, if you want to define a link but not start it you can do this by specifying the STOP parameter on the appropriate EEPATH statement. For example, to prevent APIAS from activating the SYSC path on the SYSA system in the example above, you would specify:

EEPath SYSC 192.168.11.201 192.168.13.201 TCP CERT **STOP**

In this case, you could then later use the START PATH command to activate the path like this:

F APIAS,START PATH SYSC

Active paths may also be stopped by replacing the START parameter in the example F command above with the STOP parameter. See “START Command” on page 59 and “STOP Command” on page 59 for further information, and see also the section “EE Path Activation and the START and STOP Commands” on page 61 for additional notes on the effect of starting and stopping EE paths.

%Include Statement

This statement is provided so that you can break a large configuration member into several smaller, more manageable members.

The statement is defined like this:

%Include *membername*

where *membername* is the name of a member in the same PARMLIB in which the configuration file resides. *membername* can contain any number of (valid) APIAS configuration statements.

CHAPTER 7

APIAS Commands

APIAS has a number of commands available that enable, for example, a changed profile to be brought on-line, or for tracing options to be altered. The commands are all described in this chapter.

Command Entry

APIAS commands can be entered from the operator console using the standard F (modify) command. For example, to bring a new APIAS profile on-line you would issue the command:

F APIAS,OBEYFILE PROFILE

The command is issued as spoon as you press Enter. A look at the system console reveals the message ‘APS0005I Apias processing profile member PROFILE’ and the results would be viewable in the SYSOUT dataset.

A list of the commands available is shown in the following section.

Commands

The following commands are available via the operator modify command:

TABLE 3. APIAS Commands

Command	Description
SHow	Displays the status of APIAS EE paths.
STArt	Used to start EE paths.
STOp	Used to stop EE paths.
OBEYfile	Causes the APIAS profile to be reloaded.
SHUTDOWN	Terminates APIAS cleanly.
TRACE	Used to start and stop traces.
DUMP	Takes an SVC dump of the APIAS address space.

Apart from Shutdown and Dump, these commands require parameters as shown below.

SHOW Command

This command may be used to display the status of all or a selected EE path.

SHOW **ALL | *pathname***

ALL
Displays the status of all pathnames specified on EEPATH parameters defined in the SecureEE section.

pathname
Displays the status of the specified pathname as defined on an EEPATH parameter in the SecureEE section.

START Command

START PATH ***pathname***

Use this command to start the EE path ***pathname*** which must be defined using an EEPATH statement (see page 52). This command would normally be used if a path were defined with the STOP parameter (and was therefore not started during APIAS initialization), or if the path had previously been stopped using the STOP command.

For example:

F APIAS,START PATH SYSCSSCP

For additional details see "EE Path Activation and the START and STOP Commands" on page 61

STOP Command

STOP PATH ***pathname***

Use this command to stop the path ***pathname***. Once this command has been processed no more data will flow on the defined path. For example:

F APIAS,STOP PATH SYSCSSCP

For additional details see "EE Path Activation and the START and STOP Commands" on page 61

OBEYFILE Command

OBEYFILE ***profile-name***

Specify the name of the profile that APIAS is to load.

This command provides the means to alter, for example, the product password, or define a new EE path, without having to end and restart the product. Note, however, that any changes made to the source of digital certificates, or to parameters relating to path definitions that are currently active, will not take place until the product is restarted.

SHUTDOWN Command

SHUTDOWN

Use this command when you want to shutdown APIAS. The product will terminate all active paths cleanly and then terminate. This command has no parameters.

TRACE Command

TRACE   START | STOP *trace-type*

trace-type

This mandatory parameter specifies a trace type to start or stop. Normally you would only use this command at the request of a William Data Systems technician since a great deal of spooled output can be created.

trace-type can be one of the following:

EVent		Initiates an event trace
STOrage		Initiates a storage buffer trace
PORT	*port IPaddr*	Initiates a detailed trace for a specific port
PATH	*pathname*	Initiates a detailed trace for a specific path

Note that once a trace is started it remains active until a TRACE STOP command is issued. In the case of PATH trace, this applies *even if the path is stopped and restarted* (using the START PATH and STOP PATH commands - see above).

DUMP Command

DUMP

Takes an SVC dump of the APIAS address space. This may sometimes be needed for diagnostic reasons but would only be requested by a WDS technician. This command has no parameters.

EE Path Activation and the START and STOP Commands

APIAS activates all defined paths at initialization time by default. However, if you want to define a link but not start it you can do this by specifying the STOP parameter on the appropriate EEPATH statement.

For example, suppose we have a path defined on a system SYSA to a system SYSC like this:

EEPath SYSC 192.168.11.201 192.168.13.201 SSL CERT

This particular definition is for a SSL Link to SYSC with certificate authentication and will by default be started at APIAS initialization time. To cause the path to be defined but not started, you add the STOP parameter to the end of the EEPATH definition (see “SecureEE Section” on page 51) like this:

EEPath SYSC 192.168.11.201 192.168.13.201 SSL CERT STOP

In this case, you could then later use the START PATH command to activate the path like this:

F APIAS,START PATH SYSC

Active paths may also be stopped by replacing the START parameter in the example above by the STOP parameter.

When using the STOP and START commands the following points should be noted:

1. As there are no connections when running in UDP only mode, the START/STOP PATH commands do not have any effect.
2. When running in TCP mode, the STOP command terminates all currently active TCP connection(s) on the specified path.
3. When running in SSL mode, the STOP command terminates the currently active secured TCP connection(s) on the specified path but for technical reasons only *when the next EE message flows on that connection*.

CHAPTER 8

Messages

This chapter contains a list of all information and diagnostic messages that may be encountered while running APIAS. System console messages appear in the first section followed by messages that are written in the the APIAS Audit Log.

Overview

APIAS messages to the system log are of the form ‘APSgnnns’ where ‘APS’ is the standard message prefix, ‘g’ indicates the message group, ‘nnn’ is the message number and ‘s’ is the severity code. The APIAS group can be set as follows:

TABLE 4. APIAS Message Groups

Code	Group
0	Core messages: Initialization, task, LU and socket related messages.
1	Tracing messages.
2	Configuration processing/editing messages.
9	Miscellaneous messages.

The severity code may be set as follows:

TABLE 5. APIAS Message Severity Codes

Code	Meaning
I	Information only. No action is necessary.
W	Warning. A warning message indicates a potential problem. No immediate action is necessary but be aware that further more serious messages may follow.
E	A severe error. Action may be required to resolve the situation. You may need to contact your local William Data Systems office for help.
N	Messages issued as a result of activating APIAS’s trace routines.

In the event of an error APIAS will attempt to recover and continue to operate, although at worst a task thread may abend causing loss of connectivity between two APIAS EE endpoints. If appropriate, you should try to recreate the problem before contacting William Data Systems for assistance, providing any diagnostic information or return codes given in the message text and in the output datasets - particularly SYSAUDIT and SYSTRACE.

In the message descriptions that follow variables appear in italics, for example *port-number*. These will always be replaced by values when the message is issued (in this example it might show as 14001).

In addition to the messages written to the system log, messages are also issued under the ‘Event’ column in the APIAS Audit Log. These are un-prefixed and are listed in the final section of this chapter.

Core Messages (APS0 Prefix)

TABLE 6. Core Messages (APS0 Prefix)

Message Number	Message text and explanation
APS0001I	Apias *version* Initialization in progress: PTF Level *ptflevel* Issued when APIAS has started initiation using the 'S' console command. The message text gives the current version and PTF level of the product which are essential pieces of information should you ever require technical support from William Data Systems.
APS0002I	Apias initialization complete Issued when APIAS has completed all of its initialization routines, this message is an indication that users will now be able to access the z/OS system using all APIAS defined virtual ports and direct telnet definitions.
APS0003W	Apias Shutting down Issued after the 'P' operator command has been issued to shut APIAS down. Any users still connected to the system through APIAS virtual ports will have their sessions terminated.
APS0004I	Apias ended The final message issued when APIAS has completed its shutdown routines.
APS0005I	Apias processing profile member *member-name* The message is issues as confirmation that APIAS is building its configuration from the profile member indicated in the message text by *member-name*.
APS0006E	Apias is not APF authorized For the SSL and Secure Enterprise Extender features to operate, APIAS must have access to modules in GSK.SGSKLOAD, or for z/OS 1.6 and higher *your-hlq*.SIEALNKE. This library should normally be in the system Link list, but if this is not the case, the library should be specified in the Apias STEPLIB and APF authorized. Further, the APIAS load library and SSL libraries must also be authorized. See page 17 for details on authorizing the APIAS load library.

TABLE 6. Core Messages (APS0 Prefix)

Message Number	Message text and explanation
APS0008I	Apias initialization complete The product initialization has been successfully completed.
APS0009E	Apias initialization aborted APIAS has encountered one or more serious errors during initialization and has terminated. Messages on the console, or in the printed output files will tell you what is wrong and these need to be rectified before retrying. If after reading the messages you are unable to resolve this problem, contact your local William Data Systems office with the details of the failure.
APS0010I	*taskname* task started APIAS starts a number of tasks during initialization and issues this message as each one successfully initiates. You will see the following tasks started: UDPn Handles UDP traffic connections SSLn Handles SSL connections CONTROL Main APIAS control task LOG Handles logging output
APS0011I	*taskname* task ended Normally issued during product closedown, these information messages are issued as each APIAS task is cleanly terminated.
APS0012I	*taskname* task detached APIAS has detached the task given in the message by taskname. This message is most often issued when a user has signed off.
APS0013E	*taskname* task abended This message is an indication that APIAS has detected a processing error in the task given by *taskname* and has abended the task. This may be due to something simple such as an APIAS user switching off their terminal without cleanly closing down their session first. If you are unable to determine the cause of this message, or if it is one of the APIAS internal tasks such as FP or SMTP, contact your local William Data Systems office for assistance.

TABLE 6. Core Messages (APS0 Prefix)

Message Number	Message text and explanation
APS0014E	Apias initialization partially complete Product initialization has completed but some non-fatal errors have been detected. This may or may not be an issue and you should check the SYSOUT dataset for error messages to verify if the system is running as you expect and if not make appropriate corrections.
APS0015E	Apias shutting down due to abend limit reached APIAS normally recovers from abends and continues to run. However, in extreme circumstances when it detects continual abends from which it is having to recover it may self-terminate after issuing this message. There will be other messages in the APIAS output print files that will give reasons for this failure. If you are unable to determine and correct the cause yourself, please contact your local William Data Systems office for assistance.
APS0016I	Apias profile errors detected This information message indicates that errors have been found in the product configuration profile. The product may still initiate despite this. You should check the SYSOUT dataset to see what the errors are and make corrections if necessary.
APS0017I	Apias at PTF level *ptf-level* This information message gives the latest PTF applied to the product and will be required by William Data Systems if you report a problem with the product.
APS0018I	Apias key enables features: *feature-list* Issued at initiation time, this message confirms the product features that are enabled by the current license key.
APS0034I	Operating under *opsys* version *version* When APIAS is initializing, it detects the operating system under which it is running and provides this information in this message. *opsys* will be set to OS/390 or z/OS and version will be set to the version of the operating system, for example ‘z/os version 1.4’.
APS0041I	Port *port-number* IP *IP-address* assigned to task *taskname* This information message lists the port numbers and IP addresses that have been assigned to APIAS tasks.

TABLE 6. Core Messages (APS0 Prefix)

Message Number	Message text and explanation
APS0045I	Port *port-number* Socket Error Ip=*IP-address* Call *request-type* A TCP/IP socket request *request-type* for *port-number* on IP address *IP-address* has failed. The connection to the remote APIAS involved will terminate and will need to be re-initiated if still required. There can be a variety of causes for this, the most likely being that the remote APIAS has been terminated. If you are unable to determine why the message has been issued contact your local William Data Systems office for assistance.
APS0048I	Port *port-number* Receive failed Ip=*IP-address* Fd=*file-descriptor* A TCP/IP socket receive for *port-number* on IP address *IP-address* has failed. file-descriptor gives the socket number of which the failure occurred. There can be a variety of causes for this and if you are unable to determine why the message has been issued contact your local William Data Systems office for assistance.
APS0050I	Allocating nnnn processes to *task* paths Indicates the number of processes allocated to the task(s) denoted by *task*.
APS0051I	Dubbing tasks to processes Each process in APIAS is attached as a TCB. This will generally occur when running in SSL mode.
APS0052I	Running non-swappable Confirmation that APIAS is running non-swappable.
APS0053I	*protocol* Buffer limits : Send *nnnnn* Receive *nnnnn* Confirms the maximum buffer sizes available in the TCP/IP configuration.
APS0058I	Port *port-number IP IP-address* Buffer sizes UDP *nnnnn* TCP *nnnnn* This information message is issued to let you know the buffer sizes being used for UDP and TCP data transfer on the port and IP address given in the message text. The TCP buffer size is based on a calculation given the UDP buffer size specified in the APIAS configuration.

TABLE 6. Core Messages (APS0 Prefix)

Message Number	Message text and explanation
APS0060I	Path *pathname* active for port *nnnnn* Confirmation that that the given port is now active for the path given in the message text.
APS0061I	Path *pathname* Outbound pacing active for port *nnnnn* Buffer size *nnnnn* Confirmation that the OUTBOUNDPACING ON parameter has been successfully processed and is now active for the given port number and is using the given buffer size.
APS0063I	Path *pathname* Inbound pacing active for port *nnnnn* Buffer size *nnnnn* Confirmation that the INBOUNDPACING ON parameter has been successfully processed and is now active for the given port number and is using the given buffer size.
APS0064I	Path *pathname* connection to port *nnnnn* pending This message is issued when APIAS has initiated a connection (as a client) to the destination IP address specified for *pathname*, but the connection has neither failed nor been established. This leaves the connection in a pending state. After a timer interval of about 15 seconds has expired APIAS would normally retry a completely failed connection, but in this situation cannot do so because the connection set-up is already pending. This message is issued at this point and APIAS waits for another timer interval before retrying. The most likely cause is that the remote port *nnnnn* shown in the message text is already in use, or something else in your system is preventing APIAS from establishing the connection. A netstat command will show the state of the APIAS IP address/port. It is possible that remote port *nnnnn* has been left in a CLOSE-WAIT or FIN-WAIT state from a previous connection. Ensure that the EE ports 12000-12004 on the remote system are available prior to attempting to start the path *pathname*.

TABLE 6. Core Messages (APS0 Prefix)

Message Number	Message text and explanation
APS0093	Module *module-name* had an error trying to free a buffer An internal storage management error has occurred. If this error message is seen repeatedly there could be an internal logic error. Contact your local William Data Systems office for assistance giving the module given in the message text by *module-name*. You will need access to the SYSPRINT, AUDIT and SYSTRACE output datasets when calling WDS.
APS0100I	*command* command received APIAS has received an operator command indicated in the message text by *command*. This message will be followed by a series of messages that give the response to the command.
APS0101I	*trace-type* trace started | stopped This message is issued as confirmation that the requested APIAS trace has been started or stopped. The type of trace is given at the start of the message text. See "START Command" on page 59 and "STOP Command" on page 59 for information on tracing.
APS0102I	Trace started | stopped for port *port* IP *IP-address* Issued as confirmation that the requested APIAS trace has been started or stopped for the given port and IP address. See "START Command" on page 59 and "STOP Command" on page 59 for information on tracing.
APS0103I	Obeying configuration member *member-name* This information message is written as confirmation that APIAS is processing an operator-issued OBEYFILE command and is currently processing the configuration member whose name is given in the message text.
APS0107I	Invalid command The command just issued is not recognized. See "APIAS Commands" on page 57 for a description of the valid commands.
APS0108I	Invalid parameter Although the command just issued is a recognized APIAS command, one or more of the parameters is not valid. Correct the errors and reissue the command.

TABLE 6. Core Messages (APS0 Prefix)

Message Number	Message text and explanation
APS0109I	End Indicates the end of the message set issued as a result of a keyed operator console APIAS command.
APS0110I	Path *system-id* port *port-number path-status-message IP-address* This is a general information message that provides status information for an APIAS path, that is the link between the ‘source’ system on which this message is issued and a target system given by *IP-address* in the message text. It follows that the path will have been defined using an EEPATH statement in the APIAS profile. *port-number* is the EE port number to which the message relates and *path-status-message* gives the current status of the path between the two systems. When *path-status-message* says ‘EE Link Failed’ it is an indication that either the physical or virtual EE link is down on *system-id*, or that APIAS at *system-id* is no longer active. Data will not be able to pass across the given path.
APS0120I	Path starting | stopped for port *port-number* IP *IP-address* This message is issued after a successful START or STOP PATH command has been issued. It gives the target port number and IP address for the EE path affected by the command. For further information see the section “EE Path Activation and the START and STOP Commands” on page 61
APS0150I	Apias using licence key *password-fields* This information message gives the password that APIAS has read from the GLOBAL Section in the configuration file and is using for the current initialization of the product. The password determines whether APIAS is licensed to run on this CPU, the expiry date and any optional features you might be using.
APA0151I	Apias detected CPU Id *cpuid* Model *model* During initialization, APIAS verifies that you are running the product on the CPU for which it has been licensed. This information message lets you know what APIAS has determined to be the CPU serial number (*cpuid*) and model (*model*).

TABLE 6. Core Messages (APS0 Prefix)

Message Number	Message text and explanation
APS0152I	Apias licence expired The password you are using to authorize the product has expired. APIAS will not function. Contact your local William Data Systems office for a new password.
APS0153I	Apias licence *licence-code* This information message tells you the password under which APIAS is currently running. The password determines the APIAS features you are able to use.
APS0154I	Apias licence invalid You have keyed an invalid APIAS password in the Global section of the profile that APIAS is using. This is most likely to be a keying error. If after checking you can not make the password work contact your local William Data Systems office for assistance.
APS0155I	Apias licence validation failed: Reason *reason-code* An unrecoverable error has occurred in the password processing routine that is not covered by one of the messages immediately above. Contact your local William Data Systems office for assistance.
APS0160I	Display for path *pathname* Issued as a result of the SHOW command, this gives the name of the path to which subsequent information messages relate.
APS0161I	Path transport *transport-type* Issued as a result of the SHOW command, this gives the transport type: UDP, TCP or SSL.
APS0162I	EE IP Address *IP-address* *IP-address* is the IP address of the ‘local’ host and will be the same as that specified on the EEIPAddress parameter in the SecureEE section (and the same as that on the IPADDR parameter of the ATCSTR00 member in the VTAMLST library).
APS0163I	APIAS IP Address *IP-address* This message gives the local IP address *IP-address* on which APIAS is listening.

TABLE 6. Core Messages (APS0 Prefix)

Message Number	Message text and explanation
APS0164I	Buffer length *nnnn* This message gives the buffer length nnnn that APIAS is using to receive incoming data.
APS0165I	Inbound pacing *xxx* This message gives the current setting for the INBOUNDPACING parameter, OFF (the default) or ON.
APS0166I	Outbound pacing *xxx* This message gives the current setting for the OUTBOUNDPACING parameter, OFF (the default) or ON.
APS0180I	Keyring *keyring-name* For APIAS SSL EE connections only, this message gives the name of the keyring.
APS0198I	Connections for path *pathname* This message precedes a list of 199I messages and gives the name of the path to which the following messages relate. It follows that *pathname* has been defined on an EEPath parameter in the SecureEE section.
APS0199I	Port *nnnnn* connection to *IP-address status* One of these messages is issued for each connection that APIAS is running to the target system. *nnnnn* gives the port number (12000 - 12004), IP-*address* is the IP address of the target system and *status* can be Active or Inactive.

Tracing Messages (APS1 Prefix)

These messages are issued as part of APIAS's trace routines and are included here only for completeness. You would not be using this feature of the product unless you had been asked to do so by one of William Data Systems' Technical Support Staff.

TABLE 7. Tracing Messages (APS1 Prefix)

Message Number	Message text and explanation
APS1002I	Trace (Get) [1] Len:[2] Blk:[3] Mod:[4]
APS1003I	Trace (Free) [1] Len:[2] Blk:[3] Mod:[4]/[5]
APS1005I	Trace (VPR) [4] Rvl:[1] Rcd:[2] Rsn:[3] Fd:{5}
APS1008I	Trace (UDP) RecvFrom Port=[1] IP=[2]
APS1009I	Trace (UDP) Sendto Port=[1] IP=[2]
APS1200I	Trace (DATA) [3] Address [1] Length [2]
APS1201I	Trace (+[1]) [2] [3] [4] [5] *[6]*
APS1249I	Trace (TTE) [1] [2] *[3]*
APS1250I	Trace (Abend) Apias Diagnostics
APS1251I	Trace (Abend) [1] [2] [3] [4] [5]
APS1252I	Trace (Abend) Comp Code [3] PSW [1] [2]
APS1253I	Trace (Abend) [1] [2] [3] [4] +[5]
APS1254I	Trace (Abend) End of diagnostics
APS1255I	Trace (Debug) {1} {2} {3} {4} {5}

Configuration Validation Messages (APS2 Prefix)

TABLE 8. Configuration Validation and Processing Messages (APS2 Prefix)

Message Number	Message text and explanation
APS2001I	Processing configuration member *member-name* Information message that gives the member name of the configuration file currently being processed.
APS2101E	**Error** Invalid parameter: *parameter-name* APIAS has detected an unknown parameter in the configuration file. The unknown parameter is given by *parameter-name* in the message text. This is a fatal error and needs to be corrected before retrying.
APS2102E	**Error** Invalid value : *value* APIAS has found a value that is not acceptable for the parameter on which it appears. This is a fatal error and needs to be corrected before retrying.
APS2103E	**Error** GLOBAL section missing The Global Section is mandatory since it specifies values without which the product cannot initialize, such as the product password. Correct this error and retry.
APS2104W	**Error** No EE paths defined It is not valid to have no EE paths defined. Ensure that you have at least one EEPath parameter in your SecureEE section.
APS2193E	**Error** Abend during post processing of *member-name* APIAS has detected errors during its post-processing phase. There will be preceding messages to indicate the errors that have caused the abend and these should be corrected before retrying.
APS2195I	Processing profile data completed code *nn* APIAS has completed the initial control statement validation phase. The completion code given in the message should normally be zero and you may want to check other error messages if it is not.

TABLE 8. Configuration Validation and Processing Messages (APS2 Prefix)

Message Number	Message text and explanation
APS2196I	Post processing profile data completed code *nn* APIAS has completed the post-processing validation phase. The completion code given in the message should normally be zero and you should check other error messages if it is not.
APS2197I	Post processing profile data ... APIAS has completed initial validation of the parameters and has started the post-validation processing phase.
APS2198E	**Error** Abend processing profile member *member-name* line *line-number* This message is issued when the validation routines have discovered an error so bad that there is nothing to be gained by continuing. Previous error messages will have been issued to enable you to identify the error(s) which should be corrected before retrying. APIAS will not start successfully with the configuration file in this condition.
APS2199E	**Error** Unable to open profile dataset member *member-name* APIAS has been unable to open the configuration profile. This is most likely to be a JCL error. Check that the PARM statement in the APIAS PROC is correctly specified, or that the correct profile name was specified when the product was initiated and that the PARMLIB DD statement(s) include the parameter library that contains the profile.

Miscellaneous Messages (APSW and APS9 Prefix)

TABLE 9. Miscellaneous Messages (APS9 Prefix)

Message Number	Message text and explanation
APSW990E	APIAS taking diagnostic dump This message confirms that APIAS has processed an operator console DUMP command and is now taking an SVC dump.
APS9010E	Port *port-number variable-message-text* APIAS has encountered a problem while accessing the certificate from the HFS database or RACF keyring specified by the Defaults or EEPath statement. *port-number* is the port on which it is attempting to start an SSL connection. The *variable-message-text* field will provide additional information on the specific error detected. If you are unable to resolve this given the additional information in the message, please contact your local William Data Systems office for technical support.
APS9011I	Using HFS database *HFS-database* Using HFS stash file *HFS-Stash-Filename* Using Label *Label-name* These information messages are issued when APIAS has successfully located the label *Label-name* from the HFS database indicated by *HFS-database* in the message text. The password has been read from the Stash File given in the message text by *HFS-Stash-Filename*.
APS9012E	Apias using depreciated SSL V3 API for task SSL1 This message indicates that the version of SSL that APIAS is having to use is not the most recent version 1 implementation of TLS available from z/OS v1.2 and above. By implication this is an indication that APIAS is running on an earlier release of the operating system than z/OS v1.2.
APS9013I	V*n Cipher nnnn cipher-name* Information message displaying available ciphers codes and names.

Audit Log Messages

As well as the messages given here, additional detail is shown under the 'Details' column in the log. Often the Details column will provide enough additional information to resolve a problem.

TABLE 10. Audit Log Messages

Message and explanation
Abort - Accept Error A serious fault has been detected during a APIAS's processing to accept an incoming EE client connection request. Console messages associated with this failure may help to determine the cause of this problem. If you are unable to proceed, contact your local William Data Systems office for technical support.
Abort - Client Auth A serious problem has occurred during client authentication. This could mean that the digital certificate is inaccessible, or does not match the certificate from the server system. The connection is aborted and the problem will need to be corrected before restarting APIAS.
Abort - No sockets APIAS is unable to continue processing because no sockets are available for EE datastream processing. This could be because the required sockets (12000 through to 12004 for EE and 12005 for authentication) are already being used, or because the sockets are blocked in the TCP/IP profile. This situation will need to be corrected before APIAS will work.
Abort - Send error An error has occurred sending socket data which has caused an abort. There may be additional messages on the console that give additional information regarding the cause of the error. If you are unable to resolve this issue, contact your local William Data Systems office for technical support.
Bind failed (TCP) APIAS has failed to bind to a TCP socket. The most likely cause is that the socket is not available, either because it is in use already, it has been blocked by the TCP/IP profile, or the IP stack is not active.
Bind failed (UDP) APIAS has failed to bind to a UDP socket. The most likely cause is that the socket is not available, either because it is in use already, it has been blocked by the TCP/IP profile, or the IP stack is not active.

TABLE 10. Audit Log Messages

Message and explanation
EE Authenticated When APIAS has successfully established and authenticated (using the specified digital certificate) a TCP or UDP link between two systems this message is issued as confirmation. APIAS uses port 12005 to perform its authentication handshake between the two ends of the EE link so this port must be available. Also, if there is a firewall between the two ends of the link, it must allow traffic to pass on port 12005 which may involve an alteration to the firewall configuration.
EE Buffer overflow The size of the EE packets is bigger than the allocated buffer so data is being lost. This should not normally happen if the buffer sizes are allowed to default (Buffer parameter on the Default or SecureEE statements). Ensure that the buffer size specified is large enough to cope with EE packets. See the description of the Buffer parameter in the Defaults section for details.
EE De-authenticated If an authenticated TCP or UDP EE connection fails, APIAS de-authenticates the port (12005) and issues this message as confirmation that this has happened.
EE Link accepted The server end of the EE path (the one that has the highest IP address) has accepted a connection request from an EE client.
EE Link Certificate This message indicates that a digital certificate is being processed for the link specified. The ‘Details’ column gives the name of the certificate label being used to authenticate the (UDP or TCP) connection between the source and target systems.
EE Link connected APIAS has successfully built a link between the server and client EE systems. The ‘Details’ column will give the buffer size being used. Messages will follow showing the link being initialized and the name of the certificate used for authentication if relevant.
EE Link ended An APIAS EE link has been cleanly terminated.

TABLE 10. Audit Log Messages

Message and explanation
EE Link error Although all the parameters of a SecureEE statement are valid, including the EEPath to the target system, APIAS has been unable to make the connection. This is normally because of some fault in the target system such as APIAS not being activated or a configuration fault in the EE or TCP/IP definitions. Correct the error and retry.
EE Link failed Although an EE link was initially successfully established between the server and client EE systems, a problem has subsequently occurred in the link. Further messages on the system log and audit log may help to ascertain what caused the problem which should be resolved prior to retrying.
EE Link initialized Confirmation message indicating that the EE link between two systems has been successfully initialized. The 'Details' column will give further information regarding the link.
EE Link recv error An error has been detected on an EE link immediately after it has been established, when APIAS attempted its first 'receive' of data on the link. So, the link connection crashed between the time APIAS successfully established/accepted a new (TCP) connection and tried to start receiving data. This could happen if the remote APIAS accepted the connection yet immediately closed it for some reason. An examination of the Audit Log at each end should help to establish why this message was issued.
EE Link rejected An attempt to initiate an EE link has failed.
EE Link stopped Confirmation message indicating that the EE link between two systems has been successfully stopped by the operator STOP command. The link can be restarted using the START command.
Listen failed APIAS has been unable to listen on the port because it is in use by another application. Alternatively, the corresponding IP address APIAS is trying to BIND to (the VIPA) is unavailable. This could be due to a wrong IP address being coded (EEIPAddress parameter) or the IP stack being unavailable.

TABLE 10. Audit Log Messages

Message and explanation
Listen stopped (TCP) This message is issued when the STOP command has been issued to stop an EE path and is an indication that the listener has been successfully terminated. The message 'EE link stopped' will be issued when the termination of the EE path has completed.
Listen thread abend This message indicates that the USS thread being used for listening on the port indicated has abnormally terminated. Data transmission will cease on the indicated port. Restart APIAS and contact your local William Data Systems office for technical support if this error occurs repeatedly.
Listening (SSL) APIAS has successfully initiated a the SSL Socket listener. Incoming connection requests can now be accepted.
Listening (TCP) APIAS has successfully initiated a the TCP Socket listener. Incoming connection requests can now be accepted.
Listening (UDP) APIAS has successfully initiated a UDP Socket listener. UDP traffic can be accepted on the indicated port.
Real thread abend A thread running a connection from an EE node has abended. This thread will have been handling the UDP packets from EE and without it the connection between the client and server EE systems will fail.
Socket Initialization This information message is issued when APIAS creates a socket. The 'Details' column will give further details such as the socket type - SSL, TCP or UDP, and the buffer sizes used for the send and receive processes. Additional messages in the audit log may follow giving additional details regarding the socket.
SSL Certificate This information message gives details of the digital certificate used for authentication of an EE link.

TABLE 10. Audit Log Messages

Message and explanation
SSL Error An error has been encountered on an SSL link between two systems. Further messages in the audit log should give you more details on the fault. If recovery is not possible the link will be terminated. If you are unable to resolve this issue contact your local William Data Systems office for technical support.
SSL handshake failed When APIAS initiates a connection for SSL the first process that takes place is the SSL handshake. Once the handshake (which involves digital certificate authentication) has succeeded, encrypted data can be safely passed across the link. This message indicates that a problem has occurred during this initial SSL handshake. Verify that you have connected the correct systems together and that the correct keyring and certificate name has been specified for the digital certificates. If you are unable to resolve this issue from this and any other messages that have been issued, contact your local William Data Systems office for technical support.
SSL Init Failed This message is issued when the SLL socket initialization does not complete normally. There should be additional text in the 'Details' giving a description of the reason for the failure. The error will need to be corrected before you can use APIAS's SSL support.
SSL Initialization These messages are issued when APIAS initializes the SSL environment. The messages report the SSL parameters in use for a defined SecureEE section. This information includes the type of SSL (v2, v3 and TLS) enabled, the session timeout values, the v2 and v3 ciphers and the UDP send and receive buffer sizes.
Virtual thread abend A thread running an SSL or TCP connection to a remote APIAS has abended. This thread will have been handling the transmission and receipt of converted (and possibly encrypted) packets and without it the connection between the client and server EE systems will fail.

CHAPTER 9

SMP/E Maintenance

This chapter contains notes on the SMP/E implementation of APIAS. It explains the reasons that APIAS is packaged using SMP/E, the advantages that accrue, and most importantly, the way you apply service to APIAS using the procedures provided with the product.

Introduction

APIAS is packaged and installed using SMP/E which uses a series of VSAM files to manage maintenance. To avoid conflict with other products on your system, APIAS uses its own set of VSAM datasets to manage the implementation (‘Stand-alone SMP/E Environment’), rather than trying to share the inventories of other products.

There are several benefits to this approach:

- WDS can provide maintenance more quickly
- You should never have to ‘wait for the next release’ to fix a problem
- You can easily see what maintenance (i.e. fixes) have been applied
- New maintenance can be easily and safely applied
- Bad maintenance can be restored/rejected (unlikely to occur)
- Dependencies between fixes can be enforced if required
- Hold reasons can be automatically applied to bring attention to other related issues

WDS software developers will resolve problems by modifying APIAS source code. The SMP/E PTFs are built from these tried and tested source code changes and the PTF as shipped to you contains the object code of the CSECT(s) changed.

When an SMP/E PTF is installed on your system, SMP/E takes the updated object code members and builds this into the executable code using the system linkage editor. When you restart APIAS it will be running with the new level of ‘fixed’ source code. Using SMP/E methodology means that AMASPZAP will never normally need to be used.

William Data Systems has made the process of applying PTFs very easy by (a) always requiring all preceding PTFs to be applied, and (b) providing simple panel driven procedures for generating the job control necessary for receiving, applying, accepting, restoring and rejecting PTFs. The procedures used are discussed in the following sections.

APIAS PTF Procedures

Overview

Some key datasets used in the PTF process are built during APIAS's automated install procedure:

hlq.TAPEIMGE
hlq.GLOBAL.CSI
hlq.PTFLIB

For example, if you specified during APIAS installation that the high level qualifier for your datasets should be 'APIAS.V311' then you will have datasets:

APIAS.V311.TAPEIMGE
APIAS.V311.GLOBAL.CSI
APIAS.V311.PTFLIB

on your system. The 'PTFLIB' is the target library for PTFs sent to you by WDS, and will contain all PTFs that were shipped at the time the product was originally installed consolidated into a single member called 'BASEPTF'. These will already have been applied to APIAS during the installation process. The 'TAPEIMGE' contains the REXX procedures for receiving, applying, accepting, restoring and rejecting PTFs.

Applying APIAS Maintenance (PTFs)

Overview

PTFs are shipped as a binary image of a TSO XMIT'd dataset in flat 80*3120 FB format. This dataset will be the PTFLIB that contains all available PTFs. To get this dataset on to your system and apply PTFs to APIAS follow these steps:

- BINary FTP the file provided to your system as a FB 80*3120 file
- Use the TSO RECEIVE command to receive the PTFs into *hlq*.PTFLIB
- Execute the 'PTFRECV' procedure from the *hlq*.TAPEIMGE dataset
- Execute the 'PTFAPPLY' procedure from the *hlq*.TAPEIMGE dataset

Once you have keyed the required PTFs into the panel driven PTF procedures above and submitted the generated job control, provided the jobs end with an acceptable return code the product can be ended and restarted. Reasons for receiving a non-zero return code and how to deal with that situation are discussed later.

The new PTF level is shown on the APIAS signon panel to the right of the product version. For example, 'APIAS 3.11 APS3115' means that you are running APIAS version 3.11 at PTF level APS3115. This also means that all PTFs 1 through to 14 are also applied. Here are the PTF application steps in more detail.

1. Binary FTP the PTF file to your system

The PTF file is called APS31*nn*.PTF.BIN or APS31*nn*.PTF.ZIP if it was zipped prior to e-mailing. If you have received a zipped file you will need to unzip it first - the .BIN file will be extracted from the zip file.

The binary file is a TSO XMIT'd dataset, 80 byte records in a 3120 byte fixed length block. To get this file onto your system, FTP to it and sign on with your normal user-id and password. Then, use the following commands as a guide:

type binary
literal site recfm=fb lrecl=80 blksize=3120
literal site cyl pri=5 sec=0
put APS31*nn*.ptf.bin
quit

The dataset *profile-prefix*.APS31*nn*.PTF.BIN will have been built on your system.

If you cannot use FTP you can use IND$FILE instead. In this case, the command format is:

IND$FILE PUT APS31*nn*.PTF.BIN RECFM(F) LRECL(80) BLKSIZE(3120)

If the generated statement does not have these attributes, you need to modify the IND$FILE utility to use the dataset attributes given above. It must not have ASCII or CRLF appended to the statement.

If you experience problems, or it is not possible to change the default parameters of IND$FILE, pre-allocate the dataset on the host system. (When the dataset is pre-allocated, the attributes of the pre-allocated file override the RECFM, LRECL and BLKSIZE specified on the IND$FILE PUT statement so you can omit them from the command).

2. Receive the PTFs into your PTF Dataset

From ISPF option 6 enter:

RECEIVE INDATASET('*profile-prefix*.APS31*nn*.PTF.BIN')

The input dataset is the one you built in step 1 above. At the prompt 'INMR906A Enter restore parameters or 'DELETE' or 'END' +' key the following:

DA('hlq.PTFLIB')

and press Enter. This will unpack the APIAS PTFs into the library *hlq*.PTFLIB.

3. Run the PTF Receive Procedure

The *hlq*.TAPEIMGE dataset built during APIAS installation contains a REXX procedure called 'PTFRECV' which builds the job control required to get the PTFs received on your APIAS system. If you prefer to do this process manually see the section 'Applying PTFs Manually' on page 90.

From ISPF option 6 enter the following command:

ex *hlq*.TAPEIMGE(PTFRECV)

This invokes the PTF procedure and a panel similar to this will be displayed:

```
----------------------- Apias Version 3.11 Maintenance -----------------------
Command ===> ____________________________________________________________________

To Receive specific PTFs enter the PTF number(s) below,
or leave blank to Receive all PTFs.

Specify PTFs to be Received _______  _______  _______  _______  _______
                            _______  _______  _______  _______  _______

SMP/E CSI                   APIAS.V311E.INSTALL.GLOBAL.CSI
PTF Library                 APIAS.V311E.RUNTIME.PTFLIB

Press Enter to review/edit the generated JCL.
Enter SUBMIT on the command line to start the installation.

Press END to abort






 F1=Help    F2=Split    F3=Exit    F9=Swap    F12=Cancel
```

Panel 4. APIAS PTFRECV Panel

All you need to do now is key the PTF numbers to receive. As a general rule it is a good idea to receive and apply all the PTFs provided. You can see the current PTF level both on the Signon panel and in the ‘Administer APIAS System’. For example, if the current PTF level is APS3103 and after TSO receiving the PTFs into the PTF library you see that PTFs APS3101 through to APS3109 are available, you will key PTF numbers from APS3104 through to APS3109 into this panel (since you already have APS3101, 02 and 03 applied). Note that you can simply press Enter without keying any PTF numbers on the panel. This will build a job that will RECEIVE all eligible PTFs.

Here is the panel completed:

```
----------------------- Apias Version 3.11 Maintenance -----------------------
Command ===> ____________________________________________________________________

To Receive specific PTFs enter the PTF number(s) below,
or leave blank to Receive all PTFs.

Specify PTFs to be Received _______  _______  _______  _______  _______
                            _______  _______  _______  _______  _______

SMP/E CSI                   APIAS.V311E.INSTALL.GLOBAL.CSI
PTF Library                 APIAS.V311E.RUNTIME.PTFLIB

Press Enter to review/edit the generated JCL.
Enter SUBMIT on the command line to start the installation.

Press END to abort






 F1=Help    F2=Split    F3=Exit    F9=Swap    F12=Cancel
```

Panel 5. Completed APIAS PTFRECV Panel

When you press Enter the procedure will then build the job control and give you the option to edit it, save it, or simply submit it immediately for processing. Here is the job built when Enter is pressed on the sample panel above:

```
//JOHNB1P JOB (),CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//*******************************************************/
//* Receive Apias maintenance                            */
//*******************************************************/
//RECEIVE  EXEC PGM=GIMSMP,REGION=4M,PARM='PROCESS=WAIT'
//SMPCSI   DD   DISP=SHR,DSN=APIAS.V311E.INSTALL.GLOBAL.CSI
//SMPHOLD  DD   DUMMY
//SMPPTFIN DD   DISP=SHR,DSN=APIAS.V311E.RUNTIME.PTFLIB(APS3104)
//         DD   DISP=SHR,DSN=APIAS.V311E.RUNTIME.PTFLIB(APS3105)
//         DD   DISP=SHR,DSN=APIAS.V311E.RUNTIME.PTFLIB(APS3106)
//         DD   DISP=SHR,DSN=APIAS.V311E.RUNTIME.PTFLIB(APS3107)
//         DD   DISP=SHR,DSN=APIAS.V311E.RUNTIME.PTFLIB(APS3108)
//         DD   DISP=SHR,DSN=APIAS.V311E.RUNTIME.PTFLIB(APS3109)
//SMPCNTL  DD   *
  SET BOUNDARY(GLOBAL).
  RECEIVE S(APS3104 APS3105 APS3106 APS3107 APS3108
            APS3109    ).
  LIST SYSMOD(APS3104 APS3105 APS3106 APS3107 APS3108
              APS3109    ) HOLDDATA.
//
//
```

You should always check that the RECEIVE completes with condition code 0000, or 0004 if you elect to receive all PTFs when some have previously been received - but see the note on HOLDDATA immediately below. Problems that may possibly occur are discussed on page 91. Usually problems are caused by errors that are easy to resolve - for example omitting PTFs from the list, or having errors in the job control deck.

Note that any HOLD data associated with PTFs will be reported during the receive job. You need to be aware of this since it will need to be bypassed during the second step of the process which is to APPLY the PTFs you have RECEIVEd. Also, if there is no HOLD data specified, the generated job will end with a highest return code of 0004 which is normal. In this situation provided the RECEIVE function ended with a return code of 0000 all is well and you can proceed with the PTFAPPLY step described next.

4. Run the PTF Apply Procedure

The *hlq*.TAPEIMGE dataset built during APIAS installation contains a REXX procedure called ‘PTFAPPLY’ which builds the job control required to get the PTFs applied to your APIAS system. If you prefer to do this process manually see the section ‘Applying PTFs Manually’ on page 90.

From ISPF option 6 enter the following command:

ex *hlq*.TAPEIMGE(PTFAPPLY)

This invokes the PTFAPPLY procedure and a panel similar to this will be displayed:

```
----------------------- Apias Version 3.11 Maintenance -----------------------
Command ===> __________________________________________________________________

To Apply specific PTFs enter the PTF number(s) below,
or leave blank to Apply all eligible PTFs.

Specify PTFs to be Applied _______  _______  _______  _______  _______
                           _______  _______  _______  _______  _______

Generate Apply Check       Yes
Bypass Hold reason         _______   e.g. ACTION DOC DEP AO

SMP/E CSI                  APIAS.V311E.INSTALL.GLOBAL.CSI

Press Enter to review/edit the generated JCL.
Enter SUBMIT on the command line to start the installation.

Press END to abort




 F1=Help    F2=Split    F3=Exit    F9=Swap    F12=Cancel
```

Panel 6. APIAS PTFAPPLY Panel

Using this panel you can generate (and submit) an APPLY CHECK to verify that the APPLY will run correctly as well as an APPLY. To generate an APPLY you must change the ‘Generate APPLY CHECK’ field from ‘yes’ to ‘no’. If there is any HOLD data you must key in the reasons after ‘Bypass HOLD reasons’ to APPLY the PTF(s) successfully.

Alternatively, you can simply press Enter without keying any PTF numbers on the panel. This will build a job that will APPLY all eligible PTFs, that is those that have been RECEIVEd but not yet applied.

Here is a completed example following on from the RECEIVE in the previous section for which there was no HOLD data:

```
----------------------- Apias Version 3.11 Maintenance -----------------------
Command ===> __________________________________________________________________

To Apply specific PTFs enter the PTF number(s) below,
or leave blank to Apply all eligible PTFs.

Specify PTFs to be Applied aps3104  aps3105  aps3106  aps3107  aps3108
                           aps3109  _______  _______  _______  _______

Generate Apply Check       no
Bypass Hold reason         _______   e.g. ACTION DOC DEP AO

SMP/E CSI                  APIAS.V311E.INSTALL.GLOBAL.CSI

Press Enter to review/edit the generated JCL.
Enter SUBMIT on the command line to start the installation.

Press END to abort




 F1=Help    F2=Split    F3=Exit    F9=Swap    F12=Cancel
```

Panel 7. Completed APIAS PTFAPPLY Panel

When you press Enter the procedure will then build the job control and give you the option to edit it, save it, or simply submit it immediately for processing. Here is the job built when Enter is pressed on the sample panel above:

```
//JOHNB1P JOB (),CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//*********************************************************/
//* Apply Apias maintenance                               */
//*********************************************************/
//APPLY    EXEC PGM=GIMSMP,REGION=4M,PARM='PROCESS=WAIT'
//SMPCSI   DD   DISP=SHR,DSN=APIAS.V311E.INSTALL.GLOBAL.CSI
//SMPHOLD  DD   DUMMY
//SMPCNTL  DD   *
  SET BOUNDARY(APSTZN).
  APPLY S(APS3104 APS3105 APS3106 APS3107 APS3108
          APS3109    )
  GROUPEXTEND.
//
```

If the APPLY CHECK option is used the only difference will be a ‘CHECK’ keyword immediately prior to the GROUPEXTEND.

Applying PTFs Manually

You can if preferred generate and run a job of your own to apply the PTFs in a single job once they have been SMP/E RECEIVEd. Here is a template job control deck that you can use as a guide for your own JCL:

```
//MAINT    EXEC PGM=GIMSMP,REGION=4096K,TIME=960,
// PARM='CSI=hlq.GLOBAL.CSI,PROCESS=WAIT'
//SMPPTFIN DD   DISP=SHR,DSN=hlq.PTFLIB(APS31xx)
//SMPHOLD  DD   DUMMY
//SMPCNTL  DD   *
  APPLY   S(
     APS31nn
   )
  GROUPEXTEND .
```

Replace the APS31*nn* with the one or many PTFs that you want to apply to APIAS.

Possible Problems Applying PTFs

The RECEIVE and APPLY of a PTF should normally complete with COND CODE 0000. Possible errors that can occur are:

- The PTF has already been received and/or applied
- The CSI dataset name is incorrect
- The PTFLIB dataset name is incorrect
- A prerequisite PTF is missing

If the PTF has already been received you simply need to apply it using the PTFAPPLY procedure, or by creating a simple JCL deck of your own to perform the apply. This situation is very unlikely to occur if you have used the WDS procedure.

If the PTF has already been applied, no further action is required. There are mechanisms in SMP/E to ‘un-apply’ (RESTORE) and ‘un-receive’ (REJECT) a PTF, which will be discussed later.

If the CSI or PTFLIB dataset name is incorrect, it most likely means that you have renamed or moved the datasets since the original APIAS installation. All you need to do in this case is to correct the job control deck (whether your own, or the one the APIAS PTF procedure has generated) and re-submit the job.

If a pre-requisite PTF is missing, it means that you are trying to install a PTF without having installed one or more preceding PTFs. To keep the system simple, APIAS PTFs are always sequentially numbered and each PTF relies on the application of all preceding PTFs.

This problem is easily rectified by installing all the pre-requisite PTFs in the same job. You do this by entering all the pre-requisite PTF names on the ISPF panel displayed by the procedure, or if you are using your own job control, by adding the pre-requisite PTFs to your JCL deck. Alternatively, on both the PTFRECV and PTFAPPLY you can press Enter with no data and the procedure will build a job that will receive or apply *all* eligible PTFs.

If you are familiar with the IBM supplied SMP/E dialogs you can use these to view the APIAS PTFs installed on the system.

Permanently Applying PTFs (Accepting PTFs)

Overview

When an APIAS PTF is installed using the RECEIVE and APPLY commands, the maintenance is built in the APIAS runtime libraries. Any PTF can also be removed from APIAS using the provided PTFRESTR (restore to 'un-apply) and PTFREJCT (reject to 'un-receive') procedures (see page 95).

There is another, optional, step which you may wish to do: the ACCEPT step.

The ACCEPT process has two implications:

- An accepted PTF cannot be backed out
- PTFs can only be backed out (restored and rejected) to the last ACCEPTed PTF.

In other words, if you have not ACCEPTed any PTFs, but want to back out a new PTF which is in error and that has been received and applied, you must ACCEPT all PTFs up to (but not including) the one in error, before restoring and rejecting the faulty PTF.

If you choose to ACCEPT *all* PTFs, you cannot back out any of the PTFs without first re-building the project afresh using the provided SETUP procedure in the *hlq*.TAPEIMGE dataset.

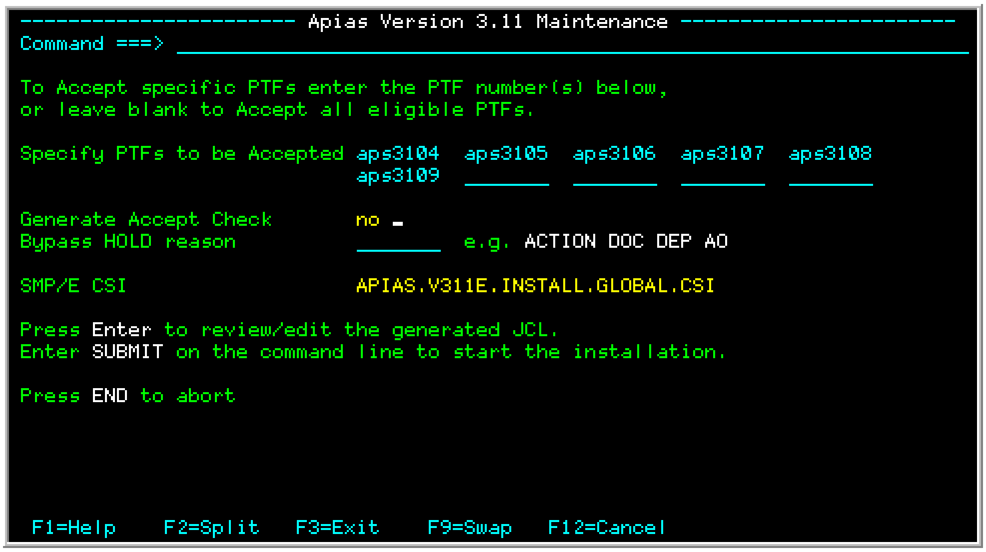
It is recommended that you should periodically ACCEPT PTFs which have been applied and are known to be stable. You will then be able to back out any new faulty PTFs to the last 'stable' level of APIAS.

Building the ACCEPT Job Using PTFACCPT

To make the job of ACCEPTing PTF easy, a REXX procedure is provided in the hlq.TAPEIMGE dataset called 'PTFACCPT'. To run it, use ISPF option 6 and enter the command:

ex 'hlq.TAPEIMGE(PTFACCPT)'

This will display a panel very similar to the PTF Apply panel into which you key the PTF numbers you want to ACCEPT. Alternatively, you can omit all the PTF fields to cause all eligible PTFs to be ACCEPTed. Here is an example panel with the PTF numbers already keyed:



Panel 8. Completed APIAS PTF Accept Panel

If you want to perform an ACCEPT CHECK prior to doing the full accept, you can leave the ‘Generate Accept check’ field as its default of ‘Yes’. In this particular example, we just want to build the ACCEPT job so have changed this field to ‘no’. As with the PTF APPLY procedure, if there is any HOLD data you will need to key the reasons into the ‘Bypass HOLD reasons’ to force the job to run.

When you press Enter, the job control will be built to perform the ACCEPT. Here is the deck built for the parameters above:

```
//JOHNB1P JOB (),CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//**************************************************************/
//* Accept Apias maintenance                                   */
//**************************************************************/
//ACCEPT   EXEC PGM=GIMSMP,REGION=4M,PARM='PROCESS=WAIT'
//SMPCSI   DD   DISP=SHR,DSN=APIAS.V311E.INSTALL.GLOBAL.CSI
//SMPHOLD  DD   DUMMY
//SMPCNTL  DD   *
  SET BOUNDARY(APSDZN) .
  ACCEPT S(APS3104 APS3105 APS3106 APS3107 APS3108
           APS3109    )
  GROUPEXTEND.
//
```

It is fine for the ACCPTF job to complete with a condition code of 0004.

Using your own ACCEPT Job Control

You can also use your own job control to do the ACCEPT process if you wish. Here is a template job control deck that you can use as a guide for building your own JCL:

```
//MAINT    EXEC PGM=GIMSMP,REGION=4096K,TIME=960,
// PARM='CSI=hlq.GLOBAL.CSI,PROCESS=WAIT'
//SMPHOLD  DD   DUMMY
//SMPCNTL  DD   *
 SET BOUNDARY(APSDZN) .
 ACCEPT   S(
    APS31nn
    APS31nn
    APS31nn
  )
  GROUPEXTEND .
```

Backing Out PTFs

Overview

In the unlikely event of WDS shipping a bad PTF which creates more problems than it resolves, WDS will take one of two approaches:

- Send a corrected version of the same PTF (i.e. with the same PTF-id)
- Send a new PTF which ‘supersedes’ the bad PTF (more likely)

In either case, you may need to back-out the bad PTF as soon as possible, probably even before WDS ships the replacement.

This is achieved using the SMP/E ‘RESTORE’ command which ‘un-applies’ the PTF (i.e. removes it from the executable code) setting APIAS back to its state prior to the PTF being applied in the first place. You are also able to REJECT (‘un-receive’) a PTF (i.e. remove it from your APIAS SMP/E environment completely) which will need to be done if WDS re-sends a corrected version of the same PTF, rather than a superseding PTF. Procedures for performing both of these activities are provided in the TAPEIMGE dataset built during product install.

However, the section ‘Permanently Applying PTFs (Accepting PTFs)’ on page 92 should also be taken into account. To un-install a single PTF (i.e. the last one APPLIED), you must ACCEPT all the preceding PTFs. If you do not do this, you must RESTORE all PTFs that have been applied up to the last ACCEPTed PTF.

If you have never ACCEPTed any APIAS PTFs, then you would have to back out all PTFs ever applied and re-install them after the RESTORE using the PTF procedure as described earlier. If WDS has shipped many PTFs for an APIAS version, this process could take several minutes. Therefore it is always advisable to ACCEPT PTFs when you are confident that APIAS is stable at the PTF level being ACCEPTed.

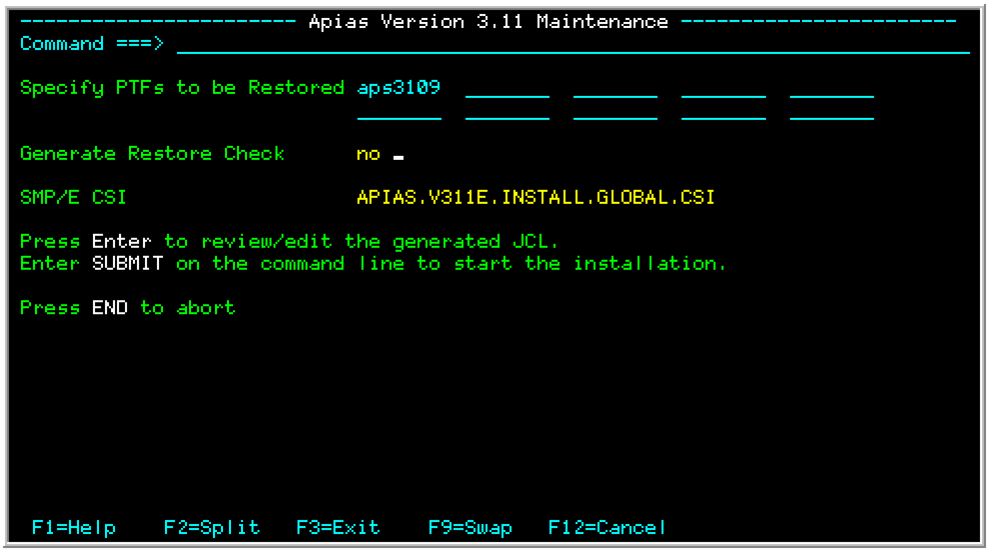
Building the RESTORE (un-apply) Job using the PTFRESTR Procedure

Procedure PTFRESTR is provided in *hlq*.TAPEIMGE to RESTORE (opposite of APPLY) PTFs. After running the job built by this procedure, the product should be restarted and will then be running without the named PTF(s) applied. Note that the PTF will still be received and could theoretically be re-applied. More normally it would be removed completely using the PTFREJCT (un-receive) procedure described in the following section.

To run the procedure, use ISPF option 6 and enter the command:

ex ‘hlq.TAPEIMGE(PTFRESTR)’

Here is the PTFRESTR panel with one PTF keyed:



Panel 9. Completed APIAS PTFRESTR Panel

If you want to perform a RESTORE CHECK prior to doing the full restore, you can leave the ‘Generate Restore check’ field as its default of ‘Yes’. In this particular example, we just want to build the ACCEPT job so have changed this field to ‘no’. As with the PTF APPLY procedure, if there is any HOLD data you will need to key the reasons into the ‘Bypass HOLD reasons’ to force the job to run.

When Enter is pressed, the PTFRESTR procedure will build the job control to remove the faulty PTF(s) from APIAS. Here is the JCL generated from the above panel:

```
//JOHNB1P JOB (),CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//**************************************************************/
//* Restore Apias maintenance                                  */
//**************************************************************/
//RESTORE  EXEC PGM=GIMSMP,REGION=4M,PARM='PROCESS=WAIT'
//SMPCSI   DD   DISP=SHR,DSN=APIAS.V311E.INSTALL.GLOBAL.CSI
//SMPHOLD  DD   DUMMY
//SMPCNTL  DD   *
  SET BOUNDARY(APSTZN).
  RESTORE S(APS3109
                )
   .
//
```

Running this job will restore the PTF so APIAS will run without that PTF after restart.

It is most probable that you would not be using this procedure without contact with your local WDS Support office.

Using Your Own RESTORE Job Control

You can also use your own job control to do the RESTORE process if you wish. Here is a template job control deck that you can use as a guide for building your own JCL:

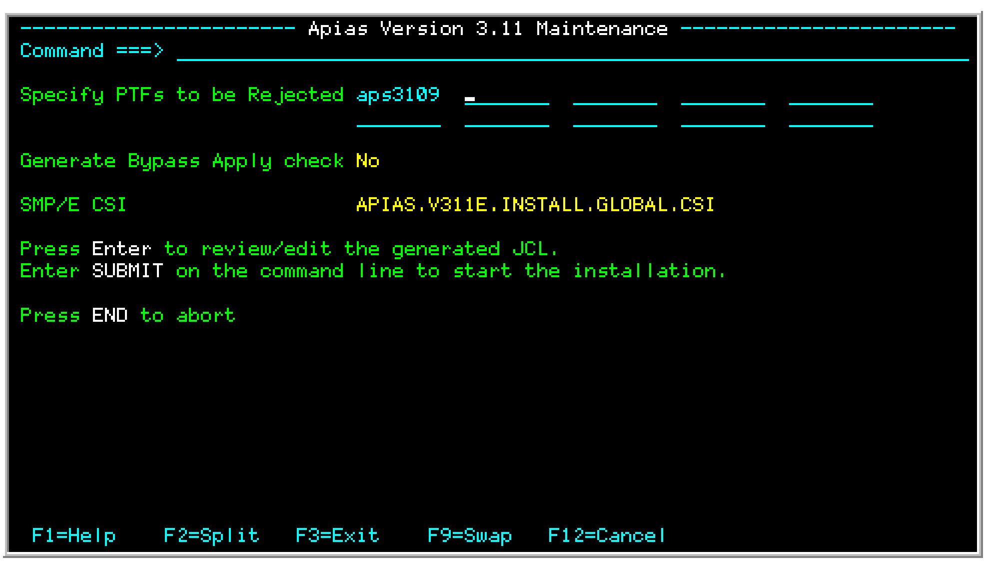
```
//MAINT    EXEC PGM=GIMSMP,REGION=4096K,TIME=960,
// PARM='CSI=hlq.GLOBAL.CSI,PROCESS=WAIT'
//SMPHOLD  DD   DUMMY
//SMPCNTL  DD   *
 SET BOUNDARY(APSTZN) .
 RESTORE   S(
    APS31nn
  ) .
SET BOUNDARY(GLOBAL) .
  REJECT S(
    APS31nn
 ) .
//
```

Building the REJECT (un-receive) Job using the PTFREJCT Procedure

Procedure PTFREJCT is provided in *hlq*.TAPEIMGE to REJECT (‘un-RECEIVE’) PTFs which fully removes it from the SMP/E environment. To run it, use ISPF option 6 and enter the command:

ex ‘hlq.TAPEIMGE(PTFREJCT)’

Here is the PTFREJCT panel with one PTF keyed:



Panel 10. Completed APIAS PTFREJCT Panel

If you want to perform a check to see whether the PTF has been applied prior to doing the full reject, you can alter the ‘Generate Bypass Apply check’ field from its default of ‘No’ to ‘yes’. In this particular example, we just want to build the REJECT job so have left this field set to ‘No’.

When Enter is pressed, the PTFREJCT procedure will build the job control to delete the faulty PTF from the SMP/E environment. Here is the JCL generated from the above panel:

```
//JOHNB1P JOB (),CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//*********************************************************/
//* Reject Apias maintenance                              */
//*********************************************************/
//REJECT   EXEC PGM=GIMSMP,REGION=4M,PARM='PROCESS=WAIT'
//SMPCSI   DD   DISP=SHR,DSN=APIAS.V311E.INSTALL.GLOBAL.CSI
//SMPHOLD  DD   DUMMY
//SMPCNTL  DD   *
  SET BOUNDARY(GLOBAL).
  REJECT S(APS3109
               )
  .
//
```

It is most probable that you would not be using this procedure without contact with your local WDS Support office.

Using Your Own REJECT Job Control

You can also use your own job control to do the REJECT process if you wish. Here is a template job control deck that you can use as a guide for building your own JCL:

```
//MAINT    EXEC PGM=GIMSMP,REGION=4096K,TIME=960,
// PARM='CSI=hlq.GLOBAL.CSI,PROCESS=WAIT'
//SMPHOLD  DD   DUMMY
//SMPCNTL  DD   *
SET BOUNDARY(GLOBAL) .
  REJECT S(
    APS31nn
 ) .
//
```

CHAPTER 10

Appendix 1 Audit Log Reports

Here are the Audit Logs from two systems called ZOS12 and ZOS14 connected using APIAS's EE TCP protocol support. First here is the APIAS configuration from the ZOS12 system:

```
Global
  licencekey  JCJWHJCWYRKBGCJ
  stackname   TCPIP2
EndGlobal
SecureEE
   EEIPAddress   Zos12   192.168.14.150
   EEPath        Zos14   192.168.19.50   192.168.15.50 TCP
EndSecureEE
```

and this is the APIAS configuration from the ZOS14 system:

```
Global
  licencekey  JCJWHJCWYRKBGCJ
  stackname   TCPIP2
EndGlobal
SecureEE
   EEIPAddress   Zos14   192.168.1.150
   EEPath        Zos12   192.168.15.50  192.168.19.50 TCP
EndSecureEE
```

In this example the connection between the two systems is running through a second TCP/IP stack called TCPIP2 in both cases. Overleaf are the audit logs from the two systems.

Here is the Audit Log from the ZOS12 system:

```
Apias 3.10 - Audit Log                                                                         Page 1
Time     EEId     Path     Event                   ------- Real -------- ------ Server ------- ----------- Details ------------
                                                   Port  IP Address      Port  IP Address
11:15:56 ZOS12    ZOS14    Listening (UDP)         12003
11:15:56 ZOS12    ZOS14    Listening (TCP)         12003
11:15:56 ZOS12    ZOS14    Listening (UDP)         12000
11:15:56 ZOS12    ZOS14    Listening (TCP)         12000
11:15:56 ZOS12    ZOS14    Listening (UDP)         12004
11:15:56 ZOS12    ZOS14    Listening (TCP)         12004
11:15:56 ZOS12    ZOS14    Listening (UDP)         12001
11:15:56 ZOS12    ZOS14    Listening (TCP)         12001
11:15:56 ZOS12    ZOS14    Listening (UDP)         12002
11:15:56 ZOS12    ZOS14    Listening (TCP)         12002
11:16:26 ZOS12    ZOS14    EE Link connected       12003 192.168.15.50
11:16:26 ZOS12    ZOS14    EE Link connected       12002 192.168.15.50
11:16:26 ZOS12    ZOS14    EE Link connected       12004 192.168.15.50
11:16:26 ZOS12    ZOS14    EE Link connected       12001 192.168.15.50
11:16:26 ZOS12    ZOS14    EE Link connected       12000 192.168.15.50
```

This is the Audit Log from the second system:

```
Apias 3.10 - Audit Log                                                                         Page 1
Time     EEId     Path     Event                   ------- Real -------- ------ Server ------- ----------- Details ------------

                                                   Port  IP Address      Port  IP Address
09:41:37 ZOS14    ZOS12    Listening (UDP)         12003
09:41:37 ZOS14    ZOS12    Listening (UDP)         12000
09:41:37 ZOS14    ZOS12    Listening (UDP)         12004
09:41:37 ZOS14    ZOS12    Listening (UDP)         12001
09:41:37 ZOS14    ZOS12    Listening (UDP)         12002
09:41:37 ZOS14    ZOS12    EE Link connected       12002                 12002 192.168.19.50
09:41:37 ZOS14    ZOS12    EE Link connected       12001                 12001 192.168.19.50
09:41:40 ZOS14    ZOS12    EE Link connected       12003                 12003 192.168.19.50
09:41:40 ZOS14    ZOS12    EE Link connected       12004                 12004 192.168.19.50
09:41:40 ZOS14    ZOS12    EE Link connected       12000                 12000 192.168.19.50
```

In this example, we can see APIAS building an SSL connection between two systems. Under the details column to the right you can see the processes APIAS is going through while it builds the secure connection between the two systems:

```
Apias 3.10 - Audit Log                                                                                      Page 1
Time     EEId     Path     Event                    ------- Real -------- ------ Server ------- ----------- Details ------------
                                                     Port  IP Address      Port  IP Address
15:18:55 SYSA01            SSL Initialization                                                    SSL V2 Enabled
15:18:55 SYSA01            SSL Initialization                                                    SSL V3 Enabled
15:18:55 SYSA01            SSL Initialization                                                    TLS V1 Enabled
15:18:55 SYSA01            SSL Initialization                                                    SSLV2 Timeout 100
15:18:55 SYSA01            SSL Initialization                                                    SSLV3 Timeout 86400
15:18:55 SYSA01            SSL Initialization                                                    Keyring /u/apias/key.kdb
15:18:55 SYSA01            SSL Initialization                                                    V2 Cipher 642
15:18:55 SYSA01            SSL Initialization                                                    V3 Cipher 090306020100
15:18:59 SYSA01   SYSD01   Socket initialization  *12004 192.168.55.50                           UDP Send/Recv Size 65535/65535
15:18:59 SYSA01   SYSD01   Listening (UDP)         12004
15:18:59 SYSA01   SYSD01   Socket initialization  *12004 192.168.55.50                           TCP Send/Recv Size 16384/16384
15:18:59 SYSA01   SYSD01   EE Link connected      *12004 192.168.55.50   12004 192.168.56.50   Buffer size 65535
15:19:00 SYSA01   SYSD01   EE Link initialized    *12004 192.168.55.50   12004 192.168.56.50   TLSV1 Cipher:09 Type:Client
15:19:00 SYSA01   SYSD01   EE Link Certificate    *12004 192.168.55.50   12004 192.168.56.50   Label:Default
15:19:02 SYSA01   SYSD01   Socket initialization  *12003 192.168.55.50                           UDP Send/Recv Size 65535/65535
15:19:02 SYSA01   SYSD01   Listening (UDP)         12003
15:19:02 SYSA01   SYSD01   Socket initialization  *12003 192.168.55.50                           TCP Send/Recv Size 16384/16384
15:19:02 SYSA01   SYSD01   EE Link connected      *12003 192.168.55.50   12003 192.168.56.50   Buffer size 65535
15:19:03 SYSA01   SYSD01   EE Link initialized    *12003 192.168.55.50   12003 192.168.56.50   TLSV1 Cipher:09 Type:Client
15:19:03 SYSA01   SYSD01   EE Link Certificate    *12003 192.168.55.50   12003 192.168.56.50   Label:Default
15:19:05 SYSA01   SYSD01   Socket initialization  *12002 192.168.55.50                           UDP Send/Recv Size 65535/65535
15:19:05 SYSA01   SYSD01   Listening (UDP)         12002
15:19:05 SYSA01   SYSD01   Socket initialization  *12002 192.168.55.50                           TCP Send/Recv Size 16384/16384
15:19:05 SYSA01   SYSD01   EE Link connected      *12002 192.168.55.50   12002 192.168.56.50   Buffer size 65535
15:19:06 SYSA01   SYSD01   EE Link initialized    *12002 192.168.55.50   12002 192.168.56.50   TLSV1 Cipher:09 Type:Client
15:19:06 SYSA01   SYSD01   EE Link Certificate    *12002 192.168.55.50   12002 192.168.56.50   Label:Default
15:19:08 SYSA01   SYSD01   Socket initialization  *12001 192.168.55.50                           UDP Send/Recv Size 65535/65535
15:19:08 SYSA01   SYSD01   Listening (UDP)         12001
15:19:08 SYSA01   SYSD01   Socket initialization  *12001 192.168.55.50                           TCP Send/Recv Size 16384/16384
15:19:08 SYSA01   SYSD01   EE Link connected      *12001 192.168.55.50   12001 192.168.56.50   Buffer size 65535
15:19:09 SYSA01   SYSD01   EE Link initialized    *12001 192.168.55.50   12001 192.168.56.50   TLSV1 Cipher:09 Type:Client
15:19:09 SYSA01   SYSD01   EE Link Certificate    *12001 192.168.55.50   12001 192.168.56.50   Label:Default
15:19:11 SYSA01   SYSD01   Socket initialization  *12000 192.168.55.50                           UDP Send/Recv Size 65535/65535
```

```
15:19:12 SYSA01   SYSD01   Listening (UDP)         12000
15:19:11 SYSA01   SYSD01   Socket initialization  *12000 192.168.55.50                          TCP Send/Recv Size 16384/16384
15:19:11 SYSA01   SYSD01   EE Link connected      *12000 192.168.55.50   12000 192.168.56.50   Buffer size 65535
15:19:13 SYSA01   SYSD01   EE Link initialized    *12000 192.168.55.50   12000 192.168.56.50   TLSV1 Cipher:09 Type:Client
15:19:13 SYSA01   SYSD01   EE Link Certificate    *12000 192.168.55.50   12000 192.168.56.50   Label:Default
```

CHAPTER 11

Appendix 2 APIAS PROC Clean-up Step

This chapter contains a suggested procedure that can be run prior to your APIAS activation step in the APIAS PROC to drop any of the EE ports which the IP stack has not properly terminated.

Overview

This appendix describes a procedure that can be used by APIAS installations to ensure that all the ports required for APIAS to initialize successfully are available. This procedure may be used in any APIAS installation, but especially those that do *not* use ephemeral ports (EPHEMERAL OFF is specified in the APIAS configuration - see page 46) and which are therefore dependent on the default ports being available at APIAS initialization time. The default ports are 12000-12004, plus 12005 if SSL mode is used.

The procedure ensures that there are no residual sessions (port listeners) from previous APIAS executions. It relies on an additional step being added prior to the APIAS initialization step. This additional step checks for any of the standard ports still having active APIAS listeners and uses a NETSTAT DROP command to close them down, thus ensuring that they are available when the APIAS initialization step runs.

RACF Considerations

IMPLEX requires enough RACF authority to issue the DROP commands for any residual sessions that it finds. You should therefore ensure that if you have defined a RACF resource ‘MVS.VARY.TCPIP.DROP’ in the OPERCMDS class, the user under which the new procedure runs is granted READ access to this resource.

Here is a complete example set of commands to define the classes and resources and grant the required authority to user *your_userid* to issue the commands:

SETROPTS CLASSACT(OPERCMDS)
SETROPTS RACLIST(OPERCMDS)

RDEFINE OPERCMDS (MVS.VARY.TCPIP.DROP) UACC(NONE)
PERMIT MVS.VARY.TCPIP.DROP ACCESS(READ) CLASS(OPERCMDS)
ID(*your_userid*)

SETROPTS RACLIST(OPERCMDS) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH

Changing the APIAS PROC

To run the procedure simply requires a new jobstep to be added at the start of your current APIAS PROC. This new step runs a REXX program (see “The Session Drop Program (APSDROP1)” on page 105) in a source library defined to the procedure when it is run. In this example the name ‘APSDROP1’ has been used as the member name that contains the REXX program.

```
//APIAS    PROC PROFILE=APIAS
//*****************************************************
//* Execute a batch TSO/REXX procedure to DROP any        *
//* residual sessions from a previous APIAS execution.    *
//*****************************************************
//TSOSTEP   EXEC PGM=IKJEFT01
//SYSEXEC   DD   DISP=SHR,DSN=yourhlq.SOURCE
//INTERDR   DD   SYSOUT=(B,INTRDR),DCB=LRECL=80
//SYSPRINT  DD   SYSOUT=*
//SYSTSPRT  DD   SYSOUT=*
//SYSTSIN   DD   *
%APSDROP1
//*****************************************************
//* Initialize Apias Version 3.11                         *
//*****************************************************
//APIAS    EXEC PGM=APIAS,REGION=4M,TIME=1440,PARM='&PROFILE'
//STEPLIB  DD   DISP=SHR,DSN=your-hlq.LOADLIB
//         DD   DISP=SHR,DSN=GSK.SGSKLOAD
//PARMLIB  DD   DSN=your_hlq.PARMLIB,DISP=SHR
//SYSOUT   DD   SYSOUT=*
//SYSAUDIT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSTRACE DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
```

The Session Drop Program (APSDROP1)

This is the source for the REXX program APSDROP1. This small program uses the NETSTAT SOCK command to list all the current ports (sessions) in use and then processes them issuing NETSTAT DROP commands for any which are for APIAS:

```
/* REXX-------------------------------------------------*/
/* Issue NETSTAT SOCKETS and DROP commands in batch        */
/* to ensure APIAS can initialize cleanly.                 */
/*------------------------------------------------------*/

/* trace r */

/* The Initial Processing Section lists all the ports and    */
/* writes them out to report file yourhlq.NETSTAT.SOCKETS.   */
/* This file is then read into the REXX stem variable CTL.   */
/* for the main processing section.                          */

   address TSO
      "NETSTAT SOCKETS REP HLQ yourhlq"

   acntrl = "yourhlq.NETSTAT.SOCKETS"

   address TSO
      "alloc f(CONTROL) da('"acntrl"') shr reuse"
      "execio * diskr CONTROL(finis stem CTL."
```

```
      "free f(CONTROL)"

/* The Main Processing Section processes the stem variable   */
/* CTL. searching for any APIAS ports. Ensure that you       */
/* change the name 'APIAS' in the code below to your actual  */
/* APIAS jobname. The program issues a NETSTAT DROP command  */
/* for any active APIAS ports that it finds.                 */

i = 0
m = 0

    Do i = 1 to CTL.0
        n = ' '
        linec = CTL.i
        parse var CTL.i 'Name: ' n 'Subtask:' rest .
        name = strip(substr(n,1,7))
        if name = "APIAS" then iterate

        m = i + 1

        Do while m > CTL.0
           linec = CTL.m
           if substr(linec,1,5) = "Name:" then LEAVE

           parse var CTL.m h1 t h2 s h3 c .
           m = m + 1
           hdr1 = strip(substr(h1,1,5))
           type = strip(substr(t,1,6))
           hdr3 = strip(substr(h3,1,5))
           conn = strip(substr(c,1,8))
           if hdr3 = "Conn:" then iterate

           address TSO
           "NETSTAT DROP "||conn
        End
    End

EXIT
```

This example REXX program uses the 'REP' option of the NETSTAT SOCKETS command so that a report of the detected sockets is produced, thus providing an audit trail of the processing. It is possible, however, to populate the stem variable CTL. directly using the OUTTRAP function. Should you want to use this alternative method, replace the complete Initial Processing Section in the program above with these alternative statements:

```
x = outtrap('CTL.')
    address TSO
      "NETSTAT CONN"
y = outtrap('OFF')
```

Alternative Session Drop Program (APSDROP2)

If you cannot, or do not wish to, use the TSO NETSTAT command, the same outcome can be achieved using the VARY TCPIP,,DROP command. This method is a little more complicated as it builds and submits a new job to the internal JES reader. Here is the REXX program APSDROP2 that uses this alternative methodology:

```
/* REXX-----------------------------------------------------*/
/* Issue NETSTAT SOCKETS and TCPIP VARY commands in batch    */
/* to ensure APIAS can initialize cleanly.                   */
/*----------------------------------------------------------*/

/* trace r */

/* The Initial Processing Section lists all the ports and    */
/* writes them out to report file yourhlq.NETSTAT.SOCKETS.   */
/* This file is then read into the REXX stem variable CTL.   */
/* for the main processing section.                          */

   address TSO
      "NETSTAT SOCK REP HLQ yourhlq"

   acntrl = "yourhlq.NETSTAT.SOCKETS"

   address TSO
      "alloc f(CONTROL) da('"acntrl"') shr reuse"
      "execio * diskr CONTROL(finis stem CTL."
      "free f(CONTROL)"

/* The Main Processing Section processes the stem variable   */
/* CTL. searching for any APIAS ports. Ensure that you       */
/* change the name ‘APIAS’ in the code below to your actual  */
/* APIAS jobname. The program issues VARY TCPIP DROP commands*/
/* for any active APIAS ports that it finds.                 */

i = 0
m = 0

    dropbuf
    queue "//APSDROP2 JOB"
    queue "//STEP EXEC PGM=IEFBR14"
    queue "//SYSPRINT  DD  SYSOUT=*"

    Do i = 1 to CTL.0
        n = ' '
        linec = CTL.i
        parse var CTL.i 'Name: ' n 'Subtask:' rest .
        name = strip(substr(n,1,7))
        if name = "APIAS" then iterate

        m = i + 1
```

```
        Do while m > CTL.0
           linec = CTL.m
           if substr(linec,1,5) = "Name:" then LEAVE

           parse var CTL.m h1 t h2 s h3 c .
           m = m + 1
           hdr1 = strip(substr(h1,1,5))
           type = strip(substr(t,1,6))
           hdr3 = strip(substr(h3,1,5))
           conn = strip(substr(c,1,8))
           if hdr3 = "Conn:" then iterate

           mcmd = "'"||"V TCPIP,,DROP,CONN="||conn"'"
           queue "// COMMAND "mcmd
        End
        "execio * diskw INTERDR (finis"
    End

    Dropbuf

EXIT
```

You may find two issues when using this method. The first is that the REXX-generated job needs to be run with enough authority to issue the VARY command. The second issue is that you may find that you need to suppress operator prompts when using this command in batch. To correct these issues the JES parameters may need adjusting.

The job that runs these instructions needs to have COMMAND=EXECUTE authority for the job (COMMAND=VERIFY is the default) to resolve both these issues. Since changing this requires a COLD start of JES, a temporary change can be made using a JES operator command, for example:

£T JOBCLASS(*),COMMAND=EXECUTE

After issuing this command the procedure should run without problems.

Index