

# RACF Remote Sharing Facility over TCP/IP

Implementing RRSF over TCP/IP with no APPC

Migrating from RRSF APPC to TCP/IP

Using detailed planning and implementation scenarios



Karan Singh  
Rama Ayyar  
Mike Onghena  
Phil Peters

Arunkumar Ramachandran  
Philippe Richard  
Roland Schwahn

**Redbooks**





International Technical Support Organization

**RACF Remote Sharing Facility over TCP/IP**

August 2012

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.

**First Edition (August 2012)**

This edition applies to Version 1, Release 13 of z/OS (product number 5694-A01).

**© Copyright International Business Machines Corporation 2012. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
 <b>Preface</b> .....	ix
The team who wrote this book .....	ix
Now you can become a published author, too! .....	x
Comments welcome .....	x
Stay connected to IBM Redbooks .....	xi
 <b>Chapter 1. Introduction</b> .....	1
1.1 RRSF network .....	2
1.1.1 RRSF nodes .....	2
1.1.2 RRSF modes of operation .....	2
1.1.3 Multi-system node and single-system node .....	3
1.2 User ID association .....	4
1.2.1 Peer user ID association .....	4
1.2.2 Managed user ID association .....	4
1.3 RRSF components .....	5
1.4 RRSF functions .....	6
1.4.1 Password synchronization .....	6
1.4.2 Command direction .....	6
1.4.3 Automatic password direction .....	6
1.4.4 Automatic command direct .....	7
1.5 How RRSF operates .....	7
1.6 RRSF over TCP/IP .....	8
1.7 Defining and controlling RRSF environment .....	8
1.7.1 SET command .....	8
1.7.2 TARGET command .....	9
1.7.3 RRSFDATA class .....	9
1.8 RRSF setup overview .....	10
 <b>Chapter 2. Planning</b> .....	13
2.1 Implementing RRSF using TCP/IP .....	14
2.1.1 Original configuration .....	14
2.1.2 Nodes and systems .....	14
2.1.3 RRSF configuration .....	15
2.1.4 RRSF functions .....	16
2.1.5 Comparing original environments .....	17
2.1.6 Workspace data sets .....	21
2.1.7 Network considerations .....	22
2.2 Converting from APPC to TCP/IP .....	23
2.2.1 Original APPC protocol configuration .....	24
2.2.2 Earlier systems .....	25
2.2.3 RACF parameter library IRROPTxx .....	25
2.2.4 Workspace data sets .....	25
2.2.5 Network considerations .....	26
 <b>Chapter 3. Configuring RRSF for TCP/IP</b> .....	27
3.1 Overview of AT-TLS .....	28

3.2	Setting up the PAGENT-started task . . . . .	28
3.2.1	Setting up the PAGENT started task procedure . . . . .	29
3.2.2	Defining the necessary RACF profiles for PAGENT . . . . .	30
3.2.3	Restricting access to the pasearch command . . . . .	31
3.2.4	Set up TTLS Stack Initialization access control . . . . .	31
3.2.5	Verifying the PAGENT setup . . . . .	32
3.3	Configuring AT-TLS policy for RRSF server and client . . . . .	32
3.3.1	Identifying the traffic that you need to protect . . . . .	33
3.3.2	Authenticating the remote RRSF node . . . . .	33
3.3.3	Encrypting the data across the TCP/IP connection . . . . .	34
3.3.4	Coding the policy . . . . .	34
3.4	Creating digital certificates and key rings . . . . .	46
3.4.1	Implementing an RRSF trust policy . . . . .	46
3.4.2	Digital certificates . . . . .	46
3.4.3	Using the same, self-signed certificate for all RRSF nodes . . . . .	48
3.4.4	Using an internal CA to sign a server certificate for each RRSF node . . . . .	50
3.4.5	Using an external CA to sign a server certificate for each RRSF node . . . . .	66
3.5	Updating TCP/IP to enable RRSF . . . . .	68
3.5.1	Enabling AT-TLS . . . . .	68
3.5.2	Defining and protecting RRSF port 18136 . . . . .	68
3.6	Updating necessary RACF profiles . . . . .	69
3.6.1	Adding an OMVS segment to RACF subsystem user ID . . . . .	69
3.6.2	Enabling CSFSERV resources . . . . .	70
3.6.3	Protecting the TCP/IP stack . . . . .	70
3.7	Verifying the RRSF setup . . . . .	71
3.7.1	Activating and verifying AT-TLS . . . . .	72
3.7.2	Stopping RACF on all systems . . . . .	74
3.7.3	Starting RACF on all systems without defining any RRSF nodes . . . . .	75
3.7.4	Defining the RRSF node on local main system SC75 in PLEX75 . . . . .	76
3.7.5	Defining the RRSF node on local system SC74 in PLEX75 . . . . .	77
3.7.6	Defining the RRSF node on local system SC76 in PLEX76 . . . . .	78
3.7.7	Defining the remote node to SC75 . . . . .	79
3.7.8	Defining the remote node to SC74 . . . . .	79
3.7.9	Defining the remote node to SC76 . . . . .	80
3.7.10	Displaying the RRSF status on every system . . . . .	81
<b>Chapter 4.</b>	<b>Converting APPC connections to TCP connections . . . . .</b>	<b>83</b>
4.1	Sample configuration . . . . .	84
4.2	Protocol conversion overview . . . . .	84
4.3	Protocol conversion process . . . . .	85
4.4	Sample configuration IRROPTxx files . . . . .	85
4.5	A simple protocol conversion . . . . .	87
4.6	Target command changes . . . . .	89
4.7	Protocol conversion and the local node . . . . .	92
4.8	Multi-System Node considerations . . . . .	92
4.9	Order of protocol definition . . . . .	92
4.10	Workspace data set considerations . . . . .	93
4.11	Information required before protocol conversion . . . . .	94
4.12	IRROPTxx considerations . . . . .	95
4.13	Conversion of the sample configuration . . . . .	95
4.13.1	Preliminary setup . . . . .	96
4.13.2	Updating local node definitions to support TCP in addition to APPC . . . . .	96
4.13.3	Converting communications between PLEX75 and PLEX76 from APPC to TCP . . . . .	97

4.14	Earlier system considerations . . . . .	100
4.14.1	Adding TCP to local node definition . . . . .	100
4.14.2	Converting remote nodes . . . . .	101
4.14.3	Execution of shared IRROPTxx file on current and earlier systems . . . . .	103
4.14.4	Conversion of communications between MSN PLEX75 and MSN PLEX81 . . . . .	104
<b>Chapter 5.</b>	<b>Operations . . . . .</b>	<b>107</b>
5.1	Administration . . . . .	108
5.1.1	SET command . . . . .	108
5.1.2	TARGET commands (nodes) . . . . .	110
5.1.3	IP addresses and ports . . . . .	111
5.1.4	PAGENT and Certificates: AT-TLS . . . . .	111
5.1.5	RESTART command . . . . .	112
5.2	RRSFDATA resources . . . . .	112
5.2.1	RRSFDATA RESOURCES related to AUTODIRECT . . . . .	112
5.2.2	CLAUTH . . . . .	113
5.2.3	FIELD level access . . . . .	114
5.2.4	RRSF Considerations for Automatic ID Assignment . . . . .	114
5.3	SETROPTS and SYSPLEX COMMUNICATION . . . . .	114
5.4	Revoked IDs or IDs revoked at different nodes . . . . .	115
5.5	Certificates and certificate renewal . . . . .	116
5.5.1	Renewing an expiring RRSF server certificate issued by a local CA . . . . .	117
5.5.2	Renewing an expired RRSF server certificate . . . . .	117
5.5.3	RRSF remote node server certificate expired . . . . .	122
5.5.4	Renewing (rekeying) a certificate with a new private key . . . . .	125
5.5.5	Considerations for CA certificate management . . . . .	127
5.5.6	Renewing a local CERTAUTH certificate . . . . .	128
5.5.7	Renewing an expired CERTAUTH certificate . . . . .	128
5.6	Diagnosing RRSF problems . . . . .	131
5.6.1	Error messages from components . . . . .	131
5.6.2	Obtaining information about RRSF connections . . . . .	133
5.6.3	z/OS UNIX System Services errors . . . . .	134
5.6.4	TCP/IP errors . . . . .	135
5.6.5	AT-TLS errors . . . . .	135
5.6.6	Secure Sockets Layer (SSL) . . . . .	136
5.6.7	RACF errors . . . . .	137
5.6.8	PAGENT errors . . . . .	137
5.6.9	RRSF errors . . . . .	138
5.6.10	Useful commands for problem diagnosis . . . . .	138
5.7	Maintenance and recovery: RRSF data sets . . . . .	139
5.7.1	RRSFLIST data sets . . . . .	139
5.7.2	Moving/resizing/renaming the RRSF workspace data sets . . . . .	140
5.7.3	INMSG and OUTMSG data set names . . . . .	142
	<b>Related publications . . . . .</b>	<b>143</b>
	IBM Redbooks . . . . .	143
	Other publications . . . . .	143
	Help from IBM . . . . .	143
	<b>Index . . . . .</b>	<b>145</b>





# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

CICS®	MVS™	System z®
DB2®	Parallel Sysplex®	WebSphere®
IBM®	RACF®	z/OS®
IMS™	Redbooks®	
Language Environment®	Redbooks (logo)  ®	

The following terms are trademarks of other companies:

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

# Preface

The IBM RACF® remote sharing facility (RRSF) allows RACF to communicate with other IBM z/OS® systems that use RACF, allowing you to maintain remote RACF databases. RRSF support for the security administrator provides these benefits:

- ▶ Administration of RACF databases from anywhere in the RRSF network
- ▶ Creation of User ID associations for password and password phrase synchronization
- ▶ Automatic synchronization of databases

Before to z/OS V1R13, RRSF only supported the APPC protocol. With z/OS release V1R13, TCP/IP can be used to extend the RACF Remote Sharing Facility (RRSF) functionality to a network of RRSF nodes capable of communicating over the TCP/IP protocol. Using TCP/IP connections for RRSF nodes provides advantages over APPC such as improved security, including stronger encryption levels.

This IBM® Redbooks® publication addresses the issue of implementing a new RRSF network using the TCP/IP protocol. It covers planning, implementation, and operational issues for deploying RRSF using TCP/IP. In addition, It addresses migration of an RRSF network from APPC to TCP/IP, including in-depth examples of the migration process.

## The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Karan Singh** is a Project Leader at the IBM ITSO GCS organization in Poughkeepsie, New York.

**Rama Ayyar** is an independent IT consultant based in Sydney, Australia. He was formerly with IBM Australia for 12 years. He has also worked for CSC Australia in senior technical roles. He is one of the founding members of HCL India. Rama has over 25 years of experience with the IBM MVS™ Operating System. His areas of expertise include TCP/IP, RACF, DFSMS, z/OS Operating System, HCD and Configuration Management, Dump Analysis, and Disaster Recovery. Rama has co-authored nine IBM Redbooks. He holds a Master's Degree in Computer Science from the Indian Institute of Technology, Kanpur. Rama has been in the computer industry for more than 35 years.

**Mike Onghena** is a Senior Software Engineer in the z/OS security area at IBM Poughkeepsie. He is a developer for RACF and has been with IBM for 20 years.

**Phil Peters** has over 20 years of experience with RACF and security on z/OS. He currently is a member of the IBM z/OS Integration Test team in Poughkeepsie, NY (USA). His responsibilities include the product support and pre-release testing of RACF and z/Secure. In his previous assignment with IBM Global Services, he performed RACF support and security audit compliance for numerous IBM internal and commercial accounts.

**Arunkumar Ramachandran** is a Senior IT Specialist with IBM India Software Lab. He has over 11 years of experience in the area of IBM System z® software systems programming. He holds a Master's Degree in Chemical Engineering from Indian Institute of Technology, Roorkee.

**Philippe Richard** works in the IBM EMEA Products and Solutions Support Center in Montpellier, France where he is involved in benchmarks and education. Philippe has a Master's Degree in Enterprise Management and Economy. He joined IBM France in 1985 to work in software support for MVS. Philippe has held a number of positions in this field, including teaching, systems programming, migration consultant, educational curriculum planning, and project planning. Philippe is also a developer of technical training and classes for STG System z where he is in charge of the z/OS curriculum content, including z/OS, IBM Parallel Sysplex®, UNIX System Services, and IBM WebSphere® for z/OS.

**Roland Schwahn** is an Advisory Remote Technical Support Professional in Canada. He has 20 years experience as an MVS Systems Programmer supporting Insurance, Banking, Manufacturing, and Retail/Wholesale. He has worked at IBM for over 16 years as a Remote Technical Specialist in Supportline. His areas of expertise include RACF, SMP/E, and the BCP. He has been involved in customer engagements regarding RACF setup and configuration. He holds a Bachelor's Degree in Science/Computer Science.

Thanks to the following people for their contributions to this project:

Robert Haimowitz  
Roy Costa  
International Technical Support Organization, Poughkeepsie Center

Neil Shah  
Bruce Wells  
IBM

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:  
[ibm.com/redbooks](http://ibm.com/redbooks)
- ▶ Send your comments in an email to:  
[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)
- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:  
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:  
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:  
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:  
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:  
<http://www.redbooks.ibm.com/rss.html>





# Introduction

This chapter addresses the basic concepts, components, and functions of the Resource Access Control Facility (RACF) remote sharing facility (RRSF). These concepts include the new TCP/IP feature available with z/OS V1R13. It also provides an overview of RRSF setup over TCP/IP.

RRSF is a RACF function that allows you to easily manage RACF databases across the enterprise by propagating RACF TSO commands, application updates, and user password changes. It is similar to a network of two or more nodes that communicate with each other to control updates to RACF database. RRSF determines which updates are propagated and to where.

RRSF provides the following functions:

- ▶ Administers RACF databases distributed throughout an enterprise from any location in the enterprise.
- ▶ Synchronizes RACF databases, which is helpful during disaster recovery scenarios.
- ▶ Allows users to synchronize their passwords. A user with more than one user ID on the same system, or different user IDs on different systems, can keep passwords synchronized between all user IDs. This feature benefits RACF users who work on several systems, or who use multiple user IDs on the same system.

This chapter includes the following sections:

- ▶ RRSF network
- ▶ User ID association
- ▶ RRSF components
- ▶ RRSF functions
- ▶ How RRSF operates
- ▶ RRSF over TCP/IP
- ▶ Defining and controlling RRSF environment
- ▶ RRSF setup overview

## 1.1 RRSF network

An RRSF network is a collection of RRSF nodes that use TCP/IP or APPC/MVS as the network transport mechanism. RRSF over TCP/IP is available only with z/OS V1R13 and later versions. An RRSF node consists of one or more z/OS systems that shares a RACF database with RRSF enabled and defined as an RRSF node to RACF. This configuration is shown in Figure 1-1.

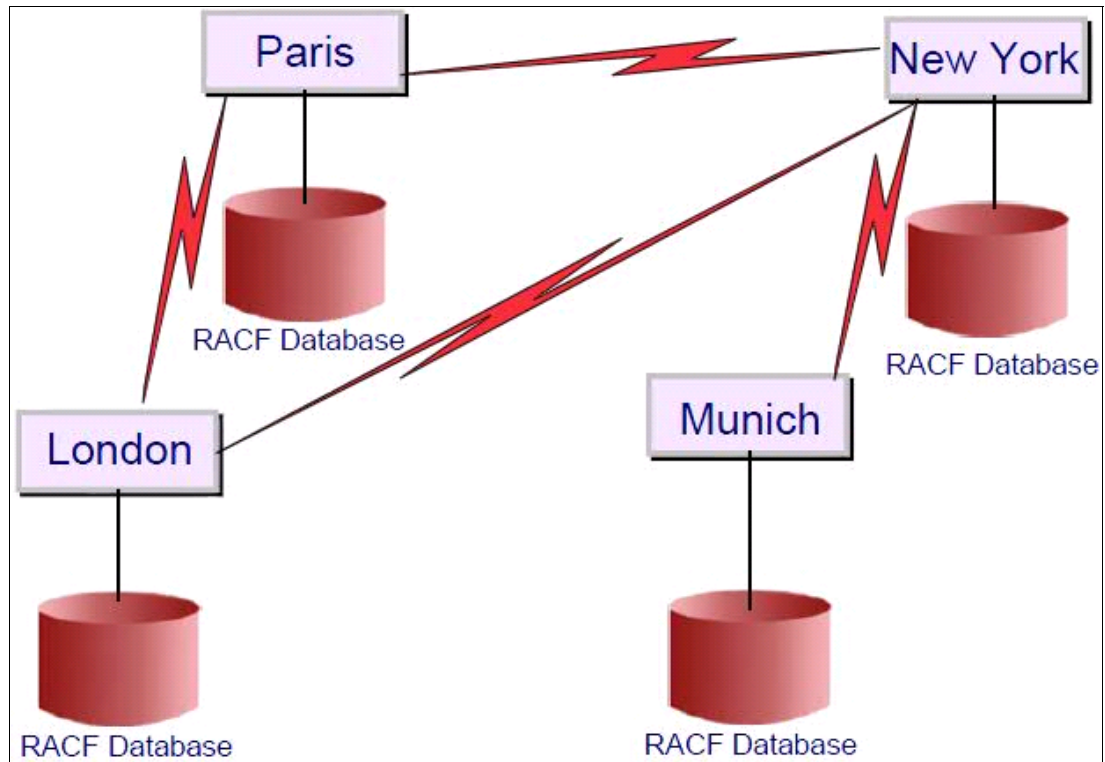


Figure 1-1 RRSF network

### 1.1.1 RRSF nodes

RRSF nodes are local nodes or remote nodes, depending on where the originating event (password change, command, or application update) occurs. If you log on to the RRSF node Paris, that is the local node and the other RRSF nodes are the remote nodes. You can use **TARGET**, a RACF command, to define RRSF nodes to RACF.

### 1.1.2 RRSF modes of operation

RRSF nodes have two modes of operation: Local and remote. An RRSF node that has no RRSF connection with any other remote node operates in local mode. A node that is operating in local mode cannot communicate with other nodes. RRSF cannot send or receive remote RRSF requests.

Any RRSF node correctly configured to communicate with any other remote node operates in remote mode. When a user ID issues a command from Paris node to be run at the remote node, New York, the Paris node is operating in remote mode.



### 1.1.3 Multi-system node and single-system node

Nodes can also be single-system (SSN) or multi-system (MSN). An MSN node consists of several z/OS systems that share a RACF database. Each system is defined to RACF as an RRSF node, and one system is identified as the MAIN system.

An SSN node consists of only one system. Figure 1-2 shows an RRSF network of two MSNs and a one SSN.

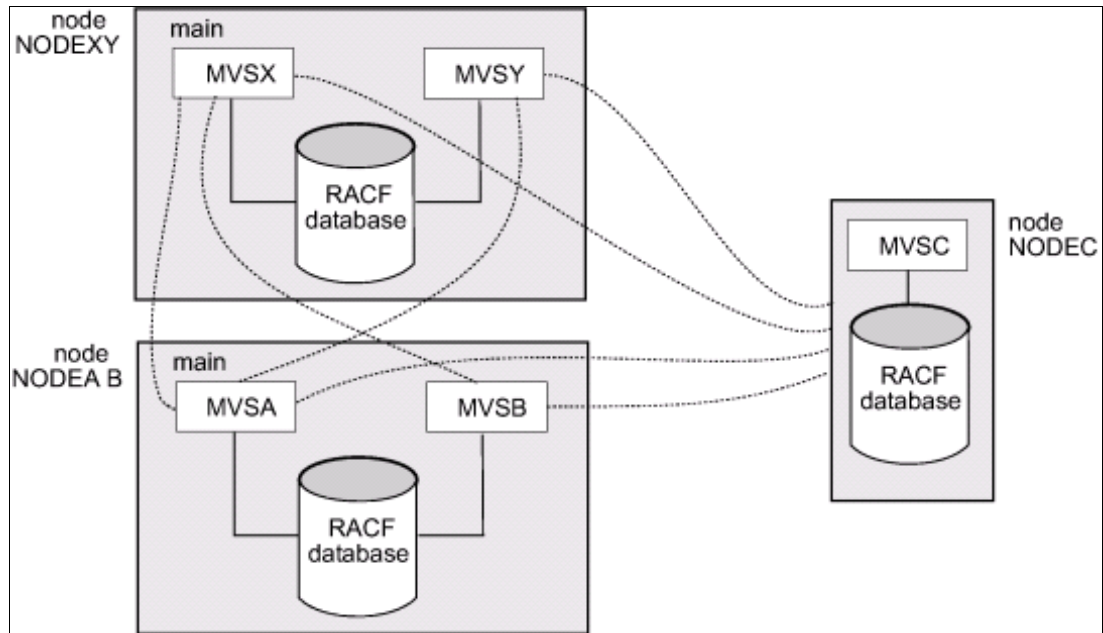


Figure 1-2 Multi-system node (MSN) and single system node (SSN)

MSN and SSN have the following aspects:

- ▶ All SSNs in an RRSF network send requests only to the main system of an MSN. In Figure 1-2, SSN NODEC sends requests only to the main system, MVSX of MSN NODEXY, and main system, MVSA of MSN NODEAB.
- ▶ All peer systems in an MSN send requests only to SSNs and to the main systems of remote MSNs. In Figure 1-2, all peer systems in MSN NODEXY send requests only to SSN, NODEC, and to the main system MVSA of remote MSN NODEAB.
- ▶ Peer systems in an MSN do not communicate with each other, and do not communicate with non-main systems of remote MSNs. In the Figure 1-2, system MVSY in MSN NODEXY does not communicate with peer system MVSX in the same MSN. It also does not communicate with system MVSAB of remote MSN NODEAB.

## 1.2 User ID association

A user ID association is required before you can use certain RRSF functions such as command direction and password synchronization. A user ID association is an agreement between two user IDs. One user ID defines the association, and the other user ID approves or rejects it. Use the **RACLINK** command to define the association. Figure 1-3 shows this user ID association.

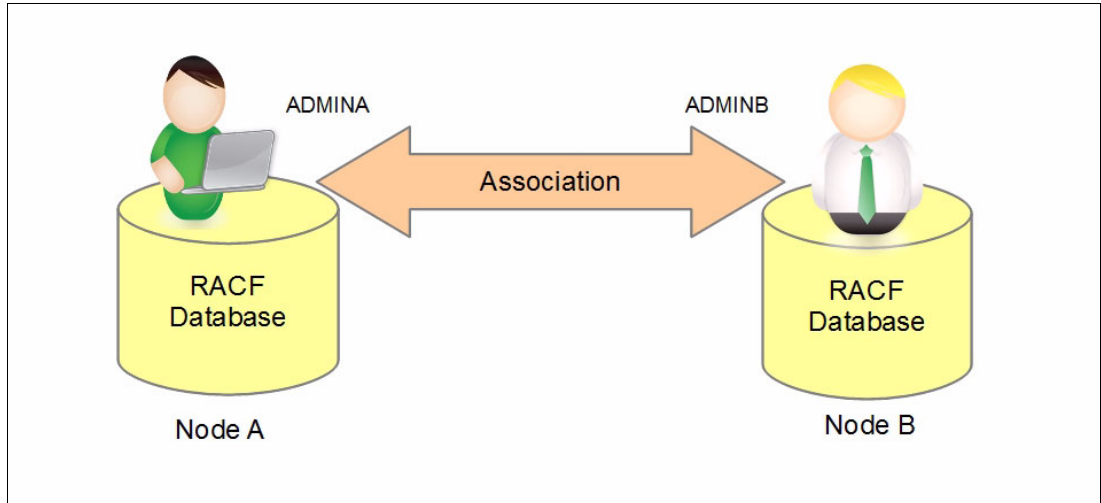


Figure 1-3 User ID association

The **RACLINK** command allows RACF users to specify password synchronization between the associated user IDs. The RACF user profiles associated with each of the involved user IDs are updated by RRSF to point to the other user ID. RRSF uses this information to authorize the users requests for command direction and password synchronization. There are two types of user ID association: Peer and managed.

### 1.2.1 Peer user ID association

Peer association allows RACF users, under either user ID in the association, to run allowed RACF TSO commands under the authority of the other user ID with command direction. They can run commands whether they are on the same node or on a remote node. Peer association also allows password synchronization between these user IDs. You might use this association for a security administrator with two different user IDs defined on two different systems. This configuration allows the administrator to manage the RACF database of the remote system while being logged on to the local system.

### 1.2.2 Managed user ID association

In a managed association, one of the user IDs is designated as the managing ID, and the other is designated as the managed ID. The managing user ID can direct commands to the managed ID, but the managed ID cannot direct commands to the managing ID. The user IDs in a managed association cannot synchronize their passwords. If ADMINA user ID of Node A manages association with ADMINB user ID of Node B, ADMINA is the managing user ID and ADMINB is the managed user ID.

## 1.3 RRSF components

Each RRSF node contains the following components:

- ▶ RACF subsystem address space is an important component that coordinates the activities of RRSF. RACF masks the command before sending it to another node over a network. It then unmaskes the command received from another node for further processing.
- ▶ TCP/IP can be used to communicate between RRSF nodes of z/OS V1R13 and later.
- ▶ RACF uses Workspace data sets, which are VSAM data sets to temporarily hold data that RACF is sending or receiving from one node to another node. RACF deletes data from the workspace data sets after successful processing. Workspace data sets have these variations:
  - OUTMSG data sets. RACF temporarily saves a copy of the masked command in OUTMSG data set of the local node before sending to a remote node. After the remote nodes acknowledge the receipt of the command, RACF deletes the command from OUTMSG data set of the local node.
  - INMSG data sets. RACF temporarily saves a copy of the received masked command from other remote nodes. Later, RACF unmaskes the command and runs under appropriate authority. After running the command, RACF deletes the masked command saved in INMSG data set.
  - RRSFLIST data set is a repository in the local node for holding the command outputs run at remote nodes. The user that sent a command to a remote node is notified when the command is run at the remote node. Any command output flows back from the remote node and is placed in the RRSFLIST data set of the command issuer.

Figure 1-4 shows two RRSF nodes, each node has RACF address space, TCP/IP, RACF database and workspace data set, OUTMSG, INMSG, and RRSFLIST.

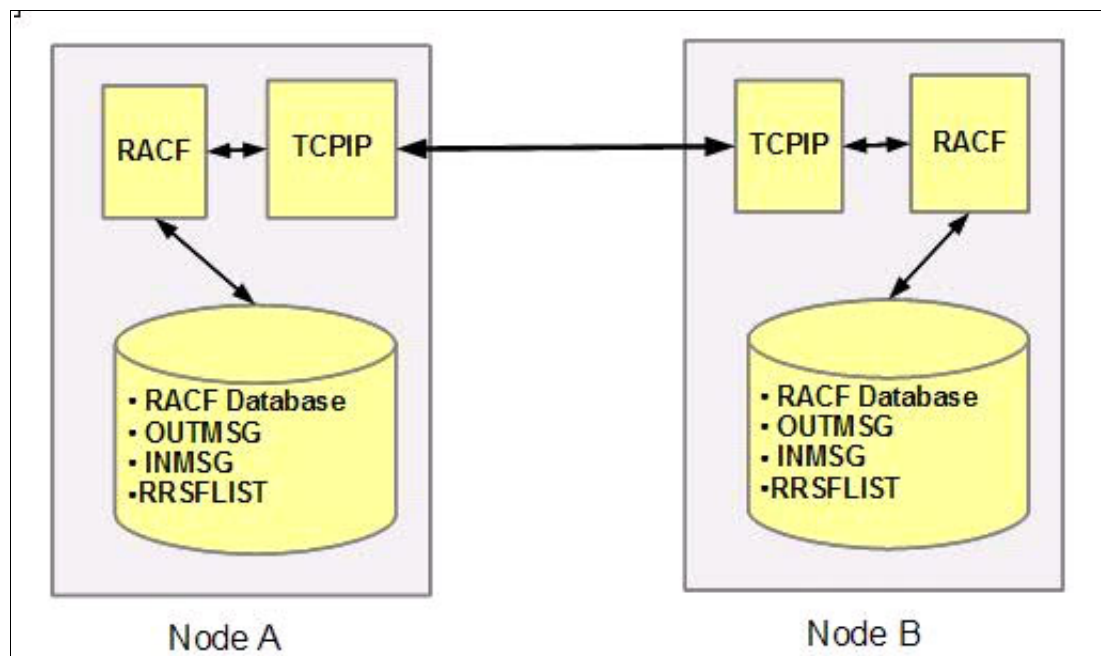


Figure 1-4 RRSF Components

## 1.4 RRSF functions

RRSF includes the following main functions:

- ▶ Password synchronization
- ▶ Command direction
- ▶ Automatic password direction
- ▶ Automatic command direction

### 1.4.1 Password synchronization

RRSF provides password synchronization between two or more user IDs defined on the same node or on different nodes. You need to establish association between the involved user IDs to use this function. After peer association is established and password synchronization is specified, a password change for a RACF user on Node A is propagated to all the user IDs associated with that user ID. If a RACF user owns three user IDs, A1 and A2 on Node A, and B1 on Node B, the user needs three peer associations to propagate the password change. This requirement exists regardless of the user ID that is used to log on.

Before the password change is propagated, RACF checks to see whether the user ID that initiated the change has the authority to run password synchronization. The **RRSFDATA** RACF class controls this authorization.

### 1.4.2 Command direction

Command direction allows you to maintain remote RACF databases by using the **AT** keyword on specific RACF TSO commands. After a user ID association is established, RACF users can use the **AT** keyword to direct specific RACF TSO commands. These commands run in the RACF subsystem address space of the specified target node under the authority of the specified target user ID. Output produced by the command is captured by RRSF and written to the RRSFLIST user data set of the user ID that issued the command.

**Tip:** The naming convention for RRSFLIST data set is USERID.RRSFLIST. If ADMINA is the user ID, the RRSF data set name is ADMINA.RRSFLIST. For each user ID, RACF creates such a data set or uses a pre-allocated data set, if it exists.

Before RRSF sends the command to the target node, RACF checks whether the command issuer is authorized to run command direction for the specified target node. This process uses the **RRSFDATA** RACF class.

### 1.4.3 Automatic password direction

Automatic password direction is used to synchronize password changes run by the same user ID defined on different nodes. It does not require a user ID association between the involved user IDs, but the same user ID must exist on the concerned nodes. Any RACF user who has the same user ID defined on several nodes can take advantage of this function. In fact, it allows these users to have their passwords for these user IDs synchronized. They do not need to log on to each node to update the password every time it is changed as RRSF updates it automatically.

To take advantage of this function, each system must work in an automatic command direction environment. Use the **SET AUTODIRECT** command to establish this configuration. The **RRSFDATA** RACF class controls user eligibility for automatic password direction.

Automatic password direction can work together with password synchronization. This configuration allows RACF users to keep their password synchronized between the same user ID on different nodes. The password also synchronizes between user IDs that have established a peer user ID association. Set up the user ID association only when the user IDs are not the same.

When using automatic password direction, the notification of the password change and the command output are always sent to the user ID that initiated the password change.

## 1.4.4 Automatic command direct

Automatic command direction can automatically synchronize two or more RACF databases or applications in different RACF databases. With automatic command direct, specific RACF TSO commands or all RACF TSO commands that update the RACF database are automatically directed to run on a remote node. This process occurs after being run on the local node. Automatic password direction is a type of automatic command direction. Automatic command direction can work together with command direction because they are not mutually exclusive.

You need to establish an automatic command direction environment by using the **SET AUTODIRECT** command. A user ID association is not required, but the same user ID must exist on both systems. RACF runs authorization checking by using the **RRSFDATA** RACF class.

You can also specify which user IDs receive the notification message and command output of the RACF TSO command execution.

## 1.5 How RRSF operates

Figure 1-5 shows how the command issued by a user ID from RRSF node (Node A) to remote RRSF node (Node B) flows. It also shows how the output is returned to the user ID that issued the command.

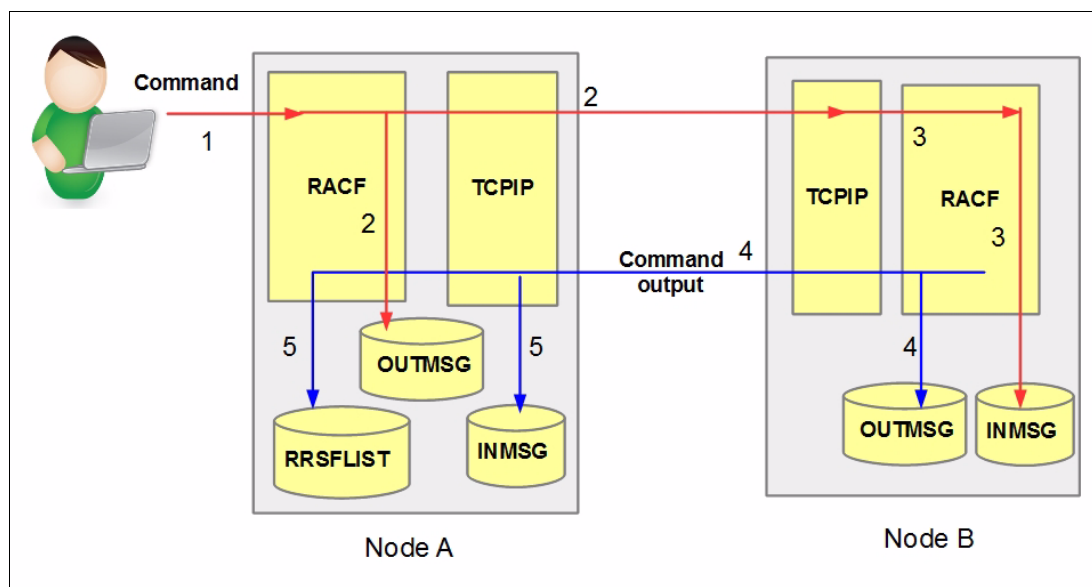


Figure 1-5 How RRSF operates

The command process includes the following steps:

1. User ID on Node A issues a command to be run at RRSF Node B.
2. RACF in Node A determines that the command is for remote Node B. It masks the command, saves a copy of the masked command in OUTMSG data set, and sends the masked command to Node B over TCP/IP. If there are problems with TCP/IP, the request is queued and sent when the TCP/IP connection is reestablished. The masked command in OUTMSG data set is retained until an acknowledgement is received from Node B.
3. In Node B, RACF receives the masked command from TCP/IP and saves a copy of the masked command in its INMSG workspace data set. It also sends an acknowledgement the receipt of the command to Node A.
4. In Node B, RACF unmaskes and runs the command. After running the command, RACF deletes the saved copy of the masked command in its INMSG data set. RACF in Node B masks the command output and sends it back to Node A, keeping a copy of the masked output in its OUTMSG data set.
5. In Node A, RACF receives the masked command output from TCP/IP and saves a copy of the masked command output in its INMSG data set. It then acknowledges to Node B that the message has been received.
6. RACF in Node B deletes the masked command output saved in its OUTMSG data set.
7. RACF in Node A unmaskes the command output and writes the output to the RRSFLIST of the user ID that issued the command. It then deletes the masked command output saved in its INMSG data set. Finally, RACF notifies the user ID that issued the command.

## 1.6 RRSF over TCP/IP

RRSF supports TCP/IP (IPV4 only) as an alternative transport protocol to APPC/MVS. This protocol can be used only between z/OS V1R13 systems. In an RRSF network that has a mix of z/OS V1R13, z/OS V1R12, and z/OS V1R11, TCP/IP can be used between z/OS V1R13 nodes. APPC/MVS must be used between the following nodes:

- ▶ z/OS V1R13 nodes and z/OS V1R12 nodes
- ▶ z/OS V1R13 nodes and z/OS V1R11 nodes
- ▶ z/OS V1R12 nodes and z/OS V1R11 nodes.

Using TCP/IP connections for RRSF nodes instead of APPC provides benefits such as improved overall security, including the availability of stronger encryption levels.

## 1.7 Defining and controlling RRSF environment

Administering RRSF is primarily accomplished through the use of the SET and TARGET commands, and the RRSFDATA class.

### 1.7.1 SET command

SET is one of the key RACF commands used to control the operational characteristics of RRSF. It is a RACF operator command that can be issued from a z/OS console. It can also be placed in RACF parameter libraries, IRROPTxx, to run during startup of RACF address space. For more information about RACF parameter libraries, see 2.2.3, “RACF parameter library IRROPTxx” on page 25.

The SET command can run the following functions:

- ▶ Specify and process IRROPTxx members of the RACF parameter library during RACF address space startup or dynamically.
- ▶ Specify tracing options for any diagnostic purposes.
- ▶ Display attributes of local RRSF node.
- ▶ Optionally specify the name of the local JESNODE to send any command output to the user ID. This process uses TSO transmit when RRSFLIST data set of the concerned user ID is full.

Because the SET command is an operator command, it has these characteristics:

- ▶ Can be issued only on the local node, and cannot be sent to remote node by using command direction or automatic command direction.
- ▶ Requires access to **OPERCMDS** RACF class apart from **RRSFDATA** class.

For more information about SET command, see section 5.55 in *z/OS V1R13.0 Security Server RACF Command Language Reference*, SA22-7687-16.

## 1.7.2 TARGET command

The TARGET command is another key RACF command used to define local RRSF node and remote RRSF nodes. It is also a RACF operator command that can be issued from the z/OS console. This command can be placed in the RACF parameter library, IRROPTxx, to be run during startup of the RACF address space.

TARGET commands must be issued on each member of an MSN or SSN node to define the nodes that are part of RRSF network. There must be one TARGET command for each remote node. The protocol to communicate with it is also specified, and one TARGET command for the local node. The TARGET command also describes the workspace data sets for each node.

Because the TARGET command is an operator command, it has these characteristics:

- ▶ Can be issued only on the local node and cannot be sent to remote node by using command direction or automatic command direction.
- ▶ Requires access to **OPERCMDS** RACF class apart from **RRSFDATA** class.

For more information about SET command, see 5.59 in *z/OS V1R13.0 Security Server RACF Command Language Reference*, SA22-7687-16.

## 1.7.3 RRSFDATA class

Profiles in the RRSFDATA class determine which remote sharing functions are available on a node and which users have access to them. This class must be active on an RRSF node before you can use many RRSF functions. These functions include defining associations, synchronizing passwords, directing commands with the AT keyword, and automatic direction. The RRSFDATA class can be used as a switch to turn these remote sharing functions on and off as you activate and deactivate the class.

Initially, the RRSFDATA class is not active, and no profiles are defined in the class. The RRSF functions controlled by RRSFDATA class are not available to any users. You must define profiles for the functions you want to use, and activate the RRSFDATA class to make the functions available.

For more information about RRSFDATA class, see 7.21, Controlling the Use of Remote Sharing Functions in *z/OS V1R13.0 Security Server RACF Security Administrator's Guide*, SA22-7683-15.

## 1.8 RRSF setup overview

RRSF (over TCP/IP) uses several z/OS system components as shown in Figure 1-6.

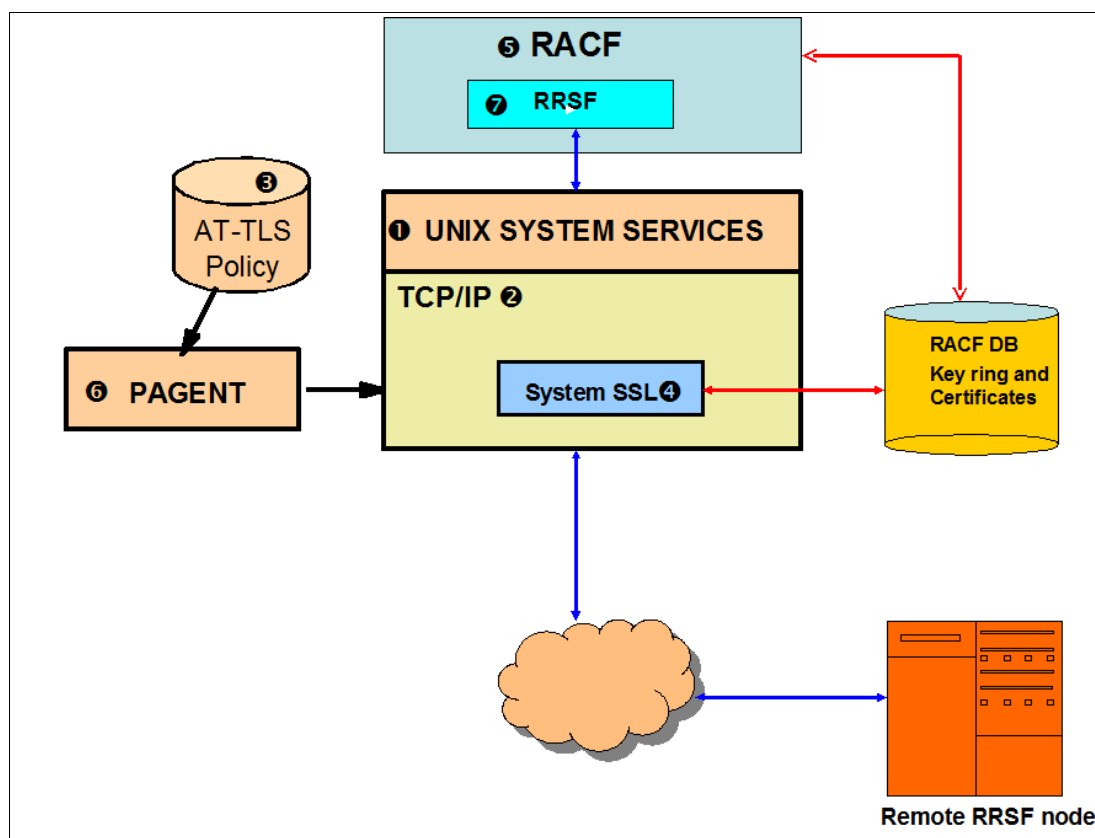


Figure 1-6 Components used by RRSF

- ▶ **1** z/OS UNIX System Services: RRSF makes UNIX System Services socket API calls to TCP/IP.
- ▶ **2** TCP/IP: The communication vehicle for exchanging data between the RRSF nodes.
- ▶ **3** AT-TLS: The traffic between the RRSF nodes is protected by AT-TLS.
- ▶ **4** Secure Sockets Layer (SSL): AT-TLS uses SSL to authenticate the RRSF nodes.
- ▶ **5** RACF: The resources used for RRSF are protected by RACF profiles.
- ▶ **6** Policy Agent (PAGENT): The AT-TLS policy is enforced by the PAGENT.
- ▶ **7** RRSF itself

Setting up RRSF over TCP/IP consists of the following steps:

1. Enable the RACF component of the z/OS Security Server.
2. Activate the RACF subsystem address space.



3. Requires the use of sockets, which requires UNIX System Services, to use TCP protocol.  
Add the following objects:
  - a. OMVS segment and UID to RACF subsystem user ID
  - b. OMVS segment and GID to the default group of RACF subsystem user ID
4. Deploy digital certificates and key rings that are used to authenticate RRSF servers to each other using TLS protocol.
5. Enable AT-TLS policy required for RRSF connections.
6. Permit the RACF subsystem user ID to the necessary resources.
7. Use **RACF TARGET** command to complete the following steps:
  - a. Establish a socket listener on each RRSF node.
  - b. Activate the connection from both the nodes for each connection.
  - c. Harden the **TARGET** commands in RACF parameter library for automatic activation during system start.





# Planning

This chapter describes the planning for RACF remote sharing facility (RRSF). Both a first time implementation that uses TCP/IP and a conversion from RRSF using Advanced Program-to-Program Communication (APPC) to RRSF using TCP/IP are addressed. The chapter uses an example environment to illustrate the planning process. This example includes the original configuration, the wanted configuration, prerequisites to implementation, and items to consider.

This book covers only the items most important in planning an RRSF network. There are other aspects not covered. For more information about these aspects, see the applicable sections of *z/OS V1R13.0 Security Server RACF System Programmer's Guide* and *z/OS V1R13.0 Security Server RACF Security Administrator's Guide*.

This chapter includes the following sections:

- ▶ Implementing RRSF using TCP/IP
- ▶ Converting from APPC to TCP/IP

## 2.1 Implementing RRSF using TCP/IP

This section describes the planning done to implement an RRSF network in the example environment for the first time using TCP/IP as the protocol.

### 2.1.1 Original configuration

The original configuration is shown in Figure 2-1. It consists of a group of two systems (SC74 and SC75) that share a RACF database. It also includes another group of two systems (SC80 and SC81) that share a RACF database and a single system image (SC76) that has its own RACF database. These systems make up the RRSF Network.

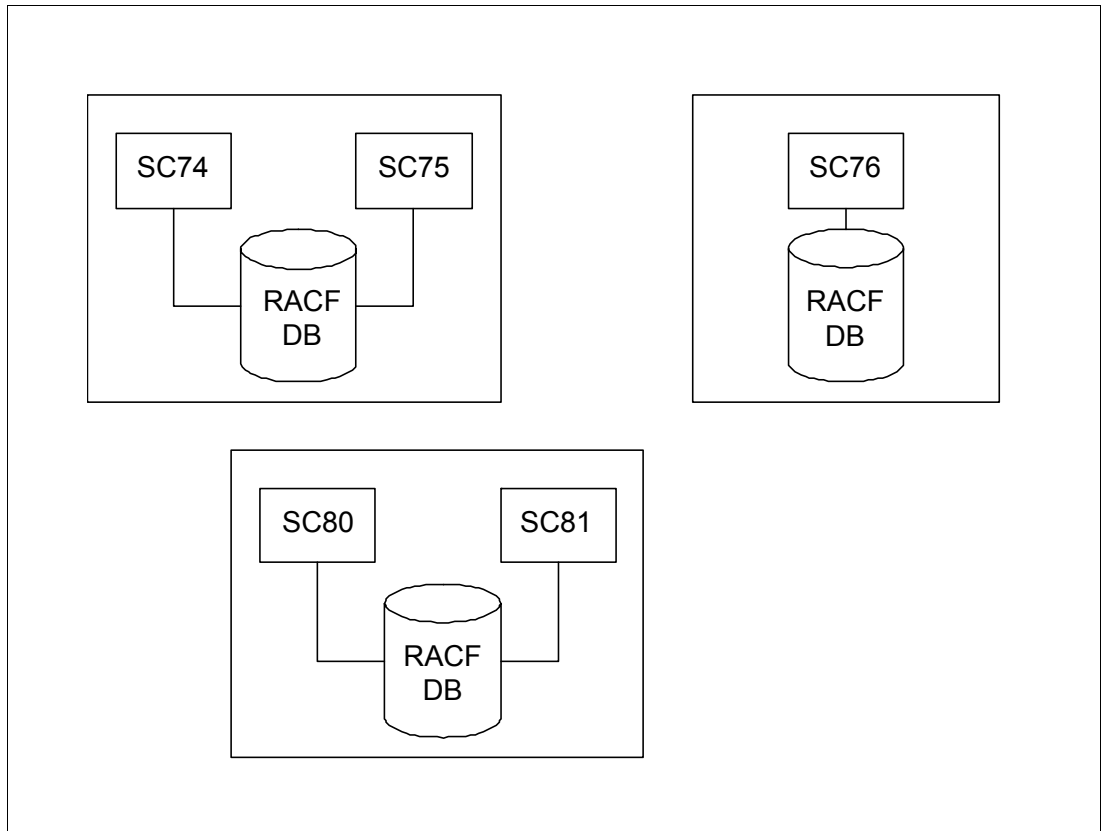


Figure 2-1 The original example environment.

### 2.1.2 Nodes and systems

RRSF uses the term “node” to refer to a group of system images that share a RACF database or to a single system image. The node names are defined to RRSF by using TARGET commands, and can have values of the implementor’s choosing. Because the setup of an RRSF network can be complex, choosing meaningful node names can be beneficial. PLEX75 (made up of systems SC74 and SC75), PLEX76 (a single system image), and PLEX81 (made up of systems SC80 and SC81) were chosen for the RRSF node names. These names were chosen as the JES node names and sysplex names in the new environment.

The systems that make up a multi-system node (MSN) are also defined to RRSF by using TARGET commands. Unlike nodes, systems cannot be given arbitrary names. The values used to identify the systems to RRSF must be the same as the values listed as SYSNAME in

the communications vector table (CVT). The CVT is found in IEASYMxx, IEASYSxx, or LOADxx. Example 2-1 shows the system names as found in our PLEX75's CVT.

*Example 2-1 Excerpt from SYS1.PARMLIB(IEASYMXX) on PLEX75.*

---

```

SYSDEF      HNAME(SCZP301)
             LPARNAME(A05)
             SYSNAME(SC74)
             SYSPARM(00)
             SYMDEF(&SYSLEVEL='ZOSR1D')
             SYMDEF(&IFAPRDXX='1A')
             SYMDEF(&LPALIST1='3A')
             SYMDEF(&OMVSPARM='3A')
             SYMDEF(&PROGX2='3A')
             SYMDEF(&PRISUBSY='JES2')
             SYMDEF(&SECSUBSY='JESA')
SYSDEF      HNAME(SCZP201)
             LPARNAME(A2C)
             SYSNAME(SC75)
             SYSPARM(00)
             SYMDEF(&SYSLEVEL='ZOSR1D')
             SYMDEF(&IFAPRDXX='1A')
             SYMDEF(&LPALIST1='3A')
             SYMDEF(&OMVSPARM='3A')
             SYMDEF(&PROGX2='3A')
             SYMDEF(&PRISUBSY='JES3')
             SYMDEF(&SECSUBSY='JES2')

```

---

For each MSN, a main system must be defined to RRSF. The main system is the one that receives most of the communications sent to the node. Give careful consideration when choosing your main system, because this system might have a considerable amount of network traffic and requests to process. Also, after the RRSF network is established, changing the main within a node is difficult. Generally, do not change the main after RRSF is in place.

For the two MSNs in the example environment, the amount of traffic or number of requests to process were not issues when choosing main systems. Instead, the main systems were chosen based on the RRSF node names. For PLEX75, SC75 was chosen as the main system. For PLEX81, SC81 was chosen as the main. Keeping the node and main names consistent helps remember which systems are the mains when setting up and working with the RRSF network.

Appendix A of *Security Server RACF System Programmer's Guide* provides a node configuration worksheet to help gather information needed to set up your network. Print a copy of this worksheet for each system and fill it in with information from your environment. This documentation helps keep track of the attributes of the systems. The more systems you have, the more complex the environment will be. Therefore it is valuable to have all of this information documented and readily accessible on a completed worksheet.

### 2.1.3 RRSF configuration

Figure 2-2 on page 16 illustrates the RRSF Network configuration. The lines between systems indicate that RRSF communication exists between those systems. Although communication occurs in both directions, the arrows on the lines show the direction that

commands and automated updates flow. Communication with the single system image occurs at the node level. All of the MSN systems communicate with the remote main system and with the single system node. The non-main systems do not communicate with remote non-main systems.

When planning for RRSF, create a similar diagram for your RRSF network. The diagram helps you visualize and track the connections between nodes. You might find it useful to refer to it often.

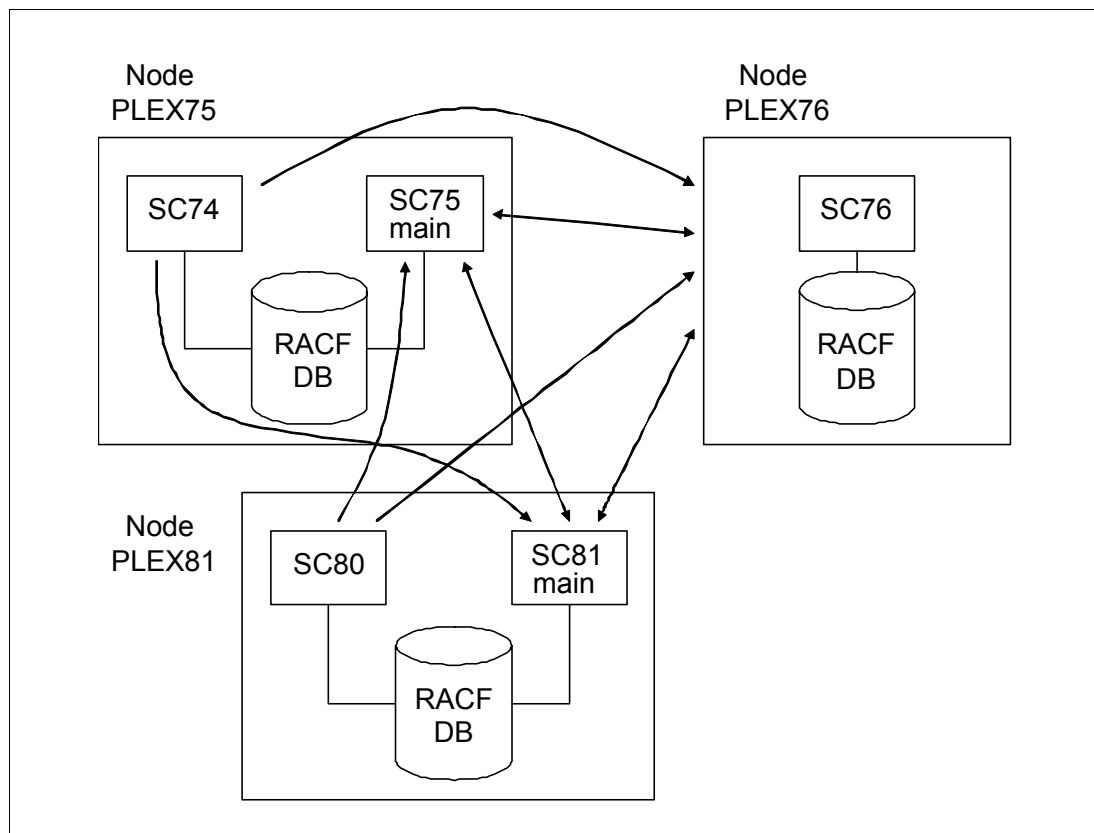


Figure 2-2 The example RRSF Network.

## 2.1.4 RRSF functions

Determine which functions of RRSF you would like to use in your RRSF environment. RRSF includes the following functions available:

- ▶ Command direction: RACF commands issued on your local system can be directed to run on your local or remote nodes.
- ▶ Automatic command direction: RACF commands issued on your local system can automatically be propagated to remote nodes.
- ▶ Automatic password direction: Password changes on a local system can automatically be propagated to remote nodes.
- ▶ Automatic direction of application updates: Updates made by RACF macros on your local system can automatically be propagated to remote nodes.
- ▶ User ID associations: Two user IDs on the same or different nodes can be linked for password synchronization and command direction.

You also need to determine which nodes, classes, commands, and user IDs will be involved with the various functions. You control these functions through resource profiles in the RRSFDATA class, and can activate/deactivate them with SET commands.

## 2.1.5 Comparing original environments

Before multiple RACF environments begin sharing, check that they are compatible and capable of supporting RRSF using TCP/IP.

### RACF level

For RRSF to communicate through the TCP/IP protocol, the nodes involved must be running RACF on z/OS V1R13. It is possible to communicate to nodes that run earlier versions of RACF by using the APPC protocol. Generally, however, upgrade any earlier nodes to V1R13 before implementing an RRSF Network.

### RACF address space

RRSF uses the RACF subsystem address space. On each RRSF system, ensure that the RACF address space is running before trying to implement RRSF. Also note the location of the subsystem address space procedure, which is typically SYS1.PROCLIB(RACF).

For the RACF subsystem address space procedure to automatically start at IPL time, an entry must exist for it in the IEFSSNxx member of your system PARMLIB. The entry must match the name of your RACF procedure. Example 2-2 shows a portion of the SYS1.PARMLIB(IEFSSN00) related to the example RACF subsystem. SUBNAME(RACF) <sup>1</sup> indicates that the name of the RACF subsystem is “RACF”. INITPARM(‘#,’M’) <sup>2</sup> indicates that the command prefix facility for RACF is “#”. When issuing commands through an operator console, the prefix “#” indicates that it is a RACF command and should be run by the RACF subsystem address space. The “M” indicates that the prefix is registered at the system level as opposed to the entire sysplex, which is the suggested setting.

*Example 2-2 SYS1.PARMLIB(IEFSSN00)*

---

```
SUBSYS SUBNAME(RACF) 1 /* RACF SUBSYS */  
INITRTN(IRRSSI00)  
INITPARM(' #,M') 2
```

---

Set up your RACF procedure so that it automatically processes a member of the RACF parameter library when the address space starts. Example 2-3 shows the RACF address space procedure.

*Example 2-3 'SYS1.PROCLIB(RACF)'*

---

```
//RACF PROC  
//RACF EXEC PGM=IRRSSM00,REGION=2M,PARM='OPT=00' 1  
//RACFPARM DD DISP=SHR,DSN=RRSF.JCL 2
```

---

PARM='OPT=00' <sup>1</sup> and DSN=RRSF.JCL <sup>2</sup> indicate that RRSF.JCL(IRROPT00) is the RACF parameter library member run when the RACF address space starts. The RACF parameter library is addressed in “RACF parameter library IRROPTxx” on page 19.

On each node, check that a profile in the STARTED class exists for the RACF address space and has an appropriate STDATA segment. TCP/IP is considered a UNIX Systems Services function. Therefore, the user ID and group associated with the STARTED profile in the STDATA segment must have a UID and GID.

Generally, make the started task TRUSTED and the associated user ID PROTECTED. Also, give the associated user ID a non-zero UID, and a GID to all of the groups to which the user ID is connected.

Example 2-4 shows excerpts taken from listing our RACF address space STARTED profile, associated user ID, and associated group.

*Example 2-4 RACF listings of the profiles associated with the example address space*

---

```

RLIST STARTED RACF.* STDATA
CLASS      NAME
-----
STARTED    RACF.* (G)
STDATA INFORMATION
-----
USER= RACF
GROUP= STCGROUP
TRUSTED= YES
PRIVILEGED= NO
TRACE= YES
READY
  LU RACF OMVS
USER=RACF NAME=UNKNOWN OWNER=SYSPROG   CREATED=99.287
DEFAULT-GROUP=STCGROUP PASSDATE=N/A    PASS-INTERVAL=N/A PHRASEDATE=N/A
ATTRIBUTES=PROTECTED
REVOKE DATE=NONE   RESUME DATE=NONE
LAST-ACCESS=12.135/16:53:00
CLASS AUTHORIZATIONS=NONE
NO-INSTALLATION-DATA
NO-MODEL-NAME
LOGON ALLOWED      (DAYS)          (TIME)
-----
ANYDAY              ANYTIME
GROUP=STCGROUP AUTH=USE    CONNECT-OWNER=SYSPROG   CONNECT-DATE=99.287
CONNECTS= 595 UACC=NONE   LAST-CONNECT=12.132/18:12:04
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE   RESUME DATE=NONE
OMVS INFORMATION
-----
UID= 0000018136
HOME= /
CPUTIMEMAX= NONE
ASSIZEMAX= NONE
FILEPROCMAx= NONE
PROCUSERMAX= NONE
THREADSMAX= NONE
MMAPAREAMAX= NONE
  LG STCGROUP OMVS
INFORMATION FOR GROUP STCGROUP
  SUPERIOR GROUP=SYS1      OWNER=SYSPROG   CREATED=99.287
  NO INSTALLATION DATA
  NO MODEL DATA SET
  TERMUACC
  NO SUBGROUPS
  USER(S)=      ACCESS=      ACCESS COUNT=      UNIVERSAL ACCESS=
    RACF          USE          000595              NONE

```



```
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE          RESUME DATE=NONE
OMVS INFORMATION
-----
GID= 0000000005
READY
END
```

---

## RACF parameter library IRROPTxx

For each system, determine whether the RACF parameter library is being used and where it is located. This configuration is referenced in the RACF address space procedure. It is possible to run an RRSF network without a RACF parameter library. However, it involves issuing RACF SET and TARGET commands every time the RACF address space is started, and can be complicated. Use the RACF parameter library instead.

Also, use the parameter library to start Dynamic Parse. Example 2-5 shows the initial RACF parameter library that starts Dynamic Parse. Add the TARGET and SET commands needed to implement the RRSF network to this library. The SET INCLUDE(xx) command can be included to start additional members of the parameter library if you want.

*Example 2-5 SYS1.PARMLIB(IRROPT00)*

---

```
ALLOCATE FILE(SYSUT1) DATASET('SYS1.SAMPLIB(IRRDPDS)') SHR
IRRDPIO0 UPDATE
FREE FILE(SYSUT1)
```

---

## RACF installation exits

Check the RACF installation exits and ensure that, if there are differences among nodes, that they are compatible. Having different exits on different nodes in the RRSF network might cause inconsistent results when a command, password, or application update is propagated from one node to another. If you successfully change your password on one node and it is propagated to a remote node running a different password exit, the new password might be unacceptable to the remote node's password exit. In this situation, the change would be unsuccessful.

Pay particular attention to these RACF exits:

- ▶ ICHPWX01
- ▶ ICHPWX11
- ▶ ICHDEX01
- ▶ ICHRDY01

## Template and dynamic parse versions

Issue the RACF SET LIST operator command to determine the dynamic parse and template versions on each system. All systems have compatible versions of dynamic parse and templates. While propagating RACF maintenance or upgrades across your systems, the dynamic parse or template versions might not be consistent throughout your environment. This inconsistency causes errors if trying to use new segment fields introduced with the maintenance/upgrade. Generally, do not use any new segment fields until the RACF updates that introduced them is propagated throughout all of the systems in your RRSF network.

## SETROPTS

List your SETROPTS options and ensure that they are compatible across nodes. RRSF issues a warning message if any of the following are not consistent:

- Enhanced Generic Name (EGN) data set settings

Use EGN data set names for all databases that participate in RRSF. If you have some databases not yet using EGN, convert those databases to use EGN as shown in **1** in Example 2-6.

*Example 2-6 Excerpt from SETR LIST showing EGN settings*

---

```
AUTOMATIC DATASET PROTECTION IS NOT IN EFFECT
ENHANCED GENERIC NAMING IS IN EFFECT 1
REAL DATA SET NAMES OPTION IS INACTIVE
```

---

- Password options

Check your password rules at each node to verify the value of the minimum change interval **1**, whether mixed case support was active **2**, and any site-specific rules **3**. Change these items so that they match on all your systems as shown in Example 2-7.

*Example 2-7 Excerpt from SETR LIST showing password options*

---

```
PASSWORD PROCESSING OPTIONS:
PASSWORD CHANGE INTERVAL IS 90 DAYS. 1
  PASSWORD MINIMUM CHANGE INTERVAL IS 0 DAYS.
  MIXED CASE PASSWORD SUPPORT IS NOT IN EFFECT 2
  NO PASSWORD HISTORY BEING MAINTAINED.
  user IDs NOT BEING AUTOMATICALLY REVOKED.
  NO PASSWORD EXPIRATION WARNING MESSAGES WILL BE ISSUED.
  NO INSTALLATION PASSWORD SYNTAX RULES ARE PRESENT. 3
```

---

- GENCMD(DATASET)

Generic commands are allowed for the DATASET **1** class on all of our nodes, as is probably the case on most systems (Example 2-8).

*Example 2-8 Excerpt from SETR LIST showing generic commands*

---

```
GENERIC COMMAND CLASSES = DATASET 1 ACCTNUM ACICSPCT AIMS ALCSAUTH
```

---

- GENERICOWNER

On the example systems, GENERICOWNER **1** is not in effect (Example 2-9). You can decide whether to have GENERICOWNER in effect in your environment, but ensure that the setting is consistent across all nodes.

*Example 2-9 Excerpt from SETR LIST showing GENERICOWNER setting*

---

```
DEFAULT RVAR PASSWORD IS IN EFFECT FOR THE SWITCH FUNCTION.
DEFAULT RVAR PASSWORD IS IN EFFECT FOR THE STATUS FUNCTION.
SECLABEL CONTROL IS NOT IN EFFECT
GENERIC OWNER ONLY IS NOT IN EFFECT 1
COMPATIBILITY MODE IS NOT IN EFFECT
MULTI-LEVEL QUIET IS NOT IN EFFECT
MULTI-LEVEL STABLE IS NOT IN EFFECT
```

---

Differences in the previous options generate a warning message. Compare all SETROPTS settings, and ensure that differences in any of them will not cause problems in your RRSF network.

### **RACF classes**

List the RACF classes defined to the class descriptor table ICHRRCODE and in the RACF CDT class. Compare the locally defined classes on each node. If updates or commands are propagated for any of these classes, make sure that they exist on all of the nodes or errors will occur.

### **RACF profiles**

If you plan on propagating commands and updates to keep your RACF databases synchronized, compare the profiles in each database. Resolve discrepancies for those classes you want to keep in sync. You must decide how to handle these situations:

- ▶ When a more specific profile on one node undercuts a more generic profile on another node
- ▶ When the same profile exists on more than one node with different attributes, such as the UACC, or when the same user ID exists with different owners on different nodes.

Run IRRDBU00 unloads of the RACF databases, then compare them with execs to find the discrepancies. SYS1.SAMPIB(IRRICE) contains several samples that produce reports to help analyze your RACF databases. An unsupported REXX tool called DBSYNC is available from the RACF web page to assist with comparing and modifying your databases to synchronize them. The RACF web page is available at:

<http://www-03.ibm.com/systems/z/os/zos/features/racf/>

## **2.1.6 Workspace data sets**

RACF uses Virtual Storage Access Method (VSAM) data sets to temporarily hold data that is sent throughout the RRSF Network. On each system, there is an INMSG and an OUTMSG workspace data set for each remote connection. There is also an INMSG and OUTMSG workspace data set for the local node.

The INMSG data set for a remote node is used to hold commands, application updates, and password changes directed in to the local node from the remote node. It also holds output of updates previously sent out to the remote node from the local node. The OUTMSG data set for a remote node holds the commands, application updates, and password changes directed out to the remote node from the local node. It also holds the output of updates previously sent in to the local node from the remote node.

The INMSG and OUTMSG data sets for the local node holds commands directed to the local node from itself and updates to user IDs that have local user ID associations defined.

When a command, application update, or password change is propagated from the local node to a remote node, a copy of the command is saved in the IOUTMSG workspace data set for the remote node. It is then passed to the remote system and stored in that system's INMSG data set for the originating system. After it is received, the command is removed from the originating system's OUTMSG. The command is processed on the remote system and removed from its INMSG data set. The results are put in the remote system's OUTMSG data set for the originating system. The results are then passed to the originating system's INMSG data set for the remote system. After they are received, they are deleted from the remote system's OUTMSG data set. The local system passes the results to the user ID who originated the request and removes it from its OUTMSG data set. In reality, it is much more

complicated than this, but this example just shows the basic flow of requests and responses through the workspace data sets.

In a TCP/IP configuration, the names of the workspace data sets for the local node follow the naming convention: *prefix.local-sys.ds-identity*. The data sets for the remote nodes follow the convention: *prefix.local-sys.remote-sys-or-node.ds-identity*. The variables have the following definitions:

<b>prefix</b>	The value of the PREFIX keyword in the TARGET command, which can be more than one qualifier. You might want to have this be a high-level qualifier that is SMS-managed to help with administration of the data sets. For an MSN where the master catalog is not shared, you might want to have this be an alias defined in a user catalog. This configuration allows all MSN workspace data sets to be viewed from any of the MSN systems.
<b>local-sys</b>	By default, this is the local system SYSNAME from the CVT. You can override this default by specifying a value for WDSQUAL on the TARGET command. Generally, leave this setting as the default, which is SYSNAME. If you use multiple workspace data sets, using the SYSNAME helps track them all.
<b>remote-sys-or-node</b>	By default, the remote system name is specified by the TARGET command. The remote node name is specified by the TARGET command in the case of a remote single system node. You can override this default with the WDSQUAL name in the TARGET command, but generally use the default.
<b>ds-identity</b>	<p>The indicator as to whether the data set is an INMSG or OUTMSG workspace data set.</p> <p>INMSG: Temporarily holds incoming requests and responses from remote nodes to requests previously sent out to the remote nodes.</p> <p>OUTMSG: Temporarily holds outgoing requests and responses from requests previously sent in from remote nodes.</p>

## 2.1.7 Network considerations

This section describes the items related to networking that you must consider when implementing RRSF.

### IP addresses

To identify the systems of the RRSF network, you need to know the IP addresses or the domain names of the systems involved. Use these addresses or names in the TARGET commands. One way to find the IP addresses is to log on to each system and issue the command **hometest**. To determine the corresponding domain names, issue the command **nslookup <ip address>** from any of the TCP/IP connected systems.

Table 2-1 lists the IP addresses and domain names for the systems in the example environment.

Table 2-1 Example IP addresses and domain names

System	IP address	Domain name
SC74	9.12.4.70	wtsc74.itso.ibm.com
SC75	9.12.4.72	wtsc75.itso.ibm.com

System	IP address	Domain name
SC76	9.12.4.126	wtsc76.itso.ibm.com
SC80	9.12.4.45	wtsc80.itso.ibm.com
SC81	9.12.4.47	wtsc81.itso.ibm.com

### Protection of TCP/IP stack and RRSF listener port

Determine whether the TCP/IP stack is protected by using a RACF SERVAUTH profile. The naming convention for this resource is: *EZB.STACKACCESS.sysname.tcpname*.

Similarly, determine whether the RRSF listener port is protected by using a SERVAUTH profile. The naming convention for this resource is:  
*EZB.PORTACCESS.stackname.SAFname.RRSF*.

If either of the previous systems are not covered by a SERVAUTH profile, create a profile and grant READ access to the user ID associated with the RACF address space. For more information about this process, see Chapter 3, “Configuring RRSF for TCP/IP” on page 27.

### RRSF trust policy

A trust policy based on digital certificates must be implemented before RRSF can use TCP/IP to communicate with remote nodes. There are three possible approaches for implementing this trust policy:

- ▶ Use a single, self-signed certificate to be used by all of the nodes.
- ▶ Set up one of the RRSF systems as a certificate authority (CA) to sign a server certificate for each RRSF node.
- ▶ Use an external CA.

In the sample environment, the second choice is implemented. For more information about this process, see 3.4, “Creating digital certificates and key rings” on page 46.

## 2.2 Converting from APPC to TCP/IP

This section describes the planning to convert an already established RRSF network to change the protocol from APPC to TCP/IP.

## 2.2.1 Original APPC protocol configuration

The configuration of the example RRSF network using APPC was similar to that of the RRSF network using TCP/IP in 2.1, “Implementing RRSF using TCP/IP” on page 14. This APPC protocol network, however, contains a system that runs a release of RACF that does not support TCP/IP as the protocol for RRSF. In Figure 2-3, the nodes PLEX75 and PLEX76 contain the same systems as before. However, node PLEX81 contains an additional system, ADCD, that is running RACF V1R11.

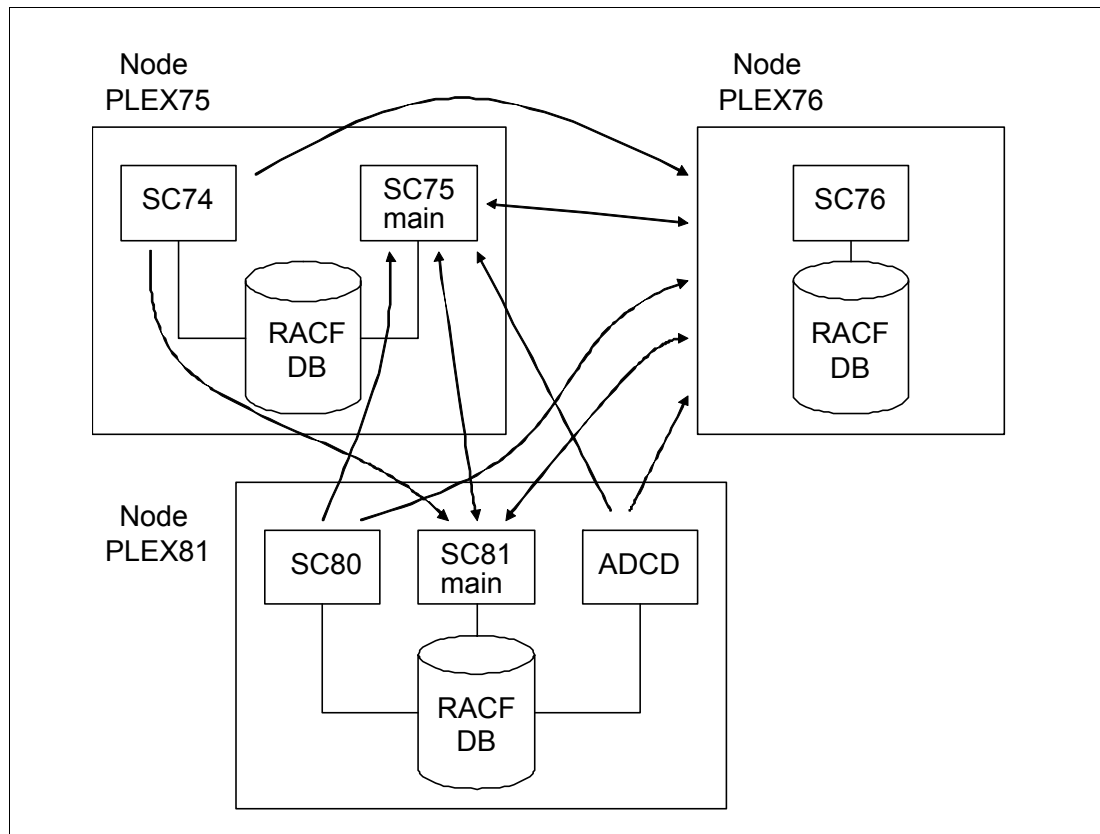


Figure 2-3 The RRSF network to be converted from an APPC protocol to TCP/IP.

Most attributes of the RRSF network using APPC as the protocol are the same as the example RRSF network using TCP/IP. The following attributes are the same:

- ▶ Nodes, systems, and main systems (except for system ADCD being a part of PLEX81)
- ▶ RRSF functions
- ▶ RACF address spaces
- ▶ Installation exits
- ▶ Template and dynamic parse versions (except for ADCD being at V1R11 levels)
- ▶ Classes and profiles (except for classes and segment fields introduced with RACF V1R12 or V1R13 that are not usable on ADCD)

Because the previous attributes were already in place, they did not have to be considered when planning for the conversion. For more information about the initial planning, see 2.1, “Implementing RRSF using TCP/IP” on page 14.

The following attributes differ between the APPC protocol RRSF network and the TCP/IP protocol RRSF network:

- ▶ Protocol used
- ▶ Existence of earlier system ADCD
- ▶ Commands in the RACF parameter library IRROPTxx
- ▶ Workspace data set names

## 2.2.2 Earlier systems

Having the earlier system ADCD in the RRSF network demonstrates that the V1R13 systems can be converted to TCP/IP while the V1R11 system continues to communicate by using APPC.

Regardless of using APPC or TCP/IP, do not use a previous version on the main system. Because ADCD is not a main system, this is not an issue in the sample network. However, there might be some newer classes and segments in the PLEX81 RACF database that ADCD cannot recognize.

## 2.2.3 RACF parameter library IRROPTxx

Determine which IRROPTxx member is currently being used to define your APPC RRSF network. Look in the SYSLOG for message IRRG008I as shown in Example 2-10. In a network that uses both TCP/IP and APPC, you will have TARGET commands in your parameter library for both TCP/IP and APPC.

*Example 2-10 Message in SYSLOG showing the RACF parameter library member.*

---

IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY **IRROPT76**.

---

## 2.2.4 Workspace data sets

The naming convention for workspace data sets when using APPC as the protocol is different than the convention used when TCP/IP is the protocol. For more information about naming conventions when using TCP/IP, see 2.1.6, “Workspace data sets” on page 21.

In an APPC configuration, the names of the workspace data sets for the local node follow the naming convention: *prefix.local-sys.ds-identity*. The data sets for the remote nodes follow the convention: *prefix.local-luname.remote-luname.ds-identity*. The variables have the following definitions:

<b>prefix</b>	The value of the PREFIX keyword in the TARGET command, which can be more than one qualifier.
<b>local-luname</b>	The LU-name of the local node.
<b>remote-luname</b>	The LU-name of the remote node unless overridden with the WDSQUAL name in the TARGET command.
<b>ds-identity</b>	The indicator of whether the data set is an INMSG or OUTMSG workspace data set.  INMSG: Temporarily holds incoming requests and responses from remote nodes to requests previously sent out to the remote nodes.  OUTMSG: Temporarily holds outgoing requests and responses from requests previously sent in from remote nodes.

In an RRSF network that uses both TCP/IP and APPC, you will have workspace data sets in both the TCP/IP and APPC conventions.

If you have had problems in the past with the size or location of your workspace data sets, the conversion is a good time to address them.

### **2.2.5 Network considerations**

When converting from APPC to TCP/IP, you have the same network considerations as when implementing RRSF using TCP/IP for the first time. For more information, see 2.1.7, “Network considerations” on page 22.





## Configuring RRSF for TCP/IP

This chapter addresses the step by step procedure to configure RRSF through TCP/IP. This chapter is important if you have never used the RRSF facility and want to configure your system to enable RRSF using TCP/IP. If you are already using RRSF using APPC, see Chapter 4, “Converting APPC connections to TCP connections” on page 83 to change to TCP/IP from APPC. However, this chapter addresses the configuration of TCP/IP that is required before changing from APPC to TCP/IP.

A major step in setting up RRSF through TCP/IP is to configure AT-TLS to protect the traffic between the RRSF nodes.

This chapter includes the following sections:

- ▶ Overview of AT-TLS
- ▶ Setting up the PAGENT-started task
- ▶ Configuring AT-TLS policy for RRSF server and client
- ▶ Creating digital certificates and key rings
- ▶ Updating TCP/IP to enable RRSF
- ▶ Updating necessary RACF profiles
- ▶ Verifying the RRSF setup

## 3.1 Overview of AT-TLS

AT-TLS stands for Application Transparent Transport Layer Security. TLS (Transport Layer Security) and SSL (Secure Sockets Layer) are protocol technologies used to protect data exchanges between client and server applications. Using TLS/SSL, the client and server can confirm each other's identity. Their data flows can be encrypted to protect e-commerce transactions and other sensitive data as they navigate across the network.

The AT-TLS protection is driven by the security policies you decide for the application. Policy Agent (PAGENT) is a system component that reads your security policies and interfaces with the TCP/IP stack to enforce them.

The AT-TLS policy is installed for the RACF remote sharing facility (RRSF). TCP/IP and PAGENT then ensure that the RRSF nodes will identify and authenticate each other using SSL handshakes before establishing a connection. They encrypt the data that is exchanged between them.

**Attention:** The RRSF application is designed so that the nodes do not connect to each other if an active AT-TLS policy is not in place. This configuration ensures that all the data that is exchanged between the RRSF nodes are fully protected

The procedure to set up RRSF including AT-TLS security consists of these steps:

1. Set up PAGENT-started task
2. Configure AT-TLS policy for RRSF server and client
3. Create digital certificates and key rings
4. Update TCP/IP to enable RRSF
5. Update necessary RACF profiles
6. Verify the RRSF setup

## 3.2 Setting up the PAGENT-started task

The Policy Agent (PAGENT) system component enforces your policies that control network security, traffic prioritization, bandwidth management, network behavior, routing, and resource balancing in the z/OS environment. It reads and parses these policies, and stores the policy definitions which are acted on by the TCP/IP task.

Your site might already have PAGENT if you are using any TCP/IP security features. In that case, you can skip this step. The following steps are needed to set up the PAGENT:

1. Set up the PAGENT-started task procedure
2. Define the necessary RACF profiles for PAGENT
  - Define PAGENT-started task to RACF
  - Permit user PAGENT to the BPX.DAEMON facility
  - Permit authorized users to control PAGENT
3. Restrict access to the **pasearch** command
4. Set up TTLS Stack Initialization access control

### 3.2.1 Setting up the PAGENT started task procedure

You can find the sample started task procedure for PAGENT in TCP/IP.SEZAINST(EZAPAGSP). The PAGENT started task procedure shown in Example 3-1 was used in the example environment.

*Example 3-1 Set up the PAGENT started task procedure*

---

```
//PAGENT PROC
//*
//* IBM Configuration Assistant for z/OS Communications Server
//*
//* Status = CSV1R11
//*
//PAGENT EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,
// PARM='ENVAR("TZ=EST5EDT4")/-c /etc/pagent.conf -l /tmp/pagent.log' 1
//*
//* Example of passing parameters to the program (parameters must
//* extend to column 71 and be continued in column 16):
//* PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/-c /etc/pagent3.conf -l
//* SYSLOGD'
//*
//* Provide environment variables to run with the desired
//* configuration. As an example, the data set or file specified by
//* STDENV could contain:
//*
//* PAGENT_CONFIG_FILE=/etc/pagent2.conf
//* PAGENT_LOG_FILE=/tmp/pagent2.log
//* LIBPATH=/usr/lib
//* TZ=EST5EDT4
//*
//* For information on the above environment variables, refer to the
//* IP Configuration Reference. Other environment variables can also
//* be specified via STDENV.
//*
//STDENV DD DUMMY
```

---

Number **1** corresponds to the following information:

- ▶ You can use environment variables configured in an IBM MVS data set or z/OS UNIX file specified by the STDENV DD to run with the required configuration.
- ▶ **-c /etc/pagent.conf** specifies the configuration file used by PAGENT to hold the policies.
- ▶ **-l /tmp/pagent.log** specifies the log file name used by PAGENT. You can also direct the PAGENT log to SYSLOGD.
- ▶ **TZ=EST5EDT4** specifies the time zone in your working location.

In the example, the AT-TLS policy was specified in a z/OS UNIX file, **/etc/pagent.conf**, as shown in Example 3-2.

*Example 3-2 PAGENT Configuration fileT*

---

```
## /etc/pagent.conf
##
## Image: SC75
##
TcpImage TCP/IP FLUSH 600
```

**1**

```
##
TTLSSConfig /etc/tlsPol_rrsf          FLUSH NOPURGE
##TTLSSConfig //'USER1.TCPCS.TCPPARMS(ATTLS)' FLUSH NOPURGE
##QoSConfig //'USER1.TCPCS.TCPPARMS(QOS)' FLUSH NOPURGE
##IDSSConfig //'USER1.TCPCS.TCPPARMS(IDS)' FLUSH NOPURGE
##IPSecConfig //'USER1.TCPCS.TCPPARMS(IPSEC)'
```

---

**1** The **TcpImage TCP/IP FLUSH 600** command has these operands:

- ▶ **TCP/IP** specifies the TCP/IP stack name on which this policy applies. Install the AT-TLS policy on the TCP/IP stack of every RRSF node.
- ▶ The **FLUSH** parameter specifies that the PAGENT deletes all the policies in the policy agent and TCP/IP stack. This deletion occurs when PAGENT is started, or when the MODIFY PAGENT,REFRESH command is issued.
- ▶ **600** specifies the time interval (in seconds) for checking for the creation or modification of the PAGENT configuration files.

**2** The **TTLSSConfig /etc/tlsPol\_rrsf** command specifies the file that holds the AT-TLS policy to be used

**Remember:** The policies installed in the TCP/IP stack are deleted only at PAGENT startup time if the FLUSH parameter is specified. This setting prevents the policies from being deleted from the TCP/IP stack unexpectedly if PAGENT terminates abnormally.

### 3.2.2 Defining the necessary RACF profiles for PAGENT

This process consists of the following steps:

1. Define PAGENT started task to Resource Access Control Facility (RACF)
2. Permit user PAGENT to the BPX.DAEMON facility
3. Permit authorized users to control PAGENT

#### Defining PAGENT started task to RACF

Define a PAGENT user ID with an OMVS segment.

**Tip:** For more information about this step, see 3.6.1, “Adding an OMVS segment to RACF subsystem user ID” on page 69.

Associate this user ID with the PAGENT-started task, and then define that task to RACF.

#### Permitting user PAGENT to the BPX.DAEMON facility

The PAGENT requires access to the BPX.DAEMON facility if it is defined.

#### Permitting authorized users to control PAGENT

Control which users can start and stop PAGENT to reduce the risk of unauthorized users affecting policy-based networking. Define a profile called MVS.SERVMGR.PAGENT in resource class OPERCMDS, and give authorized users access to this facility.

The JCL used for the previous steps is shown in Example 3-3.

*Example 3-3 Defining RACF Profiles for the PAGENT started task*

---

```
//RPAGENT JOB RPAGENT,CLASS=A,NOTIFY=&SYSUID
/* Configuration Assistant, 2012.05.08 21:36:36
/* RACF Directives for Pagent startup
/* 1. Define user PAGENT to the STARTED class
/* 2. Permit user PAGENT to the BPX.DAEMON facility
/* Note that BPX.DAEMON is optional and the PERMIT
/* is only needed if BPX.DAEMON is defined.
/* 3. Permit user -user- the ability to control pagent
/* Replace -user- with user ID(s) to receive control permission
//STEPX EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
/* Define user PAGENT to the STARTED class */
  SETROPTS CLASSACT(STARTED)
  SETROPTS RACLIST(STARTED)
  SETROPTS GENERIC(STARTED)
  ADDUSER PAGENT DFLTGRP(OMVSGRP) NOPASSWORD +
    OMVS(UID(18136) SHARED HOME('/'))
  RDEFINE STARTED PAGENT.* STDATA(USER(PAGENT))
  SETROPTS RACLIST(STARTED) REFRESH
  SETROPTS GENERIC(STARTED) REFRESH
/* Permit user PAGENT to the BPX.DAEMON facility */
  PERMIT BPX.DAEMON CLASS(FACILITY) ID(PAGENT) ACCESS(READ)
  SETROPTS RACLIST(FACILITY) REFRESH
/* Permit user -user- to control PAGENT */
  SETROPTS CLASSACT(OPERCMDS)
  SETROPTS RACLIST(OPERCMDS)
  RDEFINE OPERCMDS MVS.SERVMMGR.PAGENT UACC(NONE)
  PERMIT MVS.SERVMMGR.PAGENT CLASS(OPERCMDS) +
    ID(PAGENT,RAA,PRICHAR,SYS1) ACCESS(CONTROL)
  SETROPTS RACLIST(OPERCMDS) REFRESH
/*
```

---

### 3.2.3 Restricting access to the pasearch command

The **pasearch** command is used to obtain details of the security policies on your system. This is a sensitive command and needs to be protected. The profile is defined in the **SERVAUTH** class. Use the commands shown in Example 3-4 to restrict access to the **pasearch** command.

*Example 3-4 Restricting access to the pasearch command to authorized users*

---

```
RDEFINE  SERVAUTH EZB.PAGENT.** UACC(NONE)
PERMIT   EZB.PAGENT.** CLASS(SERVAUTH) ID(PAGENT,RAA,PRICHAR) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
```

---

### 3.2.4 Set up TTLS Stack Initialization access control

When you use AT-TLS, z/OS does not allow socket-based applications to start before PAGENT is running. This process ensures that all the security policies are enforced. However, certain essential applications need to start before PAGENT. Example 3-5 on page 32 shows the commands used to define the RACF profile. This profile then allows only authorized applications to use TCP/IP sockets before PAGENT is active to enforce any security policy. PAGENT is the only application in the example environment test that needed to use socket services before any policy is in place.

**Requirement:** Do not permit the RACF subsystem user ID to the stack initialization resource (INITSTACK). If you do, the remote connections might fail during IPL.

*Example 3-5 Setting up TTLS Stack Initialization access control*

```
//RAMA JOB (999,POK),'RAMA AYYAR',CLASS=A,MSGLEVEL=(1,1),
// MSGCLASS=A,NOTIFY=&SYSUID
//INITSTAC EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  SETROPTS CLASSACT(SERVAUTH)
  SETROPTS RACLIST (SERVAUTH)
  SETROPTS GENERIC (SERVAUTH)
  RDEFINE SERVAUTH EZB.INITSTACK.*.* UACC(NONE)
  PERMIT EZB.INITSTACK.*.* +
    CLASS(SERVAUTH) ID(PAGENT) ACCESS(READ)
  SETROPTS GENERIC(SERVAUTH) REFRESH
  SETROPTS RACLIST(SERVAUTH) REFRESH
/*
```

### 3.2.5 Verifying the PAGENT setup

Issue the command S PAGENT and look for the **EZD1576I PAGENT READY** message as shown in Example 3-6.

*Example 3-6 Verifying the PAGENT setup*

```
S PAGENT
EZZ8431I PAGENT STARTING
EZZ8432I PAGENT INITIALIZATION COMPLETE
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCP/IP :
EZD1586I PAGENT HAS INSTALLED ALL LOCAL POLICIES FOR TCP/IP
EZD1576I PAGENT IS READY FOR SERVICES CONNECTION REQUESTS
```

## 3.3 Configuring AT-TLS policy for RRSF server and client

An RRSF node acts as a client when it sends data to another RRSF node. It acts as a server when it receives data from another RRSF node. Both types of traffic need to be protected. You must code an AT-TLS policy with two policy rules: One for the RRSF client and one for the RRSF server. Define such a policy for every TCP/IP stack used by an RRSF node.

In each policy rule, you must specify these aspects:

- ▶ How to identify the traffic that you need to protect
- ▶ How to authenticate the remote RRSF node
- ▶ How to encrypt the data across the TCP/IP connection

Use the IBM z/OSMF GUI Interface to code your policies. Then install the policies in the file /etc/tlsPol\_rrsf pointed to by the PAGENT configuration file /etc/pagent.conf.

Perform the AT-TLS setup on every RRSF node, installing the same policy on each node.

**Tip:** You can also use the sample AT-TLS policy supplied by IBM in SYS1.SAMPLIB(IRRSTRSF) instead of using z/OSMF. This sample policy is enough to get RRSF communicating through TCP/IP. You can copy it to `/etc/pagent.conf`, bypassing the use of z/OSMF.

### 3.3.1 Identifying the traffic that you need to protect

The RRSF client traffic is identified by these characteristics:

- ▶ It is outbound traffic
- ▶ The remote port is 18136, the standard port used by RRSF server
- ▶ The local port can be from 1024 to 65535
- ▶ Covers all inbound and outbound IP addresses

The RRSF server traffic is identified by these characteristics:

- ▶ It is an inbound traffic
- ▶ The local port is 18136, the standard port used by RRSF server
- ▶ The remote port can be from 1024 to 65535
- ▶ Covers all inbound and outbound IP addresses

IBM supplies default traffic descriptions for the RRSF Client and the RRSF Server in z/OSMF.

### 3.3.2 Authenticating the remote RRSF node

AT-TLS uses TLS handshaking protocol between the RRSF nodes to authenticate each other before establishing a TCP/IP connection. Each node needs to have a digital certificate to identify itself. It also needs to keep the certificate of the signing authority that confirms the validity of certificates it issues. This certificate is required to validate any certificate the node receive from any other nodes.

The certificates are kept in a key ring in the RACF data base. In the policy rule, you need to specify the name of this key ring. In the example environments, the certificates are defined in a key ring named IRR.RRSF.KEYRING.

You also need to specify the options to do the authentication. For example, you can have the following options for specification of client authentication level for the server role:

- ▶ Pass-through: Bypasses client certificate validation.
- ▶ Full: Runs client certificate validation if the client presents a certificate.
- ▶ Required: Requires the client to present a certificate and runs client certificate validation. This setting is the default if the server requires client authentication.
- ▶ SAF Check: Requires the client to present a certificate, runs client certificate validation, and requires the client certificate to have an associated user ID defined to the security product.

The example uses the option Required. If you want to use SAF Check for better security, see 3.4.5, “Using an external CA to sign a server certificate for each RRSF node” on page 66. Do not specify Full.

### 3.3.3 Encrypting the data across the TCP/IP connection

You need to specify the cipher suite to be used by AT-TLS to encrypt/decrypt the data that are sent across the TCP/IP connection. The cipher suite is composed of cryptographic algorithms needed for authentication, session encryption, and hashing (digital fingerprinting). The example uses the cipher suite TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA. RACF does not enforce a minimum encryption level.

### 3.3.4 Coding the policy

Use the following steps to code your policy by using the configuration assistant tool of z/OSMF:

1. Log on to z/OSMF
2. Create a New Backing Store File
3. Add New z/OS Image for the RRSF node
4. Add a TCP/IP stack to the z/OS image
5. Enable ATTLS for the TCP/IP stack
6. Configure RRSF Client:
  - a. Enable the IBM supplied RRSF Client
  - b. Modify the RRSF client to specify the keyring
7. Configure the RRSF Server:
  - a. Enable the IBM supplied RRSF Server
  - b. Modify RRSF Server to specify the Keyring
8. Apply the changes to the TCP/IP stack and to the z/OS image
9. Install the policy:
  - a. View the policy
  - b. FTP the policy to your z/OS system
10. Activate and verify the AT-TLS policy

### Logging on to z/OSMF

Log on to z/OSMF by using the procedure used in your installation as shown in Figure 3-1.

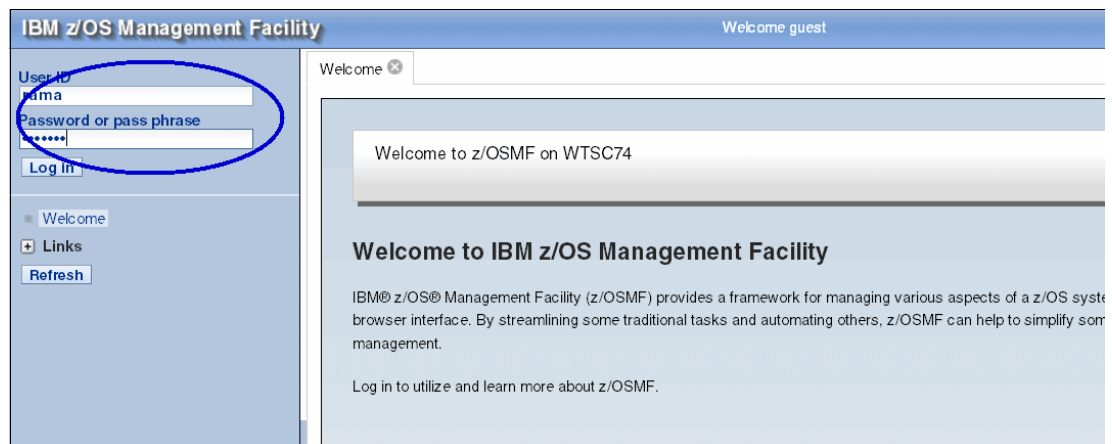


Figure 3-1 Logging on to IBM z/OS Management Facility.



The Welcome window shown in Figure 3-2 is displayed.



Figure 3-2 IBM z/OS Management Facility Welcome window

Click the **Configuration** link on the left navigation panel to display the Configuration Assistant link shown in Figure 3-2. Click that link to get the Main Configuration Assistant window as shown in Figure 3-3.

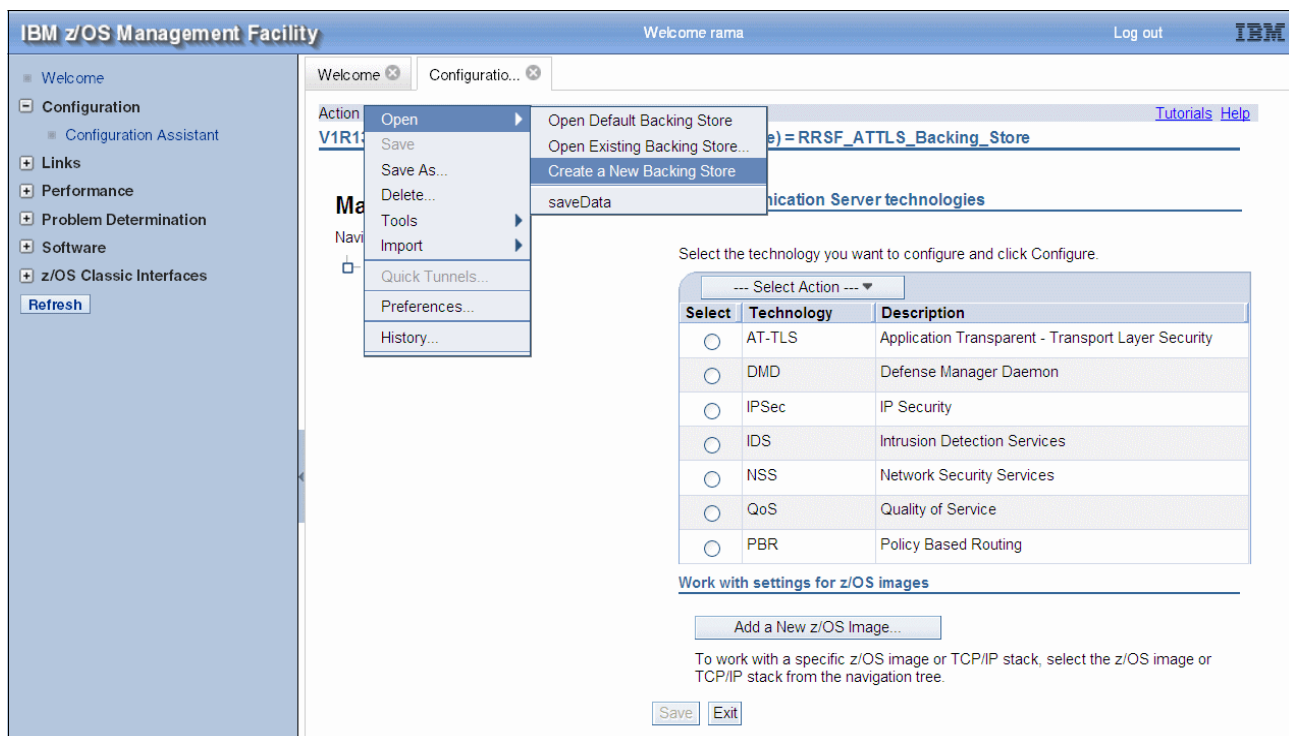


Figure 3-3 Main Configuration Assistant window

## Creating a New Backing Store File

Click **Action** → **Open** → **Create a New Backing Store**. The window to create a backing store file shown in Figure 3-4 is displayed.

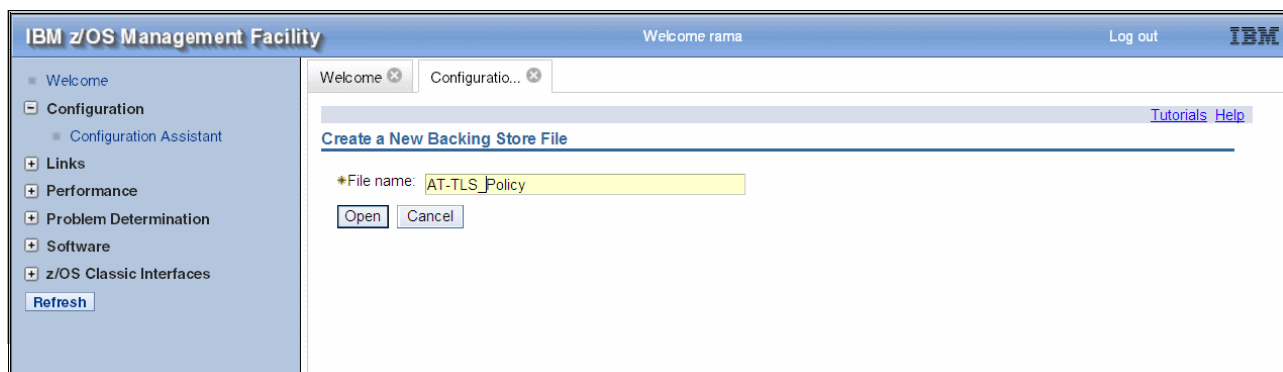


Figure 3-4 Create a New Backing Store File window

Specify a name for the new backing store and click **Open** to display the Main Perspective window shown in Figure 3-5.

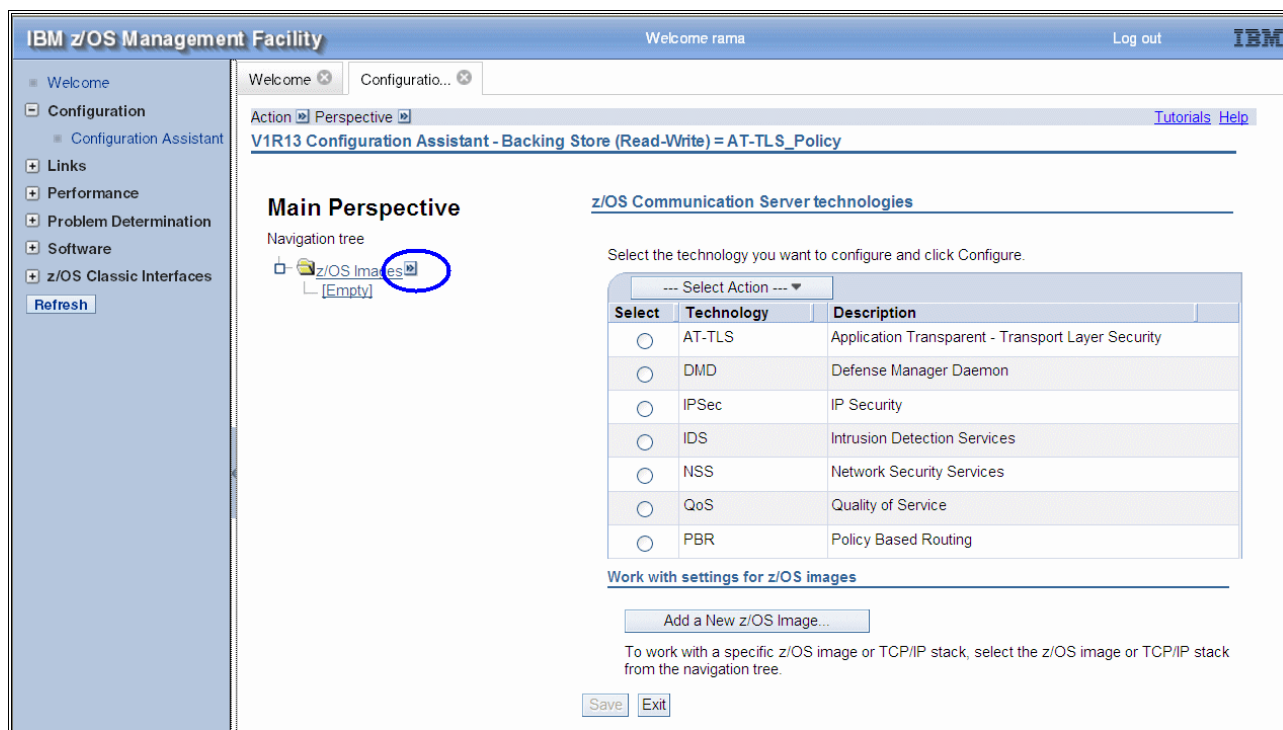
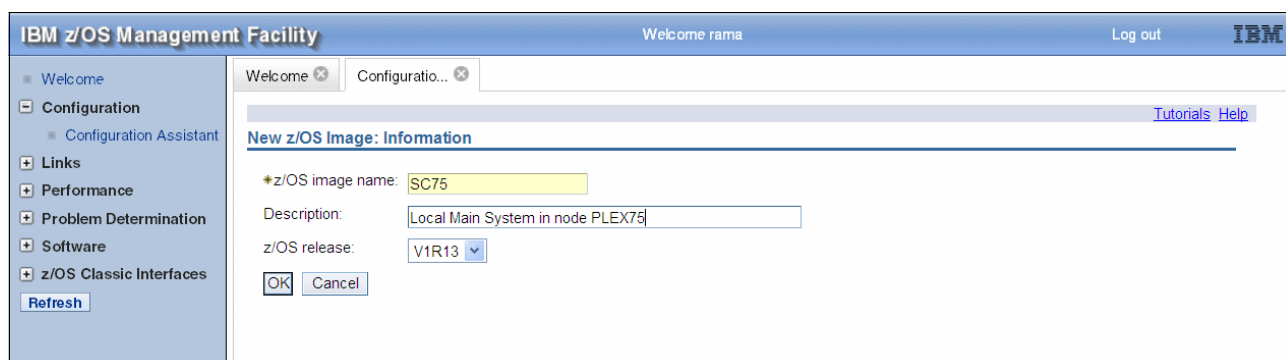


Figure 3-5 Main Perspective window

## Adding New z/OS Image

Click the arrow on the right of the **z/OS Image** link to add a z/OS image. The New z/OS Image: Information window is displayed as shown in Figure 3-6.



IBM z/OS Management Facility

Welcome rama Log out IBM

Welcome Configuration...

Tutorials Help

### New z/OS Image: Information

\*z/OS image name: SC75

Description: Local Main System in node PLEX75

z/OS release: V1R13

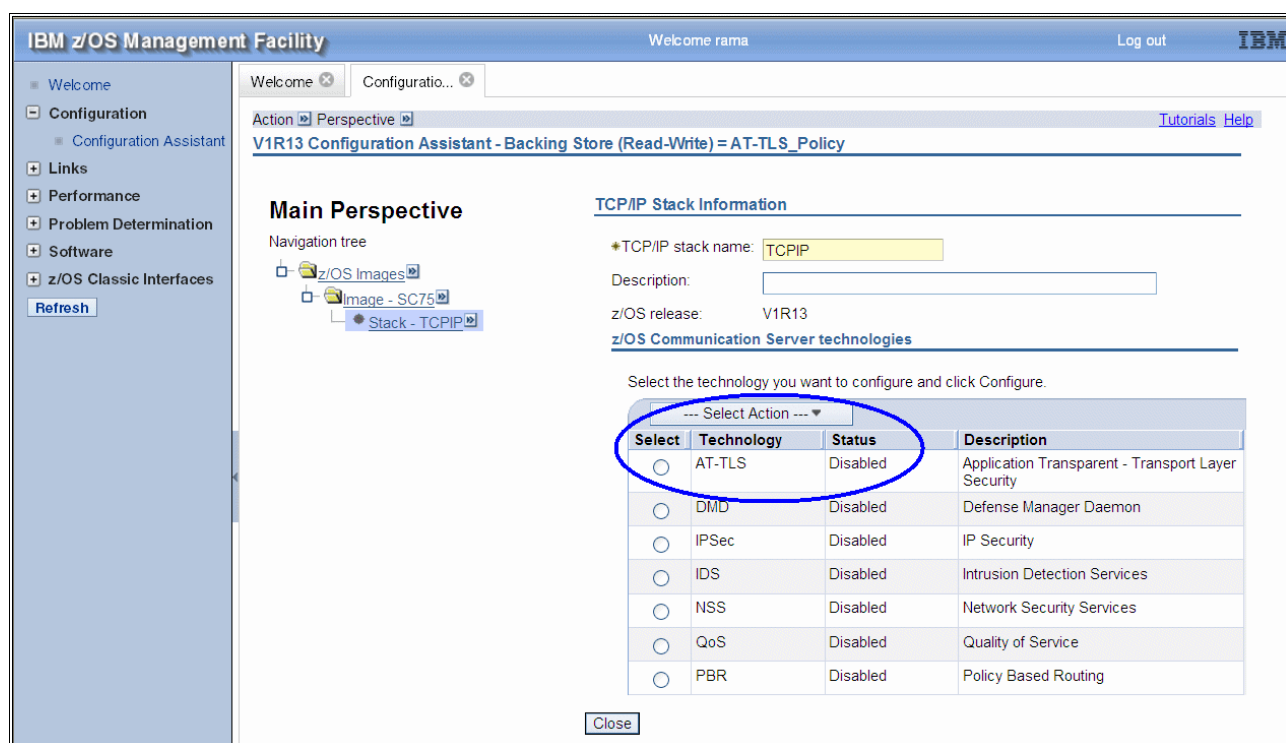
OK Cancel

Figure 3-6 New z/OS Image: Information

Now enter the z/OS image name and click **OK**. Click **Yes** to proceed to the next step to add a TCP/IP stack to the new z/OS image. The New TCP/IP Stack Information window is displayed.

## Adding TCP/IP stack to the z/OS image

Enter the TCP/IP stack name and click **OK**. The TCP/IP Stack Information window shown in Figure 3-7 is displayed.



IBM z/OS Management Facility

Welcome rama Log out IBM

Welcome Configuration...

Action Perspective

Tutorials Help

### V1R13 Configuration Assistant - Backing Store (Read-Write) = AT-TLS\_Policy

#### Main Perspective

Navigation tree

- z/OS Images
  - Image - SC75
    - Stack - TCPIP

#### TCP/IP Stack Information

\*TCP/IP stack name: TCPIP

Description:

z/OS release: V1R13

#### z/OS Communication Server technologies

Select the technology you want to configure and click Configure.

Select	Technology	Status	Description
<input type="radio"/>	AT-TLS	Disabled	Application Transparent - Transport Layer Security
<input type="radio"/>	DMD	Disabled	Defense Manager Daemon
<input type="radio"/>	IPSec	Disabled	IP Security
<input type="radio"/>	IDS	Disabled	Intrusion Detection Services
<input type="radio"/>	NSS	Disabled	Network Security Services
<input type="radio"/>	QoS	Disabled	Quality of Service
<input type="radio"/>	PBR	Disabled	Policy Based Routing

Close

Figure 3-7 TCP/IP Stack Information

## Enabling ATTLS for the TCP/IP stack

Select **AT-TLS** and click **Select Action** → **Enable** to enable the AT-TLS function for the stack. The status is displayed as incomplete as shown in Figure 3-8.

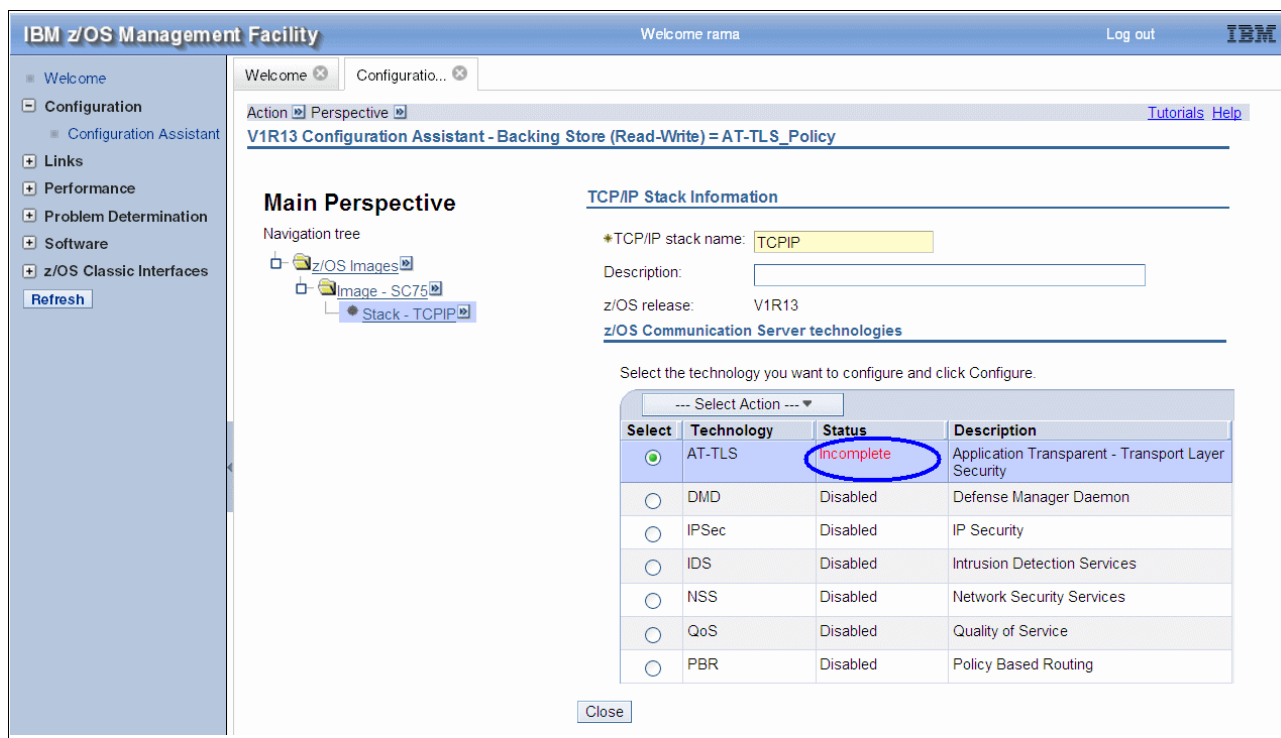


Figure 3-8 TCP/IP Stack Information part 2

Click **Select Action** → **Configure** to get to the AT-TLS Perspective window as shown in Figure 3-9. The status is incomplete for the z/OS image and the TCP/IP stack.

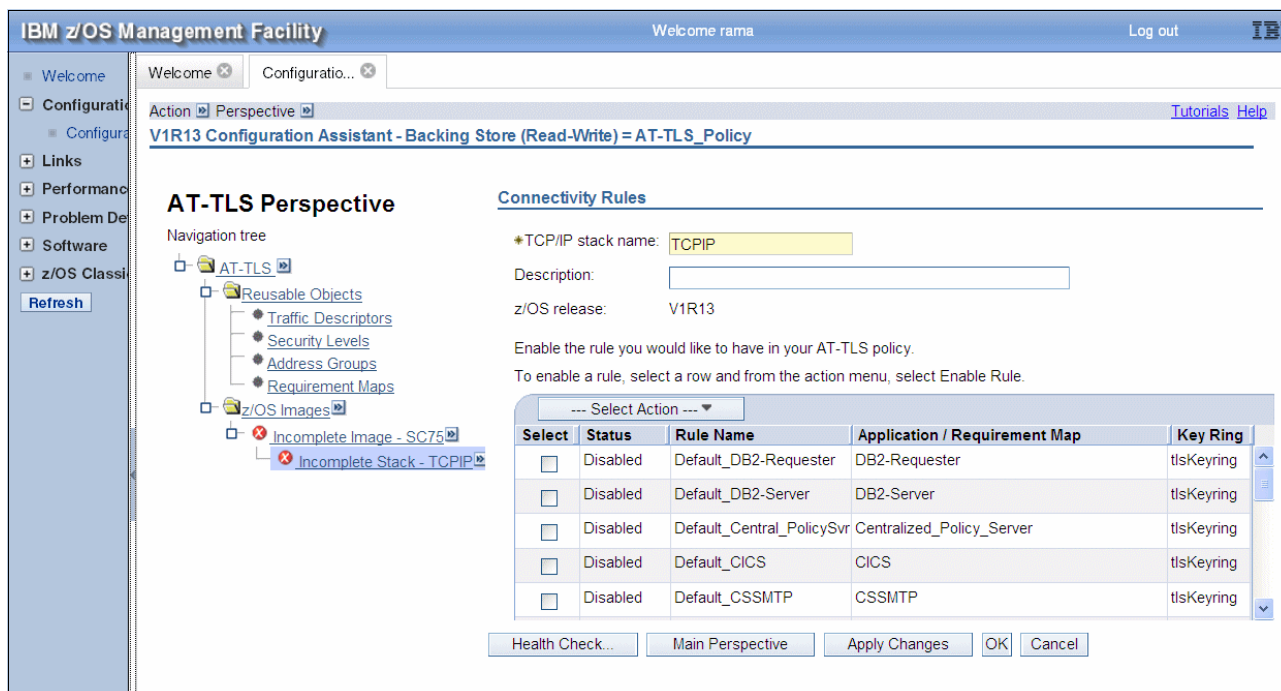


Figure 3-9 AT-TLS Perspective window

## Configuring the RRSF Client

The connectivity rules panel shows the IBM-supplied rules for various applications. Scroll down the list to see RRSF client. It shows status disabled.

## Enabling the IBM supplied RRSF Client

Select **RRSF client rules** and click **OK**. Then click **Action** → **Enable** to display the Verify Rule window as shown in Figure 3-10.

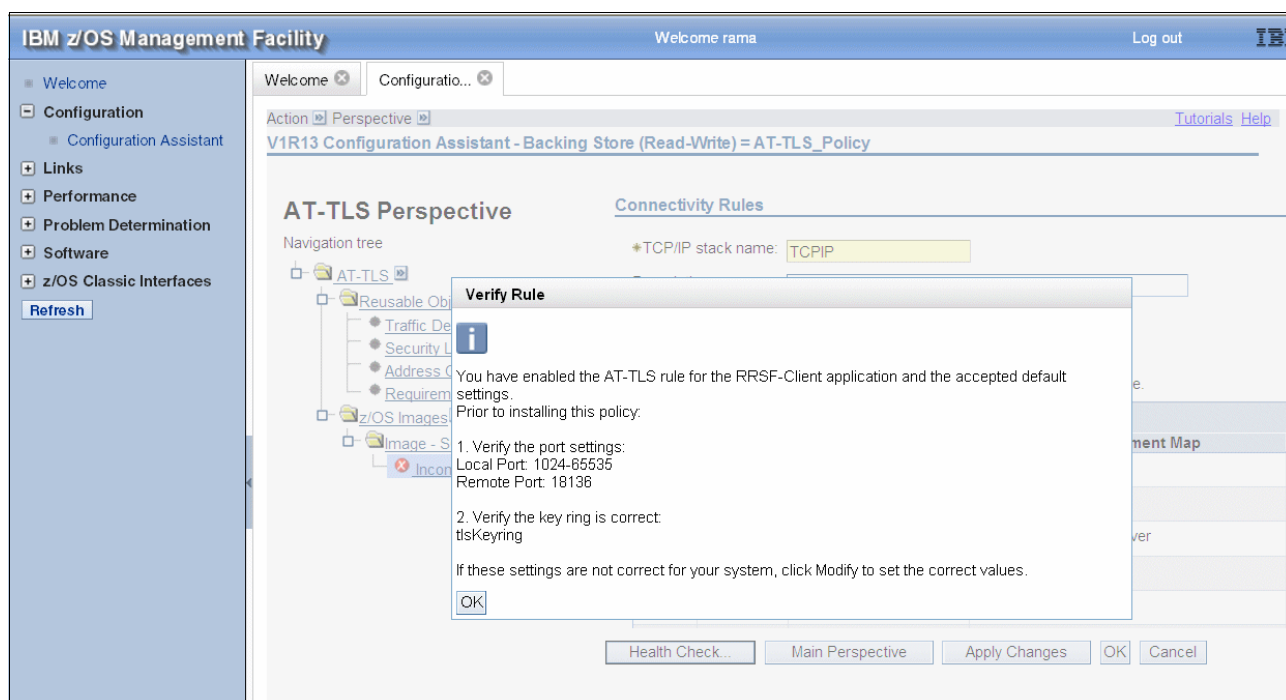


Figure 3-10 Enabling the RRSF\_Client Rule

Verify the port settings and the key ring name by clicking **OK**. The port settings are OK, but you need to modify the Key Ring name.

## Modifying RRSF client to specify the Keyring

With the RRSF client selected, click **Action** → **Modify** to get the next window as in Figure 3-11.

The screenshot shows the 'Modify Rule' window in the IBM z/OS Management Facility. The window has a blue header bar with 'IBM z/OS Management Facility', 'Welcome rama', 'Log out', and the IBM logo. A left sidebar contains a navigation menu with 'Welcome', 'Configuration', 'Links', 'Performance', 'Problem De', 'Software', and 'z/OS Classi'. The main content area is titled 'Modify Rule' and includes a 'Tutorials Help' link. Below the title bar, there's a section for 'AT-TLS rule name' with a text field containing 'Default\_RRSF-Client', an 'Enable rule' checkbox (checked), and a 'Restore Defaults' button. A tabbed interface follows, with 'Traffic' selected. The 'Traffic' tab contains instructions to 'specify the traffic settings' and an 'Application name' field with 'RRSF-Client'. It is divided into 'Local port' and 'Remote port' sections. The 'Local port' section has radio buttons for 'All ports', 'All ephemeral ports' (selected), and 'Ports:' with a text field. The 'Remote port' section has similar options, with 'Ports:' selected and a text field containing '18136'. At the bottom, there are two sections: 'Indicate the TCP connect direction' with radio buttons for 'Either', 'Inbound only', and 'Outbound only' (selected); and 'Specify jobname and user ID' with 'Jobname:' and 'User ID:' text fields.

Figure 3-11 AT-TLS Modify Rule window

RRSF port 18136 is already filled in. Accept all the defaults except the name of the keyring that is going to hold the digital certificates for the RRSF client and server. This keyring does not have to exist already. To create a keyring named IRR.RRSF.KEYRING, click the **Key Ring** tab and go to the window shown in Figure 3-12.

The screenshot shows the IBM z/OS Management Facility interface. The main window is titled 'AT-TLS rule name'. Below the title bar, there are tabs: 'Traffic', 'Role', '\*Key Ring' (selected), 'Data Endpoints', 'Security Level', and 'Advanced'. The 'Key Ring' tab contains the following fields and options:

- \*Rule name: Default\_RRSF-Client
- ☒ Enable rule
- Restore Defaults button
- Use this panel to specify the key ring database and certificate label to use for this rule.
- Key ring database:
  - ☐ Use the key ring database defined for the z/OS image
  - ☒ Use a Simple name (as in an SAF product or in PKCS #11 Token format):
    - \*Key ring: IRR.RRSF.KEYRING
  - ☐ Use this z/OS UNIX file system key database:
    - \*Key database: [empty field]
    - ☒ Key database stash file: [empty field] or
    - ☐ Key database password: [empty field]
- Certificate label: [empty field]
- OK and Cancel buttons at the bottom.

Figure 3-12 Specifying the Key ring database name.gif

Select **Use a Simple name (as in an SAF product or in PKCS #11 Token format)**, specify the keyring name IRR.RRSF.KEYRING, and click **OK**.

**Tip:** The digital certificates for the RRSF client and the RRSF server are created in a new keyring in coming steps. You will create a keyring named IRR.RRSF.KEYRING.

## Configuring RRSF Server

Create the policy for the RRSF-Server the same way as you did for the RRSF-Client. Select RRSF-Server and click **Action** → **Enable**; **Action** → **Modify**. Click the **Key Ring** tab, then specify the keyring IRR.RRSF.KEYRING and click **OK**.

The window shown in Figure 3-13 is displayed.

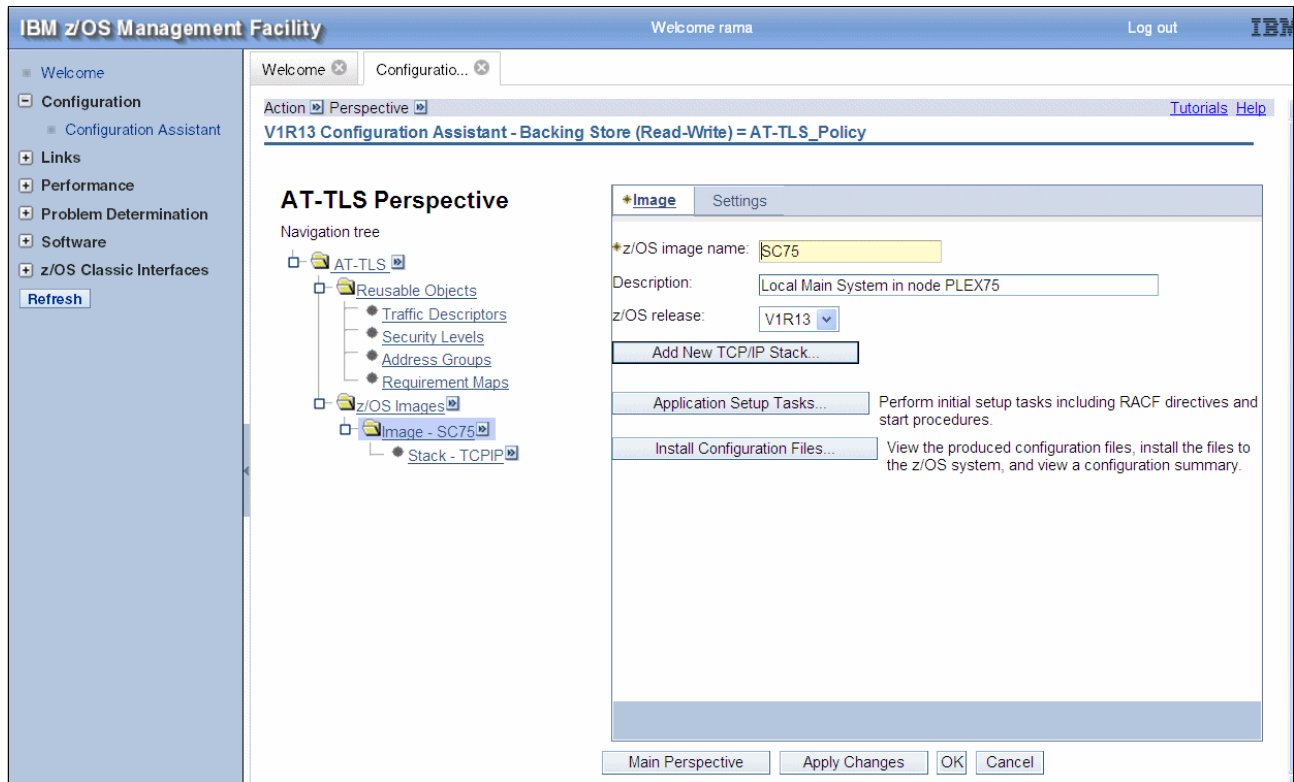


Figure 3-13 Install the policy

## Applying the changes to the TCP/IP stack and to the z/OS image

Click **Apply Changes** to update the TCP/IP stack and the z/OS image.

## Installing the policy

Click the **Install Configuration files** tab to get to the window shown in Figure 3-14.

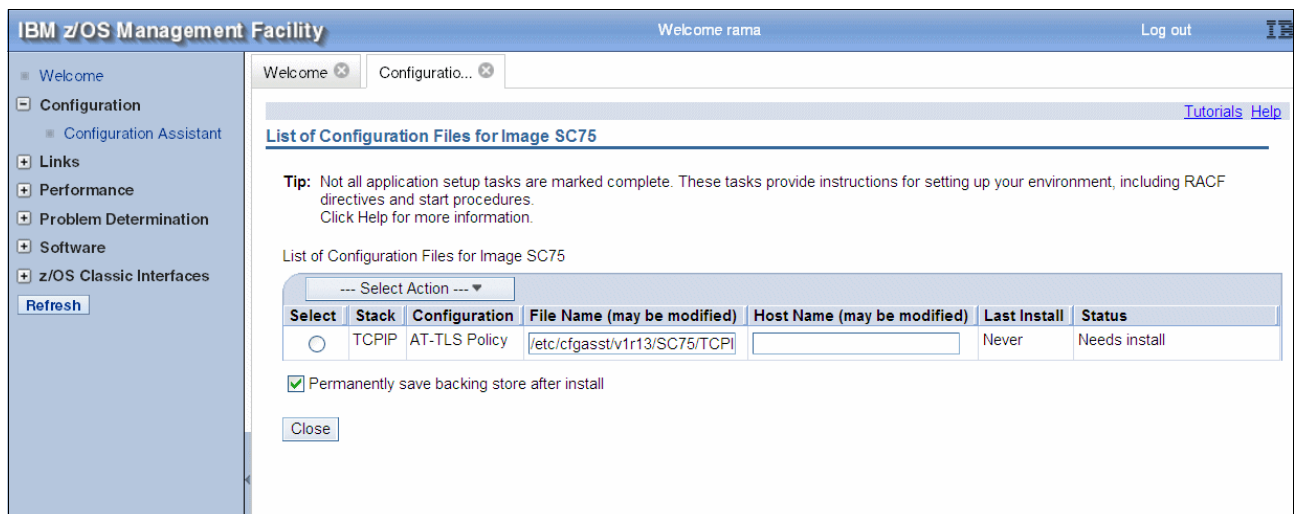


Figure 3-14 List of Configuration Files for Image SC75



Select the radio button and click **Select Action** → **Show Configuration File** to examine the policy generated.

FTP the policy to your z/OS system. On the z/OS image, click **Select Action** → **Install** to display the Install File window. Specify `/etc/tlsPol_rrsf` as the file name to install the policy. Select the FTP option, and specify the host IP address of the SC75 image 9.12.4.72, the user ID, and password for the FTP. Then click **Go** as shown in Figure 3-15.

The screenshot shows the 'IBM z/OS Management Facility' interface. On the left is a navigation pane with 'Configuration' selected. The main area is titled 'Install File'. It contains several sections: 'Install file name' with a text box containing '/etc/tlsPol\_rrsf'; 'Select installation method' with radio buttons for 'Save to disk' and 'FTP' (selected); 'FTP login information' with fields for 'Host name' (9.12.4.72), 'Port number' (21), 'User ID' (rama), and 'Password' (masked with dots), plus a 'Save password' checkbox (checked) and a 'Use SSL' checkbox (unchecked); 'Data transfer mode' with radio buttons for 'Default' (selected), 'Passive', and 'Active'; and a 'Comment for the configuration file prologue (optional)' section with a text box. At the bottom are 'Go', 'Close', and 'View FTP Log' buttons.

Figure 3-15 Install File

The message The FTP file transfer was successful is displayed on a successful FTP transfer.

You can now click **Close** and then **OK** to return to the Main window.

Example 3-7 shows the policy that was created.

#### Example 3-7 Policy created

```
##
## AT-TLS Policy Agent Configuration file for:
##   Image: SC75
##   Stack: TCP/IP
##
## Created by the IBM Configuration Assistant for z/OS Communications Server
## Version 1 Release 13
## Backing Store = AT_TLS_POLICY
## FTP History:
## 2012-05-15 15:53:17 : Rama Ayyar to 9.12.4.72
##
## TLS default rules: Default_RRSF-Client| Default_RRSF-Server |
## End TLS default rules
##
```

```

## End of Configuration Assistant information
TLSRule                                Default_RRSF-Client~1
{
    LocalAddr                          ALL
    RemoteAddr                         ALL
    LocalPortRangeRef                  portR1
    RemotePortRangeRef                  portR2
    Direction                          Outbound
    Priority                            255
    TLSGroupActionRef                   gAct1
    TLSEnvironmentActionRef              eAct1~RRSF-Client
    TLSConnectionActionRef              cAct1~RRSF-Client
}
TLSRule                                Default_RRSF-Server~2
{
    LocalAddr                          ALL
    RemoteAddr                         ALL
    LocalPortRangeRef                  portR2
    RemotePortRangeRef                  portR1
    Direction                          Inbound
    Priority                            254
    TLSGroupActionRef                   gAct1
    TLSEnvironmentActionRef              eAct2~RRSF-Server
    TLSConnectionActionRef              cAct2~RRSF-Server
}
TLSGroupAction                          gAct1
{
    TLSEnabled                        On
}
TLSEnvironmentAction                    eAct1~RRSF-Client
{
    HandshakeRole                      Client
    EnvironmentUserInstance              0
    TLSKeyringParmsRef                  keyR~SC75
}
TLSEnvironmentAction                    eAct2~RRSF-Server
{
    HandshakeRole                      ServerWithClientAuth
    EnvironmentUserInstance              0
    TLSKeyringParmsRef                  keyR~SC75
}
TLSConnectionAction                    cAct1~RRSF-Client
{
    HandshakeRole                      Client
    TTLS cipherParmsRef                  cipher1~AT-TLS_PlatinumClientAut
    TLSConnectionAdvancedParmsRef        cAdv1~RRSF-Client
    CtraceClearText                      Off
    Trace                                2
}
TLSConnectionAction                    cAct2~RRSF-Server
{
    HandshakeRole                      ServerWithClientAuth
    TTLS cipherParmsRef                  cipher1~AT-TLS_PlatinumClientAut
    TLSConnectionAdvancedParmsRef        cAdv2~RRSF-Server
    CtraceClearText                      Off
}

```

```

    Trace                2
}
TTLSTransactionAdvancedParms  cAdv1~RRSF-Client
{
    SSLv3                Off
    TLSv1                Off
    SecondaryMap          Off
}
TTLSTransactionAdvancedParms  cAdv2~RRSF-Server
{
    SSLv3                Off
    TLSv1                Off
    SecondaryMap          Off
}
TTLSTransactionKeyringParms   keyR~SC75
{
    Keyring             IRR.RRSF.KEYRING
}
TTLSTransactionCipherParms    cipher1~AT-TLS_PlatinumClientAut
{
    V3CipherSuites      TLS_RSA_WITH_AES_256_CBC_SHA
}
PortRange                    portR1
{
    Port                 1024-65535
}
PortRange                    portR2
{
    Port                 18136
}

```

---

The lines highlighted in bold illustrate the following aspects of the policy:

- ▶ TLS security is enabled for traffic between all IP addresses to and from port 18136
- ▶ The digital certificates are stored in the key ring named **IRR.RRSF.KEYRING**
- ▶ Cipher suites **TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA** are used for data transfer

However, you can define your own security level and select different cipher suites and different client authentication methods such as SAF Check. For more information, see 3.4.5, “Using an external CA to sign a server certificate for each RRSF node” on page 66.

**Consideration:** If you already have an AT-TLS policy for other applications, open this policy under z/OSMF. Update it to add the RRSF\_Client and RRSF\_Server rules. For more information, see “Configuring the RRSF Client” on page 39 and “Configuring RRSF Server” on page 41.

## Activating and verifying the AT-TLS policy

After loading the policy, refresh the PAGENT task. The output is shown in Example 3-8.

*Example 3-8 Refreshing PAGENT with AT-TLS policy*

```

F PAGENT,REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCP/IP : TTLS

```

---

## 3.4 Creating digital certificates and key rings

Create the digital certificates to identify the RRSF nodes and install them in the key ring specified in the AT-TLS policy setup. This policy was set up during “Modifying RRSF client to specify the Keyring” on page 40. This section addresses how to create the digital certificates.

### 3.4.1 Implementing an RRSF trust policy

When you implement TCP/IP for RRSF node connections, you must implement a trust policy based on digital certificates. This policy allows TCP/IP communication to take place between RRSF nodes. RRSF node connections that use TCP/IP are protected by using AT-TLS. This trust policy is based on the requirements of AT-TLS. It requires that you create a RACF key ring for each node, and one or more signed server certificates.

Use one of these approaches for defining a trust policy:

- ▶ Using the same, self-signed certificate for all RRSF nodes
- ▶ Using an internal certificate authority (CA) to sign a server certificate for each RRSF node
- ▶ Using an external CA

All these options are addressed with some examples. In addition, a detailed step by step implementation scenario with option 2 is provided for the example environment.

With each approach, exclusive use of the signing CA certificate is the basis for securely authenticating the nodes in your RRSF network. A local RRSF node receives a connection attempt from a remote server that presents a digital certificate signed by a CA certificate within the local server's key ring. If the name of that key ring is specified in the AT-TLS policy, the local node accepts it as a valid RRSF node connection. Therefore, you must ensure that your signing CA certificate is used to sign only RRSF server certificates.

Additional security controls can be applied. Some of these controls are briefly described in 3.4.5, “Using an external CA to sign a server certificate for each RRSF node” on page 66. These controls are presented in the context of a situation in which you lack exclusive use of the signing CA certificate.

### 3.4.2 Digital certificates

Network entities authenticate to each other using the trust policy established by digital certificates. The identities of trusted entities are represented by these digital certificates. For an application, the certificate of the application and those of trusted entities for that application are stored in a container called a key ring.

The TLS standard requires the server to send its certificate to the client for validation. It optionally requires the client to send its certificate to the server for validation (also called “client authentication”). This is not the traditional client/server model. Rather, it is a mesh of peers. Therefore, you must enforce client authentication so that both sides of the conversation are authenticated to each other.

RRSF runs within the RACF subsystem address space. Each node can initiate a connection or accept it. So, the RACF subsystem address space identity can act as either the client or the server.

On each system, the RACF address space must have access to a key ring that contains these items:

- A server certificate (with private key) for that RRSF instance
- The signing certificate (public key only) used to sign the RRSF server certificates, and no others

In the simplest case, this process can be accomplished with a single self-signed certificate added to each key ring (if your security policy allows it).

Otherwise, create the signing certificate on one of the systems, and use it to create/sign that system's server certificate. On the other systems, generate a certificate request and send it to the "signer system," where a certificate is generated. Send the certificate back to the original system and add it.

Never use the signing certificate to sign anything but RRSF certificates. For more information, see the *Security Administrator's Guide*.

**Tip:** In the simplified case, you can use a single self-signed certificate that acts as both the certificate authority and the server certificate for every RRSF instance. Define the certificate on one of the systems, add it to the key ring, and export it to all the other systems in their key rings. The downside is that the private key leaves the sanctity of the RACF data base (or ICSF) as it travels around in a password-protected PKCS#12 file. This process might violate your security policy.

If this process does not work, you can still set up RRSF securely by using these steps:

1. Change your AT-TLS policy to specify a client authentication level of SAF Check (more on AT-TLS policy shortly)
2. Map every server certificate to a RACF user ID on every other system:
  - You can use the host ID Mapping certificate extension for this purpose if it is available from your external CA or if you use PKI Services.
  - If not available, you can add a certificate name filter on each node for every other node. Or you can add each server certificate to the RACF database of every other node. Using certificate name filters requires less administrative effort.
3. Grant the mapped user ID READ access to IRR.RRSF.CONNECT in the RRSFDATA class on each system

**Remember:** Some security policies require that all certificates must be obtained from an external CA. In this case, additional controls can be established. These controls are needed because you might not be in control of what certificates get signed by the CA certificate used to sign your RRSF server certificates.

- ▶ Such certificates would be able to assert that they are RRSF servers without the additional controls.
- ▶ These controls are entirely your responsibility.
- ▶ RRSF cannot tell from where you obtained your CA certificate.

After your certificates in place, the system uses your AT-TLS policy to discover the name of the key ring. TCP/IP, by using System SSL, uses the policy to run the TLS handshake when one RRSF attempts to connect to another.

A working sample is provided in SYS1.SAMPLIB(IRRSTRSF), or can be created by using the procedure outlined in 3.2, “Setting up the PAGENT-started task” on page 28. It just needs to be enabled and installed into the Policy Agent.

RRSF is a “TLS-aware application”. RRSF refuses to connect or accept connections unless an adequate policy is in effect.

### 3.4.3 Using the same, self-signed certificate for all RRSF nodes

In this approach, implement an RRSF trust policy for TCP/IP node connections. Create one self-signed certificate that you send to each TCP/IP node in your RRSF network.

For each node, create a key ring to hold only the node's server certificate. For a multisystem node, a single server certificate and key ring are shared among the member systems.

Perform the following steps to create one self-signed certificate and send an exported copy of it to each RRSF node:

1. Choose a node in your RRSF network and create a self-signed certificate as shown in Example 3-9.

---

*Example 3-9 Create a self-signed certificate*

---

```
RACDCERT GENCERT
WITHLABEL('RRSF Server')
SUBJECTSDN(CN('RACF Address Space') O('YOURORG') C('US'))
KEYUSAGE(HANDSHAKE)
NOTAFTER(2016-09-01)
```

---

Do not specify the PKDS, PCICC, or ICSF options. The private key in this step must be stored in the RACF database so that it can be exported with the certificate in Step 2.

2. Export the certificate as a PKCS #12 package as shown in Example 3-10.

---

*Example 3-10 Export the certificate*

---

```
RACDCERT EXPORT(LABEL('RRSF Server'))
DSN(RACF.PK12DER)
FORMAT(PKCS12DER)
PASSWORD('The circus is coming 2 town.')
```

---

3. Using FTP in binary mode, transfer the export package from the local node to a data set on each remote TCP/IP node in your RRSF network. For a multisystem node, transfer the package to only one of the member systems.
4. (Optional) On the local node, move the private key from the RACF database to the ICSF PKA key data set (PKDS), if available. The private key then has hardware protection.

If your installation controls resources in the CSFSERV and CSFKEYS classes, ensure that the user ID of the RACF subsystem has sufficient authority. The user ID must have authority even if it has the TRUSTED attribute.

- a. Delete the self-signed certificate you created in Step 1:

```
RACDCERT DELETE(LABEL('RRSF Server'))
```

- b. Read the self-signed certificate by using the export package you created in Step 2. Store the private key in the ICSF PKDS as shown in Example 3-11.

*Example 3-11 Add the self-signed certificate*

---

```
RACDCERT ADD(RACF.PK12DER) ID(RACF)
TRUST
WITHLABEL('RRSF Server')
PASSWORD('The circus is coming 2 town.')
PKDS(RRSFserverkey)
```

---

5. (Optional) Delete the data set containing the export package because it is no longer needed.

If you opted to leave the private key in the RACF database, you can delete the export package. If you want to add a TCP/IP node in the future, you can reuse the same RRSF server certificate. Export it as you did in Step 2, then transfer it to the new node.

If you moved the private key to the ICSF PKDS, do not delete the export package when you want to use the same RRSF server certificate with any new TCP/IP node. If you delete the export package, you must create and distribute a new self-signed server certificate (with a different distinguished name) for a new TCP/IP node.

6. On the local node, create a RACF key ring for RRSF and add the server certificate to the ring.

- a. Create the RRSF key ring by using this command:

```
RACDCERT ID(RACF) ADDRING(IRR.RRSF.KEYRING)
```

Specify the key ring name provided by the programmer. The key ring name shown in the example is IRR.RRSF.KEYRING. This name matches the default name in the sample AT-TLS policy provided in the IRRSRRSF member of SYS1.SAMPLIB.

- b. Connect the server certificate to the key ring:

```
RACDCERT ID(RACF) CONNECT(LABEL('RRSF Server')
RING(IRR.RRSF.KEYRING)
DEFAULT
USAGE(PERSONAL))
```

- c. Permit the user ID of RACF subsystem to access the key ring:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RACF) ACCESS(READ)
```

**Attention:** Do not skip this step even when the user ID of RACF subsystem has the TRUSTED or PRIVILEGED attribute on your system.

- d. Activate the profile change in the FACILITY class, as follows.

- If the FACILITY class is not already active, activate and RACLIST it:

```
SETOPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it:

```
SETOPTS RACLIST(FACILITY) REFRESH
```

7. On each remote RRSF node, add the self-signed certificate by using the export package you transferred in Step 3. These are the same steps you performed for the local node in Steps 4b and 5.

- a. Add the certificate and store the private key in the ICSF PKDS, if available.

If your installation controls resources in the CSFSERV and CSFKEYS classes on the remote node, ensure that the user ID of the RACF subsystem has sufficient authority. This authority is needed even if the user ID has the TRUSTED attribute.

```
RACDCERT ADD(RACF.PK12DER) ID(RACF)
TRUST
WITHLABEL('RRSF Server')
PASSWORD('The circus is coming 2 town.')
PKDS(RRSFserverkey)
```

- b. (Optional) Delete the data set containing the export package because it is no longer needed.

8. On each remote RRSF node, create a RACF key ring for RRSF and add the server certificate to the ring. These are the same steps you performed for the local node in Step 6.

- a. Create the RRSF key ring:

```
RACDCERT ID(RACF) ADDRING(IRR.RRSF.KEYRING)
```

Specify the key ring name provided by the programmer. The key ring name shown in the examples is IRR.RRSF.KEYRING. This matches the default name in the sample AT-TLS policy provided in the IRRSRRSF member of SYS1.SAMPLIB.

- b. Connect the server certificate to the key ring:

```
RACDCERT ID(RACF) CONNECT(LABEL('RRSF Server')
RING(IRR.RRSF.KEYRING)
DEFAULT
USAGE(PERSONAL))
```

- c. Permit the user ID of RACF subsystem to access the key ring:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RACF) ACCESS(READ)
```

**Attention:** Do not skip this step even when the user ID of RACF subsystem has the TRUSTED or PRIVILEGED attribute on your system.

9. Activate the profile change in the FACILITY class:

- If the FACILITY class is not already active, activate and RACLIST it:

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

When you are finished, you have a key ring for each TCP/IP node and added its signed server certificate to the ring. You have now implemented an RRSF trust policy for TCP/IP node connections.

### 3.4.4 Using an internal CA to sign a server certificate for each RRSF node

This is the option used in the example environment to define the trust policy.



In this approach, implement an RRSF trust policy for TCP/IP node connections by creating an internal CA certificate. Use this CA to sign an individual server certificate for each TCP/IP node connection in your RRSF network.

**Remember:** Be sure to use this CA certificate to sign only server certificates for RRSF nodes.

For each node, create a key ring to hold the node's server certificate and the signing CA certificate. For a multisystem node, a single server certificate and key ring are shared among the member systems.

**Consideration:** Updates can be made to digital certificate information in the RACF database by the RACDCERT, RDEFINE, RALTER, and RDELETE commands. The updates can also be made by applications that start the R\_datalib and initACEE callable services.

However, the RACDCERT command itself is not eligible for command direction. It cannot be used with the AT or ONLY keyword. It can be run on remote nodes by using other methods. These updates can affect profiles in DIGTCERT, DIGTCRIT, DIGTNMAP, DIGTRING, and USER classes. Updates to profiles in these classes are eligible for automatic direction of application updates. Therefore, you must ensure consistent propagation across these classes.

Before defining your certificates, run the RACF commands shown in Example 3-12 on all nodes to prepare RACF commands.

*Example 3-12 Activate DIGTCERT class*

---

```
SETR CLASSACT(DIGTCERT)
SETR RACLIST(DIGTCERT)
```

---

Perform these steps to create a CA certificate that you use to sign an individual server certificate for each TCP/IP node connection in your RRSF network.

1. Choose a system in your RRSF network as the CA host system. This system hosts your new signing CA certificate. The example installation uses system SC74 (RSSF node PLEX75) to become the CA host system.
  - a. On the CA host system, create the RRSF CA certificate as shown in Figure 3-16.

```
RACDCERT CERTAUTH -
  DELETE -
  (LABEL('RRSF CERTAUTH CERT'))
RACDCERT CERTAUTH GENCERT -
  SUBJECTSDN( O('I.B.M Corporation') -
              CN('certauth.pok.ibm.com') -
              C('US')) -
  NOTBEFORE(DATE(2012-01-01)) -
  NOTAFTER(DATE(2019-12-31)) -
  KEYUSAGE(CERTSIGN) -
  SIZE(1024) -
  WITHLABEL('RRSF CERTAUTH CERT')
```

*Figure 3-16 GENCERT command for CA cert*

- b. List the new RRSF CA certificate and record the issuer distinguished name and serial number as shown in Example 3-13.

*Example 3-13 RACDCERT LIST*

---

```
RACDCERT CERTAUTH LIST(LABEL('RRSF CERTAUTH CERT'))
```

Digital certificate information for CERTAUTH:

```
Label: RRSF CERTAUTH CERT
Certificate ID: 2QiJmZmDhZmjgdnZ4sZAw8XZ48Hk48hAw8XZ40BA
Status: TRUST
Start Date: 2012/01/01 00:00:00
End Date: 2018/05/14 16:30:00
Serial Number:
    >10<
Issuer's Name:
    >CN=certauth.pok.ibm.com.0=I.B.M Corporation.C=US<
Subject's Name:
    >CN=certauth.pok.ibm.com.0=I.B.M Corporation.C=US<
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 1024
Private Key: YES
Ring Associations:
    Ring Owner: RACF
    Ring:
        >IRR.RRSF.KEYRING<
```

- 
2. On the CA host system, create the server certificate for the local RRSF node. Associate it with the user ID of the RACF subsystem as shown in Figure 3-17.

```
RACDCERT ID(RACF) DELETE(LABEL('RRSF SERVER CERTIFICATE'))

RACDCERT ID(RACF) GENCERT SUBJECT (CN('RRSF SERVER')      +
O('IBM') OU('IBM ITSO POK')                               +
C('US')) SIZE(1024) WITHLABEL('RRSF SERVER CERTIFICATE')  +
NOTBEFORE(DATE(2012-01-01))                                +
NOTAFTER(DATE(2019-12-30))                                  +
SIGNWITH(CERTAUTH LABEL('RRSF CERTAUTH CERT'))

RACDCERT ID(RACF) LIST(LABEL('RRSF SERVER CERTIFICATE'))
```

*Figure 3-17 Create Server certificate*

3. On the local node, create a RACF key ring for use with RRSF. Add both the RRSF CA certificate and the server certificate to the ring.
  - a. Create the RRSF key ring as shown in Figure 3-18. Specify the key ring name as defined in the AT/TLS policy (IRR.RRSF.KEYRING).

```
RACDCERT ID(RACF) DELRING(IRR.RRSF.KEYRING)
RACDCERT ID(RACF) ADDRING(IRR.RRSF.KEYRING)
RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)
RACDCERT ID(RACF) LISTRING(*)
```

Figure 3-18 Create the key ring for RRSF usage

- b. Connect the server certificate to the key ring as shown in Figure 3-19 on page 53.

```
/* -PURPOSE -CONNECT RRSF SERVER CERT TO RRSF SERVER KEYRING */
RACDCERT ID(RACF) CONNECT +
( ID(RACF) LABEL('RRSF SERVER CERTIFICATE') +
RING(IRR.RRSF.KEYRING) +
DEFAULT +
USAGE(PERSONAL) )
```

Figure 3-19 Connect server cert to RRSF keyring

- c. Connect the RRSF CA certificate to the key ring as shown in Figure 3-20.

```
/* -PURPOSE -CONNECT RACF CA CERT TO RRSF SERVER KEYRING */
RACDCERT ID(RACF) CONNECT +
( CERTAUTH LABEL('RRSF CERTAUTH CERT') +
USAGE(CERTAUTH) +
RING(IRR.RRSF.KEYRING) )
SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH
```

Figure 3-20 Connect CA cert to RRSF keyring

- d. Permit the RACF subsystem to access the key ring:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RACF) ACCESS(READ)
```

**Attention:** Do not skip this step even when the user ID of RACF subsystem has the TRUSTED or PRIVILEGED attribute on your system.

- e. Activate the profile change in the FACILITY class:
    - If the FACILITY class is not already active, activate and RACLIST it:
 

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```
    - If the FACILITY class is already active and RACLISTed, refresh it:
 

```
SETROPTS RACLIST(FACILITY) REFRESH
```

4. On the local node, list the certificate and KEYRING of the RRSF server:

```
RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)
RACDCERT ID(RACF) LIST
RACDCERT ID(RACF) LIST(LABEL('RRSF SERVER CERTIFICATE'))
```

```
RACDCERT CERTAUTH LIST
RACDCERT CERTAUTH LIST(LABEL('RRSF CERTAUTH CERT'))
```

Example 3-14 shows the listing for the RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING) command in the example environment.

*Example 3-14 RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)*

---

```
RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)
```

---

Digital ring information for user RACF:

Ring:  
 >IRR.RRSF.KEYRING<

Certificate Label Name	Cert Owner	USAGE	DEFAULT
RRSF CERTAUTH CERT	CERTAUTH	CERTAUTH	NO
RRSF SERVER CERTIFICATE	ID(RACF)	PERSONAL	YES

---

Example 3-15 shows the listing for the RACDCERT ID(RACF) LIST command in the example environment.

*Example 3-15 RACDCERT ID(RACF) LIST*

---

```
RACDCERT ID(RACF) LIST
```

---

Digital certificate information for user RACF:

```
Label: RRSF SERVER CERTIFICATE
Certificate ID: 2QTZwcPG2dnixkDixdn1xd1Aw8XZ48nGycPB48VA
Status: TRUST
Start Date: 2012/01/01 00:00:00
End Date: 2018/05/14 16:30:00
Serial Number:
  >11<
Issuer's Name:
  >CN=certauth.pok.ibm.com.O=I.B.M Corporation.C=US<
Subject's Name:
  >CN=RRSF SERVER.OU=IBM ITSO POK.O=IBM.C=US<
Key Type: RSA
Key Size: 1024
Private Key: YES
Ring Associations:
  Ring Owner: RACF
  Ring:
    >IRR.RRSF.KEYRING<
```

---

Example 3-16 shows the listing for the RACDCERT ID(RACF) LIST(LABEL('RRSF SERVER CERTIFICATE')) command in the example environment.

*Example 3-16 RACDCERT ID(RACF) LIST(LABEL('RRSF SERVER CERTIFICATE'))*

---

```
RACDCERT ID(RACF) LIST(LABEL('RRSF SERVER CERTIFICATE'))
```

---

Digital certificate information for user RACF:

```
Label: RRSF SERVER CERTIFICATE
Certificate ID: 2QTZwcPG2dnixkDixdn1xd1Aw8XZ48nGycPB48VA
Status: TRUST
Start Date: 2012/01/01 00:00:00
```

```
End Date: 2018/05/14 16:30:00
Serial Number:
>11<
Issuer's Name:
>CN=certauth.pok.ibm.com.O=I.B.M Corporation.C=US<
Subject's Name:
>CN=RRSF SERVER.OU=IBM ITSO POK.O=IBM.C=US<
Key Type: RSA
Key Size: 1024
Private Key: YES
Ring Associations:
  Ring Owner: RACF
  Ring:
>IRR.RRSF.KEYRING<
```

---

Example 3-17 shows the listing for the RACDCERT CERTAUTH LIST command in the example environment.

*Example 3-17 RACDCERT CERTAUTH LIST*

---

```
RACDCERT CERTAUTH LIST

Label: RRSF CERTAUTH CERT
Certificate ID: 2QiJmZmDhZmjgdnZ4sZAw8XZ48Hk48hAw8XZ40BA
Status: TRUST
Start Date: 2012/01/01 00:00:00
End Date: 2018/05/14 16:30:00
Serial Number:
>10<
Issuer's Name:
>CN=certauth.pok.ibm.com.O=I.B.M Corporation.C=US<
Subject's Name:
>CN=certauth.pok.ibm.com.O=I.B.M Corporation.C=US<
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 1024
Private Key: YES
Ring Associations:
  Ring Owner: RACF
  Ring:
>IRR.RRSF.KEYRING<
```

---

Example 3-18 shows the listing for the RACDCERT CERTAUTH LIST(LABEL('RRSF CERTAUTH CERT')) command in the example environment.

*Example 3-18 RACDCERT CERTAUTH LIST(LABEL('RRSF CERTAUTH CERT'))*

---

```
RACDCERT CERTAUTH LIST(LABEL('RRSF CERTAUTH CERT'))

Digital certificate information for CERTAUTH:

Label: RRSF CERTAUTH CERT
Certificate ID: 2QiJmZmDhZmjgdnZ4sZAw8XZ48Hk48hAw8XZ40BA
Status: TRUST
Start Date: 2012/01/01 00:00:00
End Date: 2018/05/14 16:30:00
Serial Number:
>10<
Issuer's Name:
>CN=certauth.pok.ibm.com.O=I.B.M Corporation.C=US<
Subject's Name:
```

```

>CN=certauth.pok.ibm.com.O=I.B.M Corporation.C=US<
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 1024
Private Key: YES
Ring Associations:
  Ring Owner: RACF
  Ring:
>IRR.RRSF.KEYRING<

```

---

5. On each remote node, repeat the following steps:
  - a. On a remote RRSF node, create a certificate request for a server certificate for this remote node. For a multisystem node, create the request on only one of the member systems.
    - i. Create a self-signed placeholder certificate as shown in Figure 3-21.

```

/**      -PURPOSE -USE THIS ON THE REMOTE NODES ONLY !
/**
/**      STEP 1      CREATE SELF SIGNED PLACEHOLDER CERTIFICATE
/**
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *

  RACDCERT ID(RACF) DELETE(LABEL('RRSF SERVER CERTIFICATE'))

  RACDCERT ID(RACF) GENCERT SUBJECT (CN('RRSF SERVER PLX8')
  O('IBM') OU('IBM ITSO POK')
  C('US')) SIZE(1024) WITHLABEL('RRSF SERVER CERTIFICATE')
  NOTBEFORE(DATE(2012-01-01))
  NOTAFTER(DATE(2019-12-30))
  KEYUSAGE(HANDSHAKE)

  RACDCERT ID(RACF) LIST(LABEL('RRSF SERVER CERTIFICATE'))
/*

```

Figure 3-21 Create self-signed placeholder certificate

- ii. Create a certificate request based on the self-signed certificate you created as shown in Figure 3-22.

```

/*          -PURPOSE -USE THIS ON THE REMOTE NODES ONLY !
/* STEP 2    CREATE A CERTIFICATE REQUEST BASED ON THE SELF SIGNED
/*          CREATED IN TSEP 1
/*
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
        RACDCERT ID(RACF) GENREQ(LABEL('RRSF SERVER CERTIFICATE')) -
        DSN('RRSF.PLX8.REQ') FORMAT(CERTB64)
/*

```

Figure 3-22 Create a certificate request

A base64-encoded certificate request is stored in the specified data set on the remote node.

- iii. Transfer the certificate request from the data set on the remote node to a data set on the CA host system. Because this is a base64-encoded request, transfer the request as text to ensure that the ASCII translation from EBCDIC takes place. To do so, use ASCII (not binary) FTP.

You can also copy the text of the request from the data set on the remote node shown in Figure 3-23. Then paste it into an empty data set with identical attributes on the CA host system. Be sure to include the BEGIN and END lines.

The example in the next step specifies a data set on the CA host system with the same name as the data set on the remote node.

```

BROWSE      RRSF.PLX8.REQ                               Line 00000000
Command ==>                                           Scro
***** Top of Data *****
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBzTCCATYCAQAwTTElMAkGA1UEBhMCVVMxDDAKBgNVBAoTA01CTTEVMBMGA1UE
CxMMSUJNIE1UU08gUE9LMRkwFwYDVQQDExBSU1NGIFNFU1ZFUiBQTFg4MIGfMAOG
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDIxG0Ke93D7ET6AZ0yhUeImotoBdi1Wx6A
Cu0u1S0sALov0UZI2cRrtdF3HH3jV/8KPEmHkhs3SN0NWA/kIBeBrHrzTL64NqZ1
zQr1NoacXUIYHImIwVJswfBU2s6aejtWGjRFTyo88aPNKYz5bos9nU2mZ2ZB4to0
mkAN2Yz8mQIDAQABoEAwPgYJKoZIhvcNAQkOMTEwLzAdBgNVHQ4EFgQUZbVh8IkY
mQoKJpTkj9XVZfFbVdAwDgYDVROPAQH/BAQDAgWgMA0GCSqGSIb3DQEBAQUAA4GB
ABiMWadpIxfDFnL9a9/yMdJ0FMbTDL7MFtTKpxVxrhaN07AdJHqYHHuya23yczyV
dhsZckHmEqz0ADsUFqd+xb9cb5PIKbL+XgM/HmR1NDEA+K1ussPmNv3hTf5pdHY6
F/YOCYEhJRy9TiMo1UjJZ3VGjIDoPvKDFToNmV00Y2kA
-----END NEW CERTIFICATE REQUEST-----

```

Figure 3-23 Contents of certificate request data set

- b. On the CA host system, create the server certificate for the remote node and sign it with the RRSF CA certificate you created in Step 1a (Figure 3-24). The certificate created in this step is for the remote node, and is not used on the host system. Therefore, you can associate the certificate with any user ID. You can choose to associate it with the same user ID as the certificate used by the local RRSF node (created in Step 2). To do so, specify a different certificate label by using the WITHLABEL operand to avoid an error in this step.

```
/*          -PURPOSE -USE THIS ON THE CA HOST SYSTEM (SC74)
/*
/* STEP 3      CREATE THE RRSF SERVER CERTIFICATE FOR THE REMOTE NODE
/*              AND SIGN IT WITH THE RRSF CA CERT
/*
/*STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
        RACDCERT ID(RACF) GENCERT('RRSF.PLX8.REQ') -
        WITHLABEL('PLX8 RRSF SERVER CERTIFICATE') -
        SIGNWITH(CERTAUTH LABEL('RRSF CERTAUTH CERT'))
/*
```

Figure 3-24 Fulfill certificate request for remote node

- c. On the CA host system, export the newly signed certificate to a data set (Figure 3-25). You can use the same data set that you used to store the certificate request in Step 3b.

```
/*          -PURPOSE -USE THIS ON THE CA HOST      (SC74)
/*
/* STEP 4      EXPORT THE NEWLY SIGNED CERTIFICATE TO A data set
/*
/*
/*STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
        RACDCERT ID(RACF) -
        EXPORT(LABEL('PLX8 RRSF SERVER CERTIFICATE')) -
        DSN('RRSF.PLX8.CERT') FORMAT(PKCS7B64)

/* NOW DELETE THIS CERTIFICATE AS NO LONGER NEEDED */

        RACDCERT ID(RACF) -
        DELETE(LABEL('PLX8 RRSF SERVER CERTIFICATE'))
/*
```

Figure 3-25 Export the newly created certificate

RACF stores the resulting export package in the specified data set in the PKCS #7 format. The package contains both the new server certificate and the RRSF CA certificate that signed it.



Optionally, delete the server certificate on the CA host system that you created in Step 5 because it is no longer needed:

```
RACDCERT ID(RACF) -  
DELETE(LABEL('PLX8 RRSF SERVER CERTIFICATE'))
```

- d. Transfer the export package from the data set on the CA host system to a data set on the remote node.

Use ASCII (not binary) FTP. You can also copy the text of the certificates from the data set on the CA host system. Then paste it into an empty data set with identical attributes on the remote node. Be sure to include the BEGIN and END lines.

Optionally, now delete the data set on the CA host system because it is no longer needed.

For a multisystem node, transfer the export package to only one of the member systems.

Figure 3-26 shows the contents of the exported certificate.

```
BROWSE      RRSF.PLX8.CERT                               Line 000000
Command ==>                                             Sc
***** Top of Data *****
-----BEGIN CERTIFICATE-----
MIIFXgYJKoZIhvcNAQcCoIIFTzCCBUSCAQExADALBgkqhkiG9w0BBwGgggUzMIIC
oDCCAgmgAwIBAgIBBDANBgkqhkiG9w0BAQUFADBIMQswCQYDVQQGEwJVUzEaMBGg
A1UEChMRSS5CLk0gQ29ycG9yYXRpb24xHTAbBgNVBAMTFGNlcnRhdXR0LnBvay5p
Ym0uY29tMB4XDTEyMDUxMDA0MDAwMFOXDTEzMDUxMTAzNTk1OVowTTElMAkGA1UE
BhMCVVMxDDAKBgNVBAoTA0lCTTEVMBMGA1UECxMMSUJNIElUU08gUE9LMRkwFwYD
VQDExBSU1NGIFNFU1ZFUlBQTFg4MIGfMA0GCsQsIb3DQEBAQUAA4GNADCBiQKB
gQDIxG0Ke93D7ET6AZ0yhUeImotoBdi1Wx6ACu0u1S0sALov0UZI2cRrtdF3HH3j
V/8KPEmHkhs3SNONWA/kIBeBrHrzTL64NqZ1zQr1NoacXUIYHImIwVJswfBU2s6a
ejtWGjRFTyo88aPNKYz5bos9nU2mZ2ZB4to0mkAN2Yz8mQIDAQABo4GUMIGRMD8G
CWCGSAGG+EIBDQyFjBHZW5lcmF0ZWQgYnkgdGhlIFNlY3VyaXR5IFNlcnZlciBm
b3Igei9PUyAoUkFDRikwDgYDVROPAQH/BAQDAgWgMBOGA1UdDgQWBBRltWHwiRiZ
CgomlOSP1dVl8VtVODAfBgNVHSMEGDAWgBT7WavHjhVzu1f5QdrCbIXdAYaf2TAN
BgkqhkiG9w0BAQUFAA0BgQCnfWGBbvsDxC5gXrcZiRrYj0+7UBMT9bQ/XXgsMYo1
PVz4Dt/kWAyMWQ7cvTN/Y0kwNKV1t53hCPArojMhcub0ai3SMgcadUkQVZxnHnJM
jg46Xs4AwsULw+us2bmwwK6Pu3QmlyQA0d8Yq1fnX+xXh+fHFE4YTg8XZoQz7NnG
FDCCAoswggH0oAMCAQICAQAwDQYJKoZIhvcNAQEFBQAQAwSDElMAkGA1UEBhMCVVMx
GjAYBgNVBAoTEUkuQi5NIENvcnBvcnF0aW9uMR0wGwYDVQDExRjZXJOYXV0aC5w
b2suaWJtLmNvbTAeFw0xMjAxMDUwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
b2suaWJtLmNvbTAeFw0xMjAxMDUwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
BgNVBAYTA1VTMR0wGAYDVQQKEwFJLkIuTSBDb3Jwb3JhdG1vbG1EdMBsGA1UEAxMU
Y2VydGF1dGgucG9rLm1ibS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgYOAMIGJAoGB
ALFJonX9+W00+ApGhS+Ga9xfcr04A1/k1502UHuHdnJThMHQAGULL1wHhwjX1+J
lyP9sFlJ3EVFVDPKmmR6mhpIERuKTznrL0zKEx0v6BppM0kXaTeRRyTU9US0IamY
7a192oD+UnMOJ6h8C9JY82rdurjKb1Q013RljZmhzcDJAgMBAAGjgYQwgYEwPwYJ
YIZIAyb4QgENBDIWMEd1bmVYXRlZCBieSB0aGUgU2VjdXJpdHkgU2VydMvYIGZv
ciB6L09TIClChSQUNGKTA0BgNVHQ8BAf8EBAMCAQYwDwYDVROTAQH/BAUwAwEB/zAd
BgNVHQ4EFgQU+1mrX44Vc7tX+UHa3GyMXQGgn9kwDQYJKoZIhvcNAQEFBQADgYEA
cR+7rGXz/i30v+8Zcmwn86nvsiods0V+2Jcppt3XS2kBwWdv40Upk7CJ+AVbcPAS
SzwK1n3aYUTDtXh9RtarruXfDzuVfUUX0m35oy3xp961vBdPr97y76Nfw8FsboVp
qD3REoH84dhkLGpr6D+GIHXSjMU5C4LoIhCWGKABbVMxAA==
-----END CERTIFICATE-----
***** Bottom of Data *****
```

Figure 3-26 Contents of exported certificate

- e. On the remote node, add the newly signed certificate to replace the self-signed placeholder certificate you created in Step 5a.
- i. Add the RRSF CA certificate as shown in Figure 3-27.

```

/*          -PURPOSE -USE THIS ON THE REMOTE NODES (PLX8)
/*
/* STEP 5    ADD RRSF CA CERT + SERVER CERTIFICATE
/*
/*
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
    RACDCERT ID(RACF)
    ADD('RRSF.PLX8.CERT') TRUST -
    WITHLABEL('RRSF SERVER CERTIFICATE')

/*

```

*Figure 3-27 Add the certificates to the RACF database*

Both the server certificate and the RRSF CA certificate are added to the RACF data base on the remote node. The RRSF CA certificate is assigned a generated label. The following message is issued:

```

IRRD152I Root Certificate Authority not currently defined to RACF. Top
CERTAUTH certificate added with the TRUST status.

```

- ii. Discover the generated label of the RRSF CA certificate. List the RRSF CA certificate by specifying the issuer distinguished name and the serial number you recorded in Step 1b. Make note of the generated certificate label.

```

RACDCERT LIST(ISSUERSDN('CN=certauth.pok.ibm.com.O=I.B.M
Corporation.C=US') SERIALNUMBER(10)) CERTAUTH

```

If you did not record the issuer name and serial number in Step 1b, issue the RACDCERT LIST CERTAUTH command as shown in Figure 3-28.

```
/*          -PURPOSE -USE THIS ON THE REMOTE NODES (SC76)
/*
/* STEP 6    LIST CERTAUTH TO GET GENERATED LABEL
/*
/*
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT +
LIST(ISSUERSDN('CN=certauth.pok.ibm.com.0=I.B.M Corporation.C=US') +
SERIALNUMBER(10)) CERTAUTH
RACDCERT LIST CERTAUTH
/*
```

Figure 3-28 RACDCERT LIST

Review the list of CA certificates shown in Example 3-19. Locate the RRSF CA certificate by the subject distinguished name, for example CN=certauth.pok.ibm.com.0=I.B.M Corporation.C=US, and make note of the label.

*Example 3-19 RACDCERT LIST output*

```
READY
RACDCERT LIST(ISSUERSDN('CN=certauth.pok.ibm.com.0=I.B.M
Corporation.C=US') SERIALNUMBER(10)) CERTAUTH

Digital certificate information for CERTAUTH:

Label: LABEL00000001
Certificate ID: 2QiJmZmDhZmjgdPBwsXT8PDw8PDw8PFA
Status: NOTRUST
Start Date: 2012/01/01 00:00:00
End Date: 2012/05/13 23:59:59
Serial Number:
>08<
Issuer's Name:
>CN=certauth.pok.ibm.com.0=I.B.M Corporation.C=US<
Subject's Name:
>CN=certauth.pok.ibm.com.0=I.B.M Corporation.C=US<
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 1024
Private Key: NO
Ring Associations:
*** No rings associated ***
```

- iii. (Optional) Modify the label of the RRSF CA certificate as shown in Figure 3-29.

```
/*          -PURPOSE -USE THIS ON THE REMOTE NODES (SC76)
/*
/* STEP 7      UPDATE CERTAUTH GENERATED LABEL (OPTIONAL)
/*
/*
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
        RACDCERT ALTER(LABEL('LABEL00000001')) CERTAUTH -
        NEWLABEL('RRSF CERTAUTH CERT')

/*
```

Figure 3-29 Modify label

- f. On each RRSF node, create a RACF key ring for use with RRSF. Add both the RRSF CA certificate and the server certificate to the ring.
- i. Create the RRSF key ring as shown in Figure 3-30.

```
/*          -PURPOSE -USE THIS ON THE REMOTE NODES (SC76)
/*
/* STEP 8      create rrsf server keyring
/*
/*
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *

        RACDCERT ID(RACF) DELRING(IRR.RRSF.KEYRING)
        RACDCERT ID(RACF) ADDRING(IRR.RRSF.KEYRING)
        RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)
        RACDCERT ID(RACF) LISTRING(*)

/*
```

Figure 3-30 Create the keyring

- ii. Connect the RRSF CA certificate to the key ring as shown in Figure 3-31.

```

/* STEP 9    connect rrsf server cert to keyring
/*
/*
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *

/*          -PURPOSE -CONNECT RRSF SERVER CERT TO RRSF SERVER KEYRING */
RACDCERT ID(RACF) CONNECT                                +
( ID(RACF) LABEL('RRSF SERVER CERTIFICATE')              +
RING(IRR.RRSF.KEYRING)                                   +
DEFAULT                                                    +
USAGE(PERSONAL)      )

SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH
/*

```

Figure 3-31 Connect RRSF server certificate to keyring

- iii. Connect the server certificate to the key ring as shown in Figure 3-32.

```

/* STEP 10   connect CA cert to keyring
/*
/*
//STEP01 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RACDCERT ID(RACF) CONNECT                                +
( CERTAUTH LABEL('RRSF CERTAUTH CERT')                  +
USAGE(CERTAUTH)                                           +
RING(IRR.RRSF.KEYRING)  )
/*

```

Figure 3-32 Connect CA cert to keyring

- iv. Permit the RACF subsystem to access the key ring:

```

RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RACF) ACCESS(READ)
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
SETROPTS RACLIST(FACILITY) REFRESH

```

**Attention:** Do not skip this step even when the user ID of RACF subsystem has the TRUSTED or PRIVILEGED attribute on your system.

- v. Activate the profile change in the FACILITY class:

If the FACILITY class is not already active, activate and RACLIST it:

```

SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)

```

If the FACILITY class is already active and RACLISTed, refresh it:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

When you are finished, you have created two key rings, one for the CA host system and one remote TCP/IP node in your RRSF network. You have also added the signed server certificate for each node and its signing CA certificate to each ring.

**Explanation:** For each additional RRSF node that you want to allow to communicate using TCP/IP, repeat Steps 5a through 5e. When you are finished, you have implemented an RRSF trust policy for TCP/IP node connections.

6. List the certificates and keyring of your RRSF server in each node as shown in Example 3-20.

*Example 3-20 Listing the keyring*

---

```
READY
```

```
RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)
```

Digital ring information for user RACF:

Ring:

```
>IRR.RRSF.KEYRING<
```

Certificate Label Name	Cert Owner	USAGE	DEFAULT
-----	-----	-----	-----
RRSF CERTAUTH CERT	CERTAUTH	CERTAUTH	NO
RRSF SERVER CERTIFICATE	ID(RACF)	PERSONAL	YES

---

7. List the certificates of your RRSF server in each node as shown in Example 3-21.

*Example 3-21 RACDCERT ID(RACF) LIST*

---

```
RACDCERT ID(RACF) LIST
```

```
Label: RRSF SERVER CERTIFICATE
```

```
Certificate ID: 2QTZwcPG2dnixkDixdn1xd1Aw8XZ48nGycPB48VA
```

```
Status: TRUST
```

```
Start Date: 2012/05/10 00:00:00
```

```
End Date: 2013/05/10 23:59:59
```

```
Serial Number:
```

```
>04<
```

```
Issuer's Name:
```

```
>CN=certauth.pok.ibm.com.O=I.B.M Corporation.C=US<
```

```
Subject's Name:
```

```
>CN=RRSF SERVER PLX8.OU=IBM ITSO POK.O=IBM.C=US<
```

```
Key Usage: HANDSHAKE
```

```
Key Type: RSA
```

```
Key Size: 1024
```

```
Private Key: YES
```

```
Ring Associations:
```

```
Ring Owner: RACF
```

```
Ring:
```

```
>IRR.RRSF.KEYRING<
```

---

### 3.4.5 Using an external CA to sign a server certificate for each RRSF node

When you must use an external CA, use one of these approaches.

- ▶ Request only an intermediate CA certificate from the external CA, and use the certificate to sign only an individual server certificate for each RRSF node. This approach is similar to the approach described in 3.4.4, “Using an internal CA to sign a server certificate for each RRSF node” on page 50.
- ▶ Use an external CA to generate a request for a server certificate for each RRSF node. It then sends those requests to the external CA. When returned, add each server certificate and CA certificate to the key ring for each RRSF node.

If the CA is well known, the CA certificate might already be in the RACF database. If so, connect it to the key ring of each node and mark it as trusted.

**Remember:** Any server that presents a certificate signed by this CA certificate is accepted as a valid RACF node.

The following guidelines for increased security apply to the second approach when you lack exclusive use of the signing CA certificate:

- Ensure that the SAF Check level of client authentication is specified in the Communication Server AT-TLS policy for RRSF.
- Using zOSMF configuration assistant, indicate the ‘Client authentication handling’ level of client authentication in the advanced settings.

These settings specify whether to run the SSL handshake for a server that requires client authentication. If so, it specifies which of these levels of client authentication to perform:

- Pass-through: Bypasses client certificate validation.
- Full: Runs client certificate validation if the client presents a certificate.
- Required: Requires the client to present a certificate and runs client certificate validation. This setting is the default if the server requires client authentication.



- **SAF Check:** Requires the client to present a certificate, runs client certificate validation, and requires the client certificate to have an associated user ID defined to the security product (Figure 3-33).

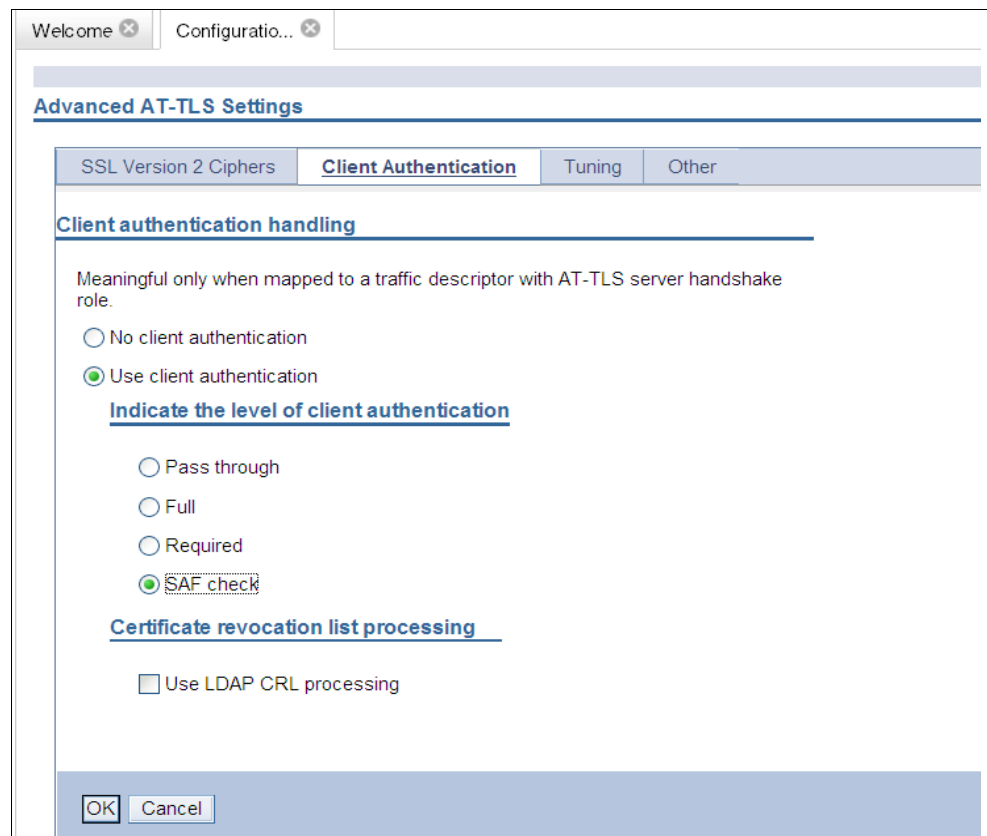


Figure 3-33 SAF Check

This level requires that you map each RRSF server certificate to a RACF user ID. You can use the `hostIdMapping` certificate extension for this purpose if it is available from your external CA or if you use PKI Services.

You can add a certificate name filter on each node for every other node or add each server certificate to the RACF database of every other node. Using certificate name filters requires less administrative effort.

If your external CA might issue other certificates by using the same CA certificate used to issue your server certificates, implement a one-to-one certificate filter for each server certificate. This configuration ensures that no other issued certificate matches your filter distinguished name (DN). If the external CA will not issue other certificates with the same DN, reduce administrative effort by implementing a single many-to-one filter on each node. Base this filter on the naming convention used for the distinguished names of your RRSF servers.

- Protect a resource named `IRR.RRSF.CONNECT` in the `RRSFDATA` class. Be sure to give the mapped user ID at least `READ` access to it even if the user ID has the `TRUSTED` or `PRIVILEGED` attribute. This `RRSFDATA` profile can also be used to create a RACF audit trail that logs the establishment of inbound RRSF connections to the local system.

The keyring must have the server certificate and the signing certificate.

Give the RACF subsystem user ID read access to IRR.DIGTCERT.LISTRING in the FACILITY Class. Connect the node server certificate as default

**Attention:** Never use the signing certificate to sign anything but RRSF certificates.

If connection fails due to keyring problems, make sure that the PAGENT reads the keyring again. Change the EnvironmentUserInstance value in the Policy Rule and then issue the **F PAGENT,update** command.

**Tip:** You can find the sample started task procedure for PAGENT in TCP/IP.SEZAINST(EZAPAGSP).

## 3.5 Updating TCP/IP to enable RRSF

Before you can use TCP/IP for RRSF, you need to update TCP/IP to enable the function and modify the TCP/IP Profile data set. Perform the following steps:

1. Enable AT-TLS
2. Define and protect the RRSF port 18136

### 3.5.1 Enabling AT-TLS

Enable AT-TLS in the TCP/IP profile by adding the TTLS parameter to the TCPCONFIG profile statement in each stack that is to support AT-TLS (Example 3-22).

*Example 3-22 Enabling AT-TLS in the TCP/IP profile*

---

```
TCPCONFIG TTLS
```

---

### 3.5.2 Defining and protecting RRSF port 18136

Define port 18136 and reserve it for the RRSF function. The example environment has the SAF key word **RRSF** specified to protect the port by using a RACF profile. The updated statements to define the RRSF port in the TCP/IP profile are shown in Example 3-23.

*Example 3-23 Define the RRSF port*

---

```
;;-----  
;; RRSF port should be reserved. The default port value is 18136.  
PORT  
18136 TCP * SAF RRSF ; RRSF listener port  
;;-----
```

---

Define a generic profile EZB.PORTACCESS.\*.\*.RRSF in class SERVAUTH to protect the RRSF TCP/IP port, and give the RACF user ID access to it.

**Attention:** The SAF keyword you specify in the PORT statement of the TCP/IP Profile must match the lowest level qualifier of the RACF profile. In the example environment, this keyword is **RRSF**.

## 3.6 Updating necessary RACF profiles

Normally, the RACF started task has automatic access to RACF-protected resources because it runs with the TRUSTED attribute. But the RRSF tasks that run TCP/IP communication run under a task level security environment (ACEE) without the TRUSTED or PRIVILEGED attributes. As a result, the RACF subsystem user ID needs to be permitted to any protected resources it is accessing by using these steps:

1. Add OMVS segment to RACF subsystem user ID
2. Enable CSFSERV resources
3. Protect the TCP/IP stack

### 3.6.1 Adding an OMVS segment to RACF subsystem user ID

Use of the TCP protocol requires the use of sockets, which requires UNIX System Services:

- ▶ Assign an OMVS segment with a UID to the RACF subsystem user ID by using the ALTUSER command.
- ▶ Assign an OMVS segment with a GID to its default group by using the ALTGROUP command.
- ▶ Assign the OMVS segments, and make the local node OPERATIVE again.

You do not have to restart the RACF subsystem. With RRSF, if you attempt to establish the TCP listener without performing the UNIX setup, you get an error message.

After you assign the UID/GID, establish the listener again.

Example 3-24 shows the sample JCL used to assign an OMVS segment to the RRSF subsystem address Space (RACF). The example uses 18136 as the UID, which makes it easy to remember because 18136 is also the default value for the listener port.

**Remember:** The RACF UID does not need to be 0 or superuser.

*Example 3-24 Add OMVS segment*

---

```
//PRICHAR3 JOB (999,POK),'RRSF',CLASS=A,MSGLEVEL=(1,1),
// MSGCLASS=A,NOTIFY=&SYSUID
//STEPX EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
    ALU      RACF +
           OMVS(UID(18136) SHARED HOME('/'))
/*
```

---

If you forget to assign an OMVS segment with a valid UID to the RACF subsystem, starting the listener function generates the error messages shown in Figure 3-34.

```

IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
578
IRRJ000I (#) RACF RACF LOCAL NODE TRANSACTION PROGRAM STARTING UNDER
577
           user ID RACF GROUP STCGROUP.
ICH408I USER(RACF    ) GROUP(STCGROUP) NAME(#####) 580
CL(PROCESS )
OMVS SEGMENT NOT DEFINED
IRRB023I (#) SYSTEM SERVICE BPX1ENV FAILED WITH RETURN CODE 581
           X'0000009C', REASON CODE X'0B0C00FB'.
IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER 579
IRROPT01 IS COMPLETE.
IRRC040I (#) RACF REMOTE SHARING CANNOT COMMUNICATE USING THE TCP 582
PROTOCOL. THE RACF SUBSYSTEM RUNNING UNDER user ID RACF
IS NOT RUNNING AS A Z/OS UNIX PROCESS.
IRRH004I (#) RACF SUBSYSTEM SET COMMAND HAS COMPLETED SUCCESSFULLY.

```

Figure 3-34 Starting RACF listener port without a valid UID.

### 3.6.2 Enabling CSFSERV resources

Use cryptographic hardware to store the server private key in ICSF in conjunction with AT-TLS security. To do so, define resources CSFDSV and CSFPKE in the CSFSERV class to protect the cryptographic services. Permit the user ID associated with RACF server to these resources as shown in Example 3-25.

Example 3-25 Enable CSFSERV resources

---

```

//RACFDEF EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
RDEFINE CSFSERV CSFDSV UACC(NONE)
RDEFINE CSFSERV CSFPKE UACC(NONE)
SETROPTS RACLIST(CSFSERV) REFRESH ID(RACF) ACCESS(READ)
PERMIT CSFPKE CLASS(CSFSERV) ID(RACF) ACCESS(READ)
SETROPTS RACLIST(CSFSERV) REFRESH
/*

```

---

The example environment does not use the cryptographic hardware.

### 3.6.3 Protecting the TCP/IP stack

The TCP/IP stack is protected by a SERVAUTH class profile with format  
EZB.STACKACCESS.sysname.tcpstackname

You can use the sample JCL shown in Example 3-26 to protect RACF and to give RACF user ID access to it.

*Example 3-26 Protect access to TCP/IP stack*

---

```
RDEFINE SERVAUTH EZB.STACKACCESS.*.* UACC(NONE)
PERMIT EZB.STACKACCESS.*.* CLASS(SERVAUTH) ID(RACF) ACCESS(READ)
SETROPTS RACLIST(SERVAUTH) REFRESH
```

---

## 3.7 Verifying the RRSF setup

This section addresses how to verify the RRSF setup. The example uses the three systems shown in Figure 3-35. The dotted lines in the diagram show the TCP/IP connections for RRSF between the systems. The local system SC74 in PLEX75 has only one-way connection to SC76. Only the main system SC75 has bidirectional connection to SC76.

For more information about constructing IRROPTxx members that contain TARGET statements to define RRSF nodes, see Chapter 4, “Converting APPC connections to TCP connections” on page 83. Although it addresses converting an existing APPC RRSF network to TCP/IP, the chapter addresses the construction and planning of the IRROPTxx member. The *z/OS Security Server RACF System Programmer’s Guide*, SA22-7681, contains numerous examples of how to define TARGET statements.

The example uses an IRROPTxx member that contains TARGET commands that are activated through the SET operator command. The TARGET commands can all be run first as operator commands. They can then be coded in IRROPTxx members to be used on subsequent starts of the RACF address space.

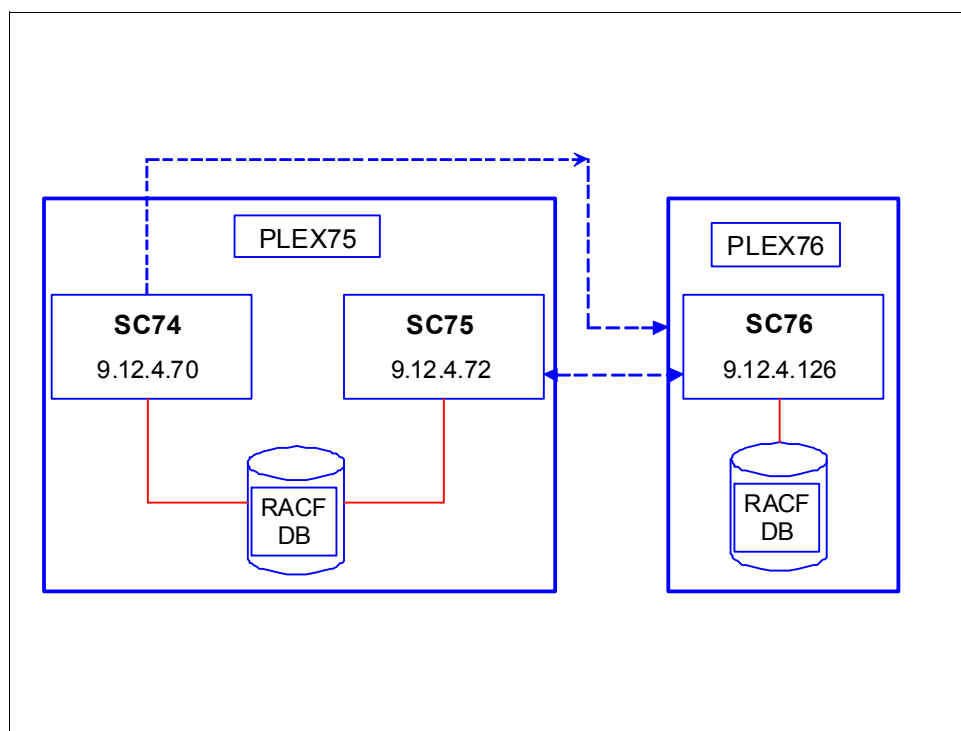


Figure 3-35 Verification of test configuration

Perform the following steps to verify the RRSF setup:

1. Activate and verify AT-TLS.
2. Stop RACF on all systems by issuing the command **#STOP**.
3. Start RACF on each system without defining any RRSF nodes by issuing **S RACF, SUB=MSTR** using a RACF parameter library.
4. Define the RRSF node on local main system SC75 in PLEX75 by running **#SET INCLUDE(75)**.
5. Define the RRSF node on local system SC74 in PLEX75 by running **#SET INCLUDE(75)**.
6. Define the RRSF node on local system SC76 in PLEX76 by running **#SET INCLUDE(76)**.
7. Define remote nodes to SC75 by running **#SET INCLUDE(76)** on SC75.
8. Define remote nodes to SC74 by running **#SET INCLUDE(76)** on SC74.
9. Define remote nodes to SC76 by running **#SET INCLUDE(75)** on SC76.
10. Display the RRSF status on every system by issuing **#TARGET LISTP** on each system.

### 3.7.1 Activating and verifying AT-TLS

Refresh PAGENT with the new AT-TLS policy, then verify that it is loaded and active. All three systems SC74, SC75, and SC76 contain the same copy of the AT-TLS policy.

1. Refresh PAGENT with the new AT-TLS policy.
2. Verify the details of the active policy.

#### Refreshing PAGENT with the new AT-TLS policy

Issue the command **F PAGENT, REFRESH**. The output is shown in Example 3-27.

*Example 3-27 Refreshing PAGENT with AT-TLS policy*

---

```
F PAGENT, REFRESH
EZZ8443I PAGENT MODIFY COMMAND ACCEPTED
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR TCPIP : TTLS
```

---

#### Verifying the details of the active policy

Start an OMVS session and issue the command **pasearch -t**. The output is shown in Example 3-28. The example includes only highlights because the output is three pages long.

*Example 3-28 pasearch -t command output*

---

```
RAMA:/u/rama: >pasearch -t
```

```
TCP/IP pasearch CS V1R13                      Image Name: TCPIP
Date:                                05/15/2012    Time:  14:25:17
TTLS Instance Id:                    1337097234

policyRule:                            Default_RRSF-Client~1
Rule Type:                            TTLS
Version:                              3              Status:          Active
.....

policyAction:                          cAct1~RRSF-Client
ActionType:                            TTLS Connection
Action Sequence:                       0
```

```

.....
TTLs Condition Summary:                               NegativeIndicator: Off
Local Address:
  FromAddr:      A11
  ToAddr:        A11
Remote Address:
  FromAddr:      A11
  ToAddr:        A11
LocalPortFrom:   1024      LocalPortTo:   65535
RemotePortFrom:  18136     RemotePortTo:  18136
JobName:
ServiceDirection: Outbound
.....
TTLs Action:      gAct1
Version:          3
Status:           Active
Scope:            Group
.....
TTLs Action:      eAct1~RRSF-Client
Version:          3
Status:           Active
Scope:            Environment
HandshakeRole:    Client
TTLsKeyringParms:
  Keyring:         IRR.RRSF.KEYRING
TTLsEnvironmentAdvancedParms:
TTLsEnvironmentAdvancedParms:
  SSLv2:           Off
  SSLv3:           On
  TLSv1:           On
  TLSv1.1:         On
  ApplicationControlled: Off
  HandshakeTimeout: 10
  ClientAuthType:  Required
.....
TTLs Action:      cAct1~RRSF-Client
Version:          3
Status:           Active
Scope:            Connection
HandshakeRole:    Client
.....
TTLsCipherParms:
  v3CipherSuites:
    35 TLS_RSA_WITH_AES_256_CBC_SHA
.....

policyRule:      Default_RRSF-Server~2
Rule Type:       TLS
Version:         3      Status:      Active
.....
No. Policy Action: 3
policyAction:      gAct1
ActionType:         TLS Group
Action Sequence:    0
policyAction:      eAct2~RRSF-Server

```

```

ActionType:          TTLS Environment
Action Sequence:     0
policyAction:        cAct2~RRSF-Server
ActionType:          TTLS Connection
Action Sequence:     0
.....
Local Address:
  FromAddr:          A11
  ToAddr:             A11
Remote Address:
  FromAddr:          A11
  ToAddr:             A11
LocalPortFrom:       18136      LocalPortTo:       18136
RemotePortFrom:      1024      RemotePortTo:      65535
.....
ServiceDirection:    Inbound
.....
TTLS Action:          eAct2~RRSF-Server
Version:              3
Status:               Active
Scope:                Environment
HandshakeRole:        ServerWithClientAuth
TTLSKeyringParms:
  Keyring:             IRR.RRSF.KEYRING
TTLSEnvironmentAdvancedParms:
.....
HandshakeTimeout:     10
  ClientAuthType:      Required
  ResetCipherTimer:    0
  TruncatedHMAC:        Off
  CertValidationMode:  Any
.....
TTLS Action:          cAct2~RRSF-Server
Version:              3
Status:               Active
Scope:                Connection
HandshakeRole:        ServerWithClientAuth
CtracedClearText:     Off
Trace:                 2
.....
TTLSCipherParms:
  v3CipherSuites:
    35 TLS_RSA_WITH_AES_256_CBC_SHA
  Policy created: Mon May 14 18:33:54 2012
  Policy updated: Tue May 15 11:53:54 2012

```

---

### 3.7.2 Stopping RACF on all systems

Issue the **#STOP** command on SC74, SC75, and SC76. Issue the **/D J,RACF** command to make sure that it is down on all systems.



### 3.7.3 Starting RACF on all systems without defining any RRSF nodes

Add a RACF parameter library specified by the RACFPARM statement to the RACF address space started task. Then issue command `/S RACF,sub=mstr` on SC74, SC75, and SC76. RACF uses the default IRROPT00 member shown in Example 3-29.

*Example 3-29 Contents of default IRROPT00 member*

---

```
ALLOCATE FILE(SYSUT1) data set('SYS1.SAMPLIB(IRRDPSDS)') SHR
IRRDPI00 UPDATE
FREE FILE(SYSUT1)
```

---

The syslog looks like Example 3-30.

*Example 3-30 RACF startup without any RRSF nodes*

---

```
S RACF,SUB=MSTR
IRR812I PROFILE RACF.* (G) IN THE STARTED CLASS WAS USED 207
      TO START RACF WITH JOBNAME RACF.
IEF196I      1 //RACF      JOB MSGLEVEL=1
IEF196I      2 //STARTING EXEC RACF
IEF196I STMT NO. MESSAGE
IEF196I      2 IEFC001I PROCEDURE RACF WAS EXPANDED USING SYSTEM
IEF196I LIBRARY SYS1.PROCLIB
IEF196I      3 XXRACF PROC OPT=00
IEF196I      00050000
IEF196I      4 XXRACF EXEC PGM=IRRSSM00,REGION=2M,PARM='OPT=&OPT'
IEF196I      00100000
IEF196I      IEFC653I SUBSTITUTION JCL - PGM=IRRSSM00,REGION=2M,
IEF196I PARM='OPT=00'
IEF196I      5 XXRACFPARM DD DISP=SHR,DSN=RRSF.JCL
IEF196I      00110000
IEF695I START RACF      WITH JOBNAME RACF      IS ASSIGNED TO USER RACF
      , GROUP SYS1
IEF196I IEF695I START RACF      WITH JOBNAME RACF      IS ASSIGNED TO
IEF196I USER RACF      , GROUP SYS1
IEF196I IEF236I ALLOC. FOR RACF RACF
IEF196I IEF237I C72E ALLOCATED TO RACFPARM
IRRBO64I (#) RACF JCL PARM SPECIFICATION IS: OPT=00. 226
IEF196I IEF237I DMY ALLOCATED TO SYSTSIN
IEF196I IEF237I DMY ALLOCATED TO SYSTSPRT
IEF196I IEF237I RACF ALLOCATED TO SYS00001
IRRBO01I (#) RACF SUBSYSTEM HRF7780 IS ACTIVE. 230
IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY MEMBER 231
      IRROPT00.
IEF196I IEF237I 9202 ALLOCATED TO SYSUT1
IEF196I IEF285I  SYS1.SAMPLIB      KEPT
IEF196I IEF285I  VOL SER NOS= Z1DRA1.
IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER 235
      IRROPT00 IS COMPLETE.
IRRC051I (#) RACF REMOTE SHARING TCP LISTENER TASK STARTING. 236
IRRB002I (#) INITIALIZATION COMPLETE FOR RACF SUBSYSTEM. 237
```

---

You can see the RRSF TCP listener task is starting. It will not do anything until the RRSF nodes are defined and activated in the next step.

Issue **#TARGET LIST** command on each system. You will see no nodes are defined as shown in Example 3-31.

*Example 3-31 TARGET LIST command*

---

```
#TARGET LIST
IRRM024I (#) RACF SUBSYSTEM TARGET COMMAND HAS FOUND THAT NO NODES
ARE CURRENTLY DEFINED.
```

---

### 3.7.4 Defining the RRSF node on local main system SC75 in PLEX75

Issue the command **#SET INCLUDE(75)** on SC75. This command uses IRROPT75 member to define the PLEX75 node. SC75 is a member of this node as defined by the local parameter on the target statement, It is also a member of the main system as defined by the main parameter of the target statement. The contents of IRROPT75 are shown in Example 3-32.

*Example 3-32 The contents of IRROPT75*

---

```
TARGET NODE(PLEX75) SYSNAME(SC75) MAIN LOCAL      -
  PREFIX(SYS1.RACF) WORKSPACE(VOLUME(BH5ST2))
TARGET NODE(PLEX75) SYSNAME(SC74) LOCAL            -
  PREFIX(SYS1.RACF) WORKSPACE(VOLUME(BH5ST2))
TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP(ADDRESS(9.12.4.72)))
TARGET NODE(PLEX75) SYSNAME(SC74) PROTOCOL(TCP(ADDRESS(9.12.4.70)))
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE
```

---

Example 3-33 shows the response from the **#SET** command that defined the RRSF node to SC75.

*Example 3-33 Define the RRSF node SC75*

---

```
#SET INCLUDE(75)
IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY MEMBER
IRROPT75.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRJ000I (#) RACF RACF LOCAL NODE TRANSACTION PROGRAM STARTING UNDER
user ID RACF GROUP STCGROUP.
IRRM035I (#) RACF SUBSYSTEM TARGET COMMAND CANNOT MAKE NODE PLEX75
SYSNAME SC74 OPERATIVE BECAUSE ONLY THE DEFINED STATE IS
ALLOWED.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER
IRROPT75 IS COMPLETE.
IRRH004I (#) RACF SUBSYSTEM SET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRC054I (#) RACF REMOTE SHARING TCP LISTENER HAS BEEN SUCCESSFULLY
ESTABLISHED.
```

---

**Consideration:** In a multi-system RRSF node, define the local main system before defining other local systems. SC75 is defined before SC74 in IRROPT75.

### 3.7.5 Defining the RRSF node on local system SC74 in PLEX75

Issue the command **#SET INCLUDE(75)** on SC74. This command uses the common IRROPT75 member, shown in Example 3-32 on page 76, to define the PLEX75 node to SC74 as shown in Example 3-34.

*Example 3-34 Define the RRSF node SC74*

---

**#SET INCLUDE(75)**

```
IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY MEMBER
              IRROPT75.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM035I (#) RACF SUBSYSTEM TARGET COMMAND CANNOT MAKE NODE PLEX75
              SYSNAME SC75 OPERATIVE BECAUSE ONLY THE DEFINED STATE IS
              ALLOWED.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRJ000I (#) RACF RACF LOCAL NODE TRANSACTION PROGRAM STARTING UNDER
              user ID RACF GROUP STCGROUP.
IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER
              IRROPT75 IS COMPLETE.
IRRH004I (#) RACF SUBSYSTEM SET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRC054I (#) RACF REMOTE SHARING TCP LISTENER HAS BEEN SUCCESSFULLY
              ESTABLISHED.
```

---

Issue **#LISTP** command to show the RRSF status on SC75 as shown in Example 3-35.

*Example 3-35 RRSF status on SC75*

---

**#TARGET LISTP**

```
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC74 IS IN THE DEFINED
              STATE.
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC75 (MAIN) IS IN THE
              OPERATIVE ACTIVE STATE.
IRRM091I (#)      - LOCAL NODE TCP LISTENER IS ACTIVE.
#TARGET LIST NODE(PLEX75)
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC74 IS IN THE OPERATIVE
              ACTIVE STATE.
IRRM091I (#)      - LOCAL NODE TCP LISTENER IS ACTIVE.
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC75 (MAIN) IS IN THE
              DEFINED STATE.
```

---

Issue **#LISTP** command to show the RRSF status on SC74. The output is shown in Example 3-36.

*Example 3-36 RRSF status on SC74*

---

**#TARGET LISTP**

```
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC74 IS IN THE OPERATIVE
              ACTIVE STATE.
IRRM091I (#)      - LOCAL NODE TCP LISTENER IS ACTIVE.
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC75 (MAIN) IS IN THE
              DEFINED STATE.
```

---

**Consideration:** This example uses the same IRROPTxx member and a common set of TARGET statements for all systems in the PLEX75 node. However, you need to take into account the following considerations:

- ▶ The systems that share the common IRROPTxx member must have access to the volume specified in the workspace parameter.
- ▶ The systems must be able to define data sets with the qualifier specified in the prefix parameter.
- ▶ Define the main system of a multi-system node first.
- ▶ All members of a multi-system node must have a local parameter defined.

### 3.7.6 Defining the RRSF node on local system SC76 in PLEX76

The IRROPT76 member contains the statements to define the PLEX76 node. Its contents are shown in Example 3-37.

*Example 3-37 The contents of IRROPT76*

---

```
TARGET NODE(PLEX76) LOCAL                -
  PREFIX(SYS1.RACF) WORKSPACE(VOLUME(BH6ST1))
TARGET NODE(PLEX76)                      PROTOCOL(TCP(ADDRESS(9.12.4.126)))
TARGET NODE(PLEX76)                      OPERATIVE
```

---

Issue command **#SET INCLUDE(76)** on SC76. This command uses the IRROPT76 member whose contents are shown by Example 3-37. The results are shown in Example 3-38.

*Example 3-38 Define the RRSF node on system SC76*

---

```
#SET INCLUDE(76)
IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY MEMBER
            IRROPT76.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRJ000I (#) RACF RACF LOCAL NODE TRANSACTION PROGRAM STARTING UNDER
            user ID RACF GROUP SYS1.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER
            IRROPT76 IS COMPLETE.
IRRH004I (#) RACF SUBSYSTEM SET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRC054I (#) RACF REMOTE SHARING TCP LISTENER HAS BEEN SUCCESSFULLY
            ESTABLISHED.
```

---

Issue **#LISTP** command to show the RRSF status on SC76. The log as shown in Example 3-39 is displayed.

*Example 3-39 RRSF status on SC76*

---

```
#TARGET LISTP
IRRM009I (#) LOCAL RRSF NODE PLEX76 IS IN THE OPERATIVE ACTIVE STATE.
IRRM091I (#) - LOCAL NODE TCP LISTENER IS ACTIVE.
```

---

### 3.7.7 Defining the remote node to SC75

From SC75 (a member of multi-system node PLEX75) the remote node is PLEX76 (a single system node). Use the member named IRROPT76 as shown in Example 3-40.

*Example 3-40 IRROPT76*

---

```
TARGET NODE(PLEX76)                -
  PREFIX(SYS1.RACF) WORKSPACE(VOLUME(BH6ST2))
TARGET NODE(PLEX76)                PROTOCOL(TCP(ADDRESS(9.12.4.126)))
TARGET NODE(PLEX76)                OPERATIVE
```

---

**Tip:** Although the IRROPT76 member used on SC75 (Example 3-40) looks almost exactly like the IRROPT76 member used on SC76 (Example 3-37), there is a difference. The volume used for the workspace data set is different. If you have similarly named volumes on both nodes, you can use the same workspace volume name. Additionally, because you are defining a remote node, there is no LOCAL parameter on the TARGET statement.

Issue the command **#SET INCLUDE(76)** on SC75. The output looks like Example 3-41.

*Example 3-41 Define remote node PLEX76 to SC75*

---

```
#SET INCLUDE(76)
IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY MEMBER
             IRROPT76.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRQ001I (#) RACF REMOTE SHARING TCP CONNECTOR TASK STARTING FOR NODE
             PLEX76 WITH HOST ADDRESS 9.12.4.126.
IRRI000I (#) LOCAL RACF NODE PLEX75 SYSNAME SC75 IS ATTEMPTING TO
             CONTACT PARTNER RACF NODE PLEX76.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER
             IRROPT76 IS COMPLETE.
IRRH004I (#) RACF SUBSYSTEM SET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRQ010I (#) RACF REMOTE SHARING TCP CONNECTOR TASK TERMINATING FOR
             NODE PLEX76 WITH HOST ADDRESS 9.12.4.126.
```

---

### 3.7.8 Defining the remote node to SC74

From SC74, which a member of multi-system node PLEX75, the remote node is PLEX76 (a single system node). Use the IRROPT76 member shown in Example 3-40 to define the remote node. Issue the command **#SET INCLUDE(76)** on SC74. The output looks like Example 3-42.

*Example 3-42 Define remote node PLEX76 to SC74*

---

```
#SET INCLUDE(76)
IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY MEMBER
             IRROPT76.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRQ001I (#) RACF REMOTE SHARING TCP CONNECTOR TASK STARTING FOR NODE
             PLEX76 WITH HOST ADDRESS 9.12.4.126.
             PLEX76 WITH HOST ADDRESS 9.12.4.126.
```

---

```

IRRI000I (#) LOCAL RACF NODE PLEX75 SYSNAME SC74 IS ATTEMPTING TO
CONTACT PARTNER RACF NODE PLEX76.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER
IRROPT76 IS COMPLETE.
IRRH004I (#) RACF SUBSYSTEM SET COMMAND HAS COMPLETED SUCCESSFULLY.

```

---

### 3.7.9 Defining the remote node to SC76

From SC76 (a single system node), the remote node is PLEX76 (a multi-system node). Use the RACF parameter library member named IRROPT75 as shown in Example 3-43.

*Example 3-43 IRROPT75*

---

```

TARGET NODE(PLEX75) SYSNAME(SC75) MAIN -
PREFIX(SYS1.RACF) WORKSPACE(VOLUME(BH5ST2))
TARGET NODE(PLEX75) SYSNAME(SC74) -
PREFIX(SYS1.RACF) WORKSPACE(VOLUME(BH5ST2))
TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP(ADDRESS(9.12.4.72)))
TARGET NODE(PLEX75) SYSNAME(SC74) PROTOCOL(TCP(ADDRESS(9.12.4.70)))
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE

```

---

**Tip:** IRROPT75 (Example 3-43) looks similar to the IRROPT75 member (Example 3-32 on page 76) that was used to define local node PLEX75 to system SC75. However, there are important points to note because this IRROPT75 member is being used by SC76 to define a REMOTE node:

- ▶ There are no LOCAL statements specified on the target commands.
- ▶ The main system is defined first because the remote node is a multi-system node.
- ▶ The workspace volume has the same name in both IRROPT75 members. System SC76 has access to a volume named BH5ST2, but this is a different physical volume than the BH5ST2 used by PLEX75. You can plan ahead to have the volume names for workspace data set for the different RRSF nodes match. This configuration reduces the number of changes that must be made to TARGET statements if you have many RRSF nodes.

Issue the command **#SET INCLUDE(75)** on SC76. The output looks like Example 3-44.

*Example 3-44 Defining remote node PLEX75 to SC76*

---

```

#SET INCLUDE(75)
IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY MEMBER
IRROPT75.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRQ001I (#) RACF REMOTE SHARING TCP CONNECTOR TASK STARTING FOR NODE
PLEX75 SYSNAME SC75 WITH HOST ADDRESS 9.12.4.72.
IRRI000I (#) LOCAL RACF NODE PLEX76 IS ATTEMPTING TO CONTACT PARTNER
RACF NODE PLEX75 SYSNAME SC75.
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.

```

```

IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER
            IRROPT75 IS COMPLETE.
IRRH004I (#) RACF SUBSYSTEM SET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRQ001I (#) RACF REMOTE SHARING TCP CONNECTOR TASK STARTING FOR NODE
            PLEX75 SYSNAME SC74 WITH HOST ADDRESS 9.12.4.70.
IRRI000I (#) LOCAL RACF NODE PLEX76 IS ATTEMPTING TO CONTACT PARTNER
            RACF NODE PLEX75 SYSNAME SC74.
IRRQ010I (#) RACF REMOTE SHARING TCP CONNECTOR TASK TERMINATING FOR
            NODE PLEX75 SYSNAME SC75 WITH HOST ADDRESS 9.12.4.72.
IRRN001I (#) RACF REMOTE SHARING TCP COMMUNICATOR TASK STARTING FOR
            NODE PLEX75 SYSNAME SC75 WITH HOST ADDRESS 9.12.4.72.
IRRI027I (#) RACF COMMUNICATION WITH TCP NODE PLEX75 SYSNAME SC75 HAS
            BEEN SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35
            TLS_RSA_WITH_AES_256_CBC_SHA.
IRRQ010I (#) RACF REMOTE SHARING TCP CONNECTOR TASK TERMINATING FOR
            NODE PLEX75 SYSNAME SC74 WITH HOST ADDRESS 9.12.4.70.
IRRN001I (#) RACF REMOTE SHARING TCP COMMUNICATOR TASK STARTING FOR
            NODE PLEX75 SYSNAME SC74 WITH HOST ADDRESS 9.12.4.70.
IRRI027I (#) RACF COMMUNICATION WITH TCP NODE PLEX75 SYSNAME SC74 HAS
            BEEN SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35
            TLS_RSA_WITH_AES_256_CBC_SHA.
IRRM091I (#)      - LOCAL NODE TCP LISTENER IS ACTIVE.

```

---

### 3.7.10 Displaying the RRSF status on every system

Issue the **#LISTP** command to show the RRSF status for each node. The log output shown in Example 3-45 is displayed when you issue the command from SC76.

#### *Example 3-45 RRSF status on SC76*

```

#TARGET LISTP
IRRM009I (#) REMOTE RRSF NODE PLEX75 SYSNAME SC74 PROTOCOL TCP IS IN
            THE OPERATIVE ACTIVE STATE.
IRRM009I (#) REMOTE RRSF NODE PLEX75 SYSNAME SC75 PROTOCOL TCP (MAIN)
            IS IN THE OPERATIVE ACTIVE STATE.
IRRM009I (#) LOCAL RRSF NODE PLEX76 IS IN THE OPERATIVE ACTIVE STATE.

```

---

Issue **#LISTP** command to show the RRSF status on SC75. The output is shown in Example 3-46.

#### *Example 3-46 RRSF status on SC75*

```

#TARGET LISTP
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC74 IS IN THE DEFINED
            STATE.
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC75 (MAIN) IS IN THE
            OPERATIVE ACTIVE STATE.
IRRM091I (#)      - LOCAL NODE TCP LISTENER IS ACTIVE.
IRRM009I (#) REMOTE RRSF NODE PLEX76 PROTOCOL TCP IS IN THE OPERATIVE
            ACTIVE STATE.

```

---

All systems are in the correct states and the RRSF network is active.







## Converting APPC connections to TCP connections

This chapter addresses how to convert existing RRSF connections from one communications protocol (APPC) to another (TCP). It focuses on converting an existing group of RRSF systems from APPC to TCP.

Consideration is given to the management of IRROPTxx RRSF parameter library files which are used to save RRSF configurations. The goal is to perform the conversion once. You can save the new configuration in IRROPTxx members in such a way that the standard RRSF safeguards against loss of work remain in place.

This chapter demonstrates protocol conversion from APPC to TCP. The examples and description involve conversion from APPC to TCP. However, the procedure to convert from TCP to APPC is similar.

This chapter includes the following sections:

- ▶ Sample configuration
- ▶ Protocol conversion overview
- ▶ Protocol conversion process
- ▶ Sample configuration IRROPTxx files
- ▶ A simple protocol conversion
- ▶ Target command changes
- ▶ Protocol conversion and the local node
- ▶ Multi-System Node considerations
- ▶ Order of protocol definition
- ▶ Workspace data set considerations
- ▶ Information required before protocol conversion
- ▶ IRROPTxx considerations
- ▶ Conversion of the sample configuration
- ▶ Earlier system considerations

## 4.1 Sample configuration

The sample environment consists of these systems:

- ▶ A single-system node (SSN), PLEX76
- ▶ A two system multi-system node (MSN), PLEX75
- ▶ A three system MSN, PLEX81

All of the systems communicate with one another using the APPC protocol as shown in Figure 4-1. Communications between all of the systems will be converted to TCP except for system ADCD in PLEX81. System ADCD is running z/OS at V1R11, which does not support TCP communications for RRSF.

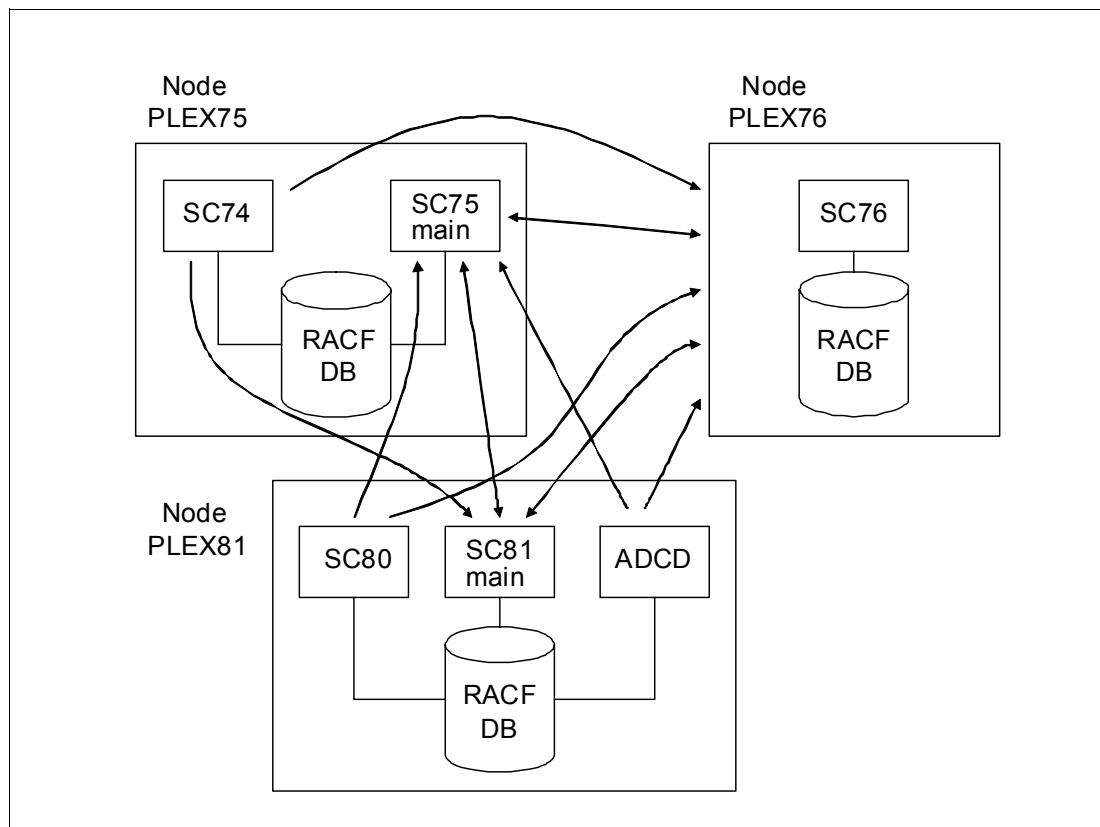


Figure 4-1 Sample configuration of a three system MSN, two system MSN, and an SSN

## 4.2 Protocol conversion overview

Protocol conversion is always done on a live, running system. RRSF will run during the conversion, and this data must not be lost. You do not have to quiesce RRSF or anything else on the system. However, it is better to schedule a protocol conversion during a quiet period if possible to minimize the conversion time.

Protocol conversion is done on one system connection at a time. Any individual connection can use either protocol, and this connection can have its protocol converted back and forth as wanted. A single RRSF system can use APPC on some connections and TCP on others. There is no requirement on the order in which connections are converted. There is no requirement on the order of performing protocol conversion within MSNs with respect to the

main system. The main system can be converted at any time in relation to the other systems, or not at all.

The steps in Chapter 3, “Configuring RRSF for TCP/IP” on page 27 must be performed and AT-TLS communications configured between the systems and systems in the RRSF environment. A protocol conversion requires the new (TCP) protocol be able to connect to and communicate with the remote systems. Otherwise, the conversion process cannot complete.

The old (APPC) protocol does not need to be in the operative active state at the time of protocol conversion. However, it must be at least dormant so its workspace files, inmsg and outmsg, are allocated. If the old protocol is active and establishing communications by using the new protocol fails, the old protocol continues to work until communication is established using the new protocol.

## 4.3 Protocol conversion process

Protocol conversion of a single system connection from APPC to TCP can be as simple as a few steps run on live systems:

- ▶ Use the RRSF **target** command to add the TCP protocol to the LOCAL node on each side of the connection.
- ▶ Use the RRSF **target** command to create a definition for the other system by using the TCP protocol and make this new definition operative. This process is done on both sides of the connection.
- ▶ In a few moments, depending on the current level of RRSF activity across this connection, messages are displayed indicating that the conversion is complete.

Repeat this process for each connection.

## 4.4 Sample configuration IRROPTxx files

The sample configuration in this chapter consists of three RRSF nodes: PLEX76, PLEX75, and PLEX81. PLEX76 is an SSN and has one system, whereas PLEX75 and PLEX81 are MSNs with multiple systems in each. The systems in each MSN share the IRROPTxx parameter library file. These files define the entire RRSF network by using APPC communications, and are modified to communicate by using TCP. All systems within an MSN share an IRROPTxx file.

The file used by all members of PLEX75 is shown in Figure 4-2.

```

TARGET NODE(PLEX75) SYSNAME(SC75) MAIN LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC75)))
TARGET NODE(PLEX75) SYSNAME(SC74) LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC74)))
TARGET NODE(PLEX81) SYSNAME(SC81) MAIN PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC81)))
TARGET NODE(PLEX81) SYSNAME(SC80) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC80)))
TARGET NODE(PLEX81) SYSNAME(ADCD) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))
TARGET NODE(PLEX76) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH6ST1)) PROTOCOL(APPC(LUNAME(RRSFSC76)))
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC80) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(ADCD) OPERATIVE
TARGET NODE(PLEX76) OPERATIVE

```

Figure 4-2 Shared IRROPTxx file for all systems in PLEX75.

IRROPTxx members used by systems in node systems in PLEX81 are shown in Figure 4-3.

```

TARGET NODE(PLEX81) SYSNAME(SC81) MAIN LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC81)))
TARGET NODE(PLEX81) SYSNAME(SC80) LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC80)))
TARGET NODE(PLEX81) SYSNAME(ADCD) LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))
TARGET NODE(PLEX75) SYSNAME(SC75) MAIN PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC75)))
TARGET NODE(PLEX75) SYSNAME(SC74) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC74)))
TARGET NODE(PLEX76) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH6ST1)) PROTOCOL(APPC(LUNAME(RRSFSC76)))
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC80) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(ADCD) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE
TARGET NODE(PLEX76) OPERATIVE

```

Figure 4-3 Shared IRROPTxx member for all systems in PLEX81

IRROPTxx members used by systems in node systems in PLEX76 are shown in Figure 4-4.

```

TARGET NODE(PLEX76) LOCAL          PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH6ST1)) PROTOCOL(APPC(LUNAME(RRSFSC76)))
TARGET NODE(PLEX81) SYSNAME(SC81) MAIN PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC81)))
TARGET NODE(PLEX81) SYSNAME(SC80)   PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC80)))
TARGET NODE(PLEX81) SYSNAME(ADCD)   PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))
TARGET NODE(PLEX75) SYSNAME(SC75) MAIN PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC75)))
TARGET NODE(PLEX75) SYSNAME(SC74)   PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC74)))
TARGET NODE(PLEX76)                OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC80) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(ADCD) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE

```

Figure 4-4 Shared IRROPTxx member for single system node PLEX76

These IRROPTxx members are the starting point for the sample protocol conversion in this chapter.

## 4.5 A simple protocol conversion

Convert communications between SSN PLEX76 and MSN main system SC75 of node PLEX75 from APPC, as configured in Figure 4-2 on page 86 and Figure 4-4.

**Tip:** For this discussion, the # is the command prefix that precedes all RRSF commands.

The current RRSF environment on the PLEX76 system is shown in Example 4-1.

Example 4-1 TARGET LIST issued from system PLEX76

```

#TARGET LISTPROTOCOL
...
IRRM009I (#) REMOTE RRSF NODE PLEX75 SYSNAME SC75 PROTOCOL APPC (MAIN)
           IS IN THE OPERATIVE ACTIVE STATE.
IRRM009I (#) LOCAL RRSF NODE PLEX76 IS IN THE OPERATIVE ACTIVE STATE.
IRRM091I (#)          - LOCAL NODE APPC LISTENER IS ACTIVE.
...

```

The current RRSF environment on the SC75 system is shown in Example 4-2

Example 4-2 TARGET LIST issued from system SC75

```

#TARGET LISTPROTOCOL
...
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC75 (MAIN) IS IN THE
           OPERATIVE ACTIVE STATE.

```

```
IRRM091I (#)      - LOCAL NODE APPC LISTENER IS ACTIVE.  
IRRM009I (#) REMOTE RRSF NODE PLEX76 PROTOCOL APPC IS IN THE OPERATIVE  
ACTIVE STATE.  
...
```

---

Add the TCP protocol to the LOCAL node of PLEX76 by using the TARGET command. This command enables PLEX76 to accept inbound TCP connections and initiate outbound TCP connections as shown in Example 4-3.

*Example 4-3 Issued from PLEX76 to define TCP as a supported protocol to system PLEX76*

---

```
#TARGET NODE(PLEX76) DORMANT  
#TARGET NODE(PLEX76) PROTOCOL(TCP)  
#TARGET NODE(PLEX76) OPERATIVE
```

---

Now, add the TCP protocol to system SC75 as shown in Example 4-4.

*Example 4-4 Issued from system SC75 to define TCP as a supported protocol to system SC75*

---

```
#TARGET NODE(PLEX75) SYSNAME(SC75) DORMANT  
#TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP)  
#TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
```

---

Define system SC75 to PLEX76. This is a new definition of system SC75. All information required to define a remote node must be specified before TCP communications can be attempted. Nothing is saved from the existing APPC definition of system SC75. The definition is broken up into multiple TARGET commands for legibility. The final TARGET OPERATIVE command fails because the corresponding definition has not yet been created on the remote node. However, the local node is operative after the remote node connection is made operative with the TCP protocol added (Example 4-5).

*Example 4-5 Issued from system PLEX76 to define remote TCP system SC75*

---

```
#TARGET NODE(PLEX75) SYSNAME(SC75) MAIN PROTOCOL(TCP(ADDRESS((9.12.4.70)))  
#TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP) PREFIX(RRSF.RACF)  
WORKSPACE(VOLUME(BH5ST2))  
#TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP) OPERATIVE
```

---

Define PLEX76 to system SC75 as shown in Example 4-6.

*Example 4-6 Issued from SC75 to define remote TCP system PLEX76*

---

```
#TARGET NODE(PLEX76) PROTOCOL(TCP(ADDRESS(9.12.4.126)))  
#TARGET NODE(PLEX76) PROTOCOL(TCP) PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH6ST1))  
#TARGET NODE(PLEX76) PROTOCOL(TCP) OPERATIVE
```

---

Completion messages are displayed on the console of both systems after a few moments as shown in Example 4-7.

*Example 4-7 Successful completion of protocol conversion on SC75*

---

```
IRRC057I (#) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE PLEX76 HAS BEEN  
INITIATED.  
...  
IRRI027I (#) RACF COMMUNICATION WITH TCP NODE PLEX76 HAS BEEN 985  
SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35  
TLS_RSA_WITH_AES_256_CBC_SHA.
```

```
...
IRRC058I (#) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE PLEX76 IS
COMPLETE.
```

---

The conversion from APPC communication to TCP is now complete for the connection between systems PLEX76 and system SC75. The TARGET command is used to verify that the new protocol is in place as shown in Example 4-8.

*Example 4-8 Output of target list on PLEX76 showing the new communication protocol.*

---

```
#TARGET LISTPROTOCOL
```

```
...
IRRM009I (#) REMOTE RRSF NODE PLEX75 SYSNAME SC75 PROTOCOL TCP (MAIN) IS IN THE
OPERATIVE ACTIVE STATE.
IRRM009I (#) LOCAL RRSF NODE PLEX76 IS IN THE OPERATIVE ACTIVE STATE.
IRRM091I (#)      - LOCAL NODE APPC LISTENER IS ACTIVE.
IRRM091I (#)      - LOCAL NODE TCP LISTENER IS ACTIVE.
...
```

---

## 4.6 Target command changes

Before z/OS V1R13, each definition of a remote system was unique. That is, there can be only one node with a node and sysname combination. Any attempt to define a second system with the same node name and sysname as an existing system is treated as a modification of the existing definition.

This configuration is changed in z/OS V1R13. Multiple nodes of the same name node and sysname can be defined if they differ in protocol (Example 4-9).

*Example 4-9 Define a single remote system with two protocols, each protocol with a different prefix*

---

```
#target node(sample) proto(appc) prefix(abc)
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
#TARGET
IRRM009I (#) REMOTE RRSF NODE SAMPLE IS IN THE ??? STATE.
#target node(sample) proto(tcp) prefix(def)
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
#target
IRRM009I (#) REMOTE RRSF NODE SAMPLE PROTOCOL APPC IS IN THE ??? STATE.
IRRM009I (#) REMOTE RRSF NODE SAMPLE PROTOCOL TCP IS IN THE ??? STATE.
```

---

After a second protocol is defined for a remote system, all subsequent **target** commands that act on that system must specify the protocol as shown in Example 4-10. This configuration allows the correct remote definition is used by **target**.

*Example 4-10 Use target command to create a single remote system with multiple protocols*

---

```
1#target node(sample) description(sample)
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
2#target node(sample) protocol(tcp(address(fake.ip)))
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
3#target node(sample) description('sample tcp node')
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
4#target list
```

```

IRRM009I (#) REMOTE RRSF NODE SAMPLE IS IN THE ??? STATE.
5#target node(sample) protocol(appc) description('sample')
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
6#target node(sample) description('sample appc')
IRRM087I (#) RACF SUBSYSTEM TARGET COMMAND REQUIRES THAT A PROTOCOL BE
SPECIFIED FOR NODE SAMPLE TO IDENTIFY THE INTENDED
PROTOCOL INSTANCE.
IRRM003I (#) RACF SUBSYSTEM TARGET COMMAND ENDED IN ERROR.
7#target node(sample) protocol(appc) description('sample appc')
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY
8#target list
IRRM009I (#) REMOTE RRSF NODE SAMPLE PROTOCOL TCP IS IN THE ??? STATE.
IRRM009I (#) REMOTE RRSF NODE SAMPLE PROTOCOL APPC IS IN THE ??? STATE.

```

---

**1-4** A new system definition is created and has its description modified and its protocol set.

**5** A second protocol is defined for the same system.

**6-7** Because multiple protocols are now defined for the system, all subsequent **target** commands must specify the wanted protocol to identify which definition to manage.

**8** Target list shows both protocols for the system.

TARGET LIST shows the protocol in the output only when more than one protocol is defined for a system. TARGET LISTPROTOCOL is similar to TARGET LIST. It always includes the protocol of the listed systems in the output, even if only one protocol is defined (Example 4-11).

*Example 4-11 The difference between target list and target list protocol*

---

```

#TARGET NODE(SAMPLEA) protocol(appc) prefix(x)
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
#TARGET NODE(SAMPLET) protocol(TCP) prefix(x)
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
#TARGET LIST
IRRM009I (#) REMOTE RRSF NODE SAMPLEA IS IN THE ??? STATE.
IRRM009I (#) REMOTE RRSF NODE SAMPLET IS IN THE ??? STATE.
#TARGET LISTPROTOCOL
IRRM009I (#) REMOTE RRSF NODE SAMPLEA PROTOCOL APPC IS IN THE ???
STATE.
IRRM009I (#) REMOTE RRSF NODE SAMPLET PROTOCOL TCP IS IN THE ??? STATE.

```

---



Detailed information about all protocol definitions for a system is displayed when the TARGET LIST command lists details for the system (Figure 4-5).

```
#TARGET NODE(PLEX81) SYSNAME(SC81) LIST
IRRM010I (#) RACF SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE PLEX81 101
      SYSNAME SC81 (MAIN) PROTOCOL APPC:
STATE      - OPERATIVE PENDING CONNECTION
DESCRIPTION - <NOT SPECIFIED>
PROTOCOL   - APPC
            LU NAME          - RRSFSC81
            TP PROFILE NAME  - IRRRACF
            MODENAME         - <NOT SPECIFIED>
TIME OF LAST TRANSMISSION TO - <NONE>
TIME OF LAST TRANSMISSION FROM - <NONE>
WORKSPACE FILE SPECIFICATION
PREFIX      - "SYS1.RACF"
WDSQUAL     - <NOT SPECIFIED>
FILESIZE    - 500
VOLUME      - BH8ST4
FILE USAGE
            "SYS1.RACF.RRSFSC76.RRSFSC81.INMSG"
              - CONTAINS 0 RECORD(S)
              - OCCUPIES 1 EXTENT(S)
            "SYS1.RACF.RRSFSC76.RRSFSC81.OUTMSG"
              - CONTAINS 4000 RECORD(S)
              - OCCUPIES 3 EXTENT(S)
IRRM010I (#) RACF SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE PLEX81
      SYSNAME SC81 (MAIN) PROTOCOL TCP:
STATE      - DORMANT REMOTE
DESCRIPTION - <NOT SPECIFIED>
PROTOCOL   - TCP
            HOST ADDRESS     - 9.12.4.47
            LISTENER PORT    - 18136
TIME OF LAST TRANSMISSION TO - <NONE>
TIME OF LAST TRANSMISSION FROM - <NONE>
WORKSPACE FILE SPECIFICATION
PREFIX      - "RRSF.RACF"
WDSQUAL     - <NOT SPECIFIED>
FILESIZE    - 500
VOLUME      - BH8ST4
FILE USAGE
            "RRSF.RACF.SC76.SC81.INMSG"
              - CONTAINS 0 RECORD(S)
              - OCCUPIES 1 EXTENT(S)
            "RRSF.RACF.SC76.SC81.OUTMSG"
              - CONTAINS 0 RECORD(S)
              - OCCUPIES 1 EXTENT(S)
```

Figure 4-5 Target list output for a specific system that has multiple protocols defined

## 4.7 Protocol conversion and the local node

The local node of an RRSF configuration has always been significant. Commands can be sent to the local node. They are run, but no network communications are used to send them around because the commands run on the system where they originated. The local node uses workspace files to checkpoint locally directed work. The definition of the local node, created by the target command, has two unrelated purposes:

- ▶ Store configuration information to support the creation and use of workspace files for locally directed work.
- ▶ Store configuration information used to accept inbound communications from remote nodes.

Conversely, in a remote system the communications definitions and workspace file definitions are both used to send work to and receive work from the remote system.

The first step of a protocol conversion is to update the local node definition so it is able to accept inbound connection from the new protocol. Unlike remote nodes, you need to add only the second protocol to the local node definition. There is no requirement to respecify the entire definition.

## 4.8 Multi-System Node considerations

There are no special considerations needed when performing protocol conversion on an MSN. Each system in the MSN is treated as a separate connection and is converted individually. The order in which the systems in the MSN are converted does not have any requirements. The main node can be converted first, last, in the middle, or not at all.

Sharing of IRROPTxx files across members of an MSN when some members of the MSN are at a z/OS release lower than z/OS V1R13 can cause issues when the TCP protocol is not supported. For more information, see 4.14, “Earlier system considerations” on page 100.

## 4.9 Order of protocol definition

The order in which the protocols are defined for a system is important. RRSF treats the **last** defined protocol as the wanted protocol. Any earlier defined protocols are deleted when the last protocol establishes communications. In Example 4-12, when APPC communications with remote node sample are established, the TCP definition for node sample is deleted.

This is true regardless of the order in which the tcp and appc protocols are made operative, or the order in which their respective communications are established. In Example 4-12, if tcp communications are established first, the appc definition remains indefinitely. When appc communications are established, the tcp definition is deleted.

The protocol definition time is the first time the protocol is displayed in a TARGET command for a remote system as shown in Example 4-12.

*Example 4-12 The tcp protocol is deleted when appc communications are established*

---

```
#TARGET NODE(SAMPLE) protocol(tcp) description('tcp node')
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
#TARGET NODE(SAMPLE) protocol(appc) description('appc node')
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
```

```
#TARGET  
IRRM009I (#) REMOTE RRSF NODE SAMPLE PROTOCOL TCP IS IN THE ??? STATE.  
IRRM009I (#) REMOTE RRSF NODE SAMPLE PROTOCOL APPC IS IN THE ??? STATE.
```

---

The order in which the protocols are shown reflects the order of protocols as they were originally defined. The order is displayed in either the short form (Example 4-12 on page 92) or the detailed form of the target list command (Figure 4-5 on page 91).

An interesting restriction levied by RRSF is that the first protocol must be in at least the dormant state before the second protocol is made operative. This restriction is because both the old and new protocol files are referenced during protocol conversion. Systems which are not dormant or operative do not have their workspace file allocated yet.

## 4.10 Workspace data set considerations

RRSF uses workspace data set to recover in-progress work in the event of a system failure. There are two workspace data sets: INMSG and OUTMSG. These files are allocated by using information specified on the RRSF TARGET command. The names of the files differ, depending on which communications protocol is used and whether the system corresponding to the file is part of an MSN. For more information about file names, see 5.7.3, “INMSG and OUTMSG data set names” on page 142.

A new set of files is created when the new communication protocol is defined and made dormant or operative. Figure 4-5 on page 91 shows a system with two protocols defined, and both sets of files allocated. The protocol conversion has not yet begun because the second protocol is in the dormant state.

A new set of workspace files is allocated for the new protocol. Therefore, a protocol conversion is an excellent opportunity to evaluate the current workspace file settings. Make any wanted changes to file prefix, size, volume, or SMS information at this time. Performing a protocol conversion is the simplest way to rename your workspace files, move them to a new volume, or reset their size. It is the only way to make these kinds of changes without deleting the system definitions and potentially losing work.

Because both the old protocol and new protocol files coexist for a short period, you must have enough DASD space available to accommodate both sets of files. If you use the **target wdsqual** option to alter the qualifiers of the workspace file names, be careful to not create a name collision. The names of the old and new protocol files must be different.

After the protocol conversion begins, the old protocol is deleted and any files associated with it are briefly assigned to the new protocol until they are empty. The files for both the old and new protocol are open and used simultaneously. New work is saved in the files of the new protocol, and old work is drained from the files of the old protocol. The TARGET command displays information about both sets of files in the output.

```

#TARGET NODE(PLEX81) SYS(SC81) LIST
IRRM010I (#) RACF SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE PLEX81 120
      SYSNAME SC81 (MAIN):
STATE      - OPERATIVE ACTIVE
DESCRIPTION - <NOT SPECIFIED>
PROTOCOL   - TCP
            HOST ADDRESS    - 9.12.4.47
            LISTENER PORT   - 18136
            AT-TLS POLICY:
              RULE_NAME      - DEFAULT_RRSF-SERVER~2
              CIPHER ALG     - 35 TLS_RSA_WITH_AES_256_CBC_SHA
              CLIENT AUTH    - REQUIRED
TIME OF LAST TRANSMISSION TO - 16:01:58 MAY 21, 2012
TIME OF LAST TRANSMISSION FROM - 16:01:58 MAY 21, 2012
WORKSPACE FILE SPECIFICATION
PREFIX      - "RRSF.RACF"
WDSQUAL     - <NOT SPECIFIED>
FILESIZE    - 500
VOLUME      - BH8ST4
FILE USAGE
            "RRSF.RACF.SC76.SC81.INMSG"
              - CONTAINS 117 RECORD(S)
              - OCCUPIES 1 EXTENT(S)
            "RRSF.RACF.SC76.SC81.OUTMSG"
              - CONTAINS 0 RECORD(S)
              - OCCUPIES 1 EXTENT(S)
CONVERSION FILE
            INMSG WORKSPACE FILE NOT ALLOCATED
            "SYS1.RACF.RRSFSC76.RRSFSC81.OUTMSG"
              - CONTAINS 3823 RECORD(S)
              - OCCUPIES 3 EXTENT(S)

```

Figure 4-6 Target list shows two sets of files during protocol conversion

## 4.11 Information required before protocol conversion

A protocol conversion is treated as a one time replacement of one communications protocol for another. It is not intended to be used as a hot swappable backup communications mechanism. As such, when changing the protocol, new definitions are created for each protocol. No information is retained or copied over from the system definition that uses the old protocol. This process creates a bit of extra work when setting up the new protocol. However, it makes it much easier to delete the old system definition from IRROPTxx when the protocol conversion is complete.

When converting from APPC to TCP, the most important piece of information required is the IP address of each system being converted. All other required information can be found by using the RRSF **target list** command. You can also find it in the IRROPTxx files used to set up your current RRSF configuration. If you want to change your workspace file attributes, now is the time to determine how you want them to be configured. You can change anything in the definition for the new protocol except the node name, system name, and main (or not main) designation.

## 4.12 IRROPTxx considerations

The protocol conversion described in 4.5, “A simple protocol conversion” on page 87 was too simple because it did not take into account updates to IRROPTxx. In a production environment, IRROPTxx must be updated alongside the protocol conversion.

The timing of updates to IRROPTxx with respect to performing the actual protocol conversion is important. In an unexpected system outage, ensure that work is not lost by having both protocols defined in the IRROPTxx member when the RACF address space restarts. This configuration allows RRSF to find both sets of workspace files when the system is restarted after the outage. The work that has accumulated in both sets of files is processed in the correct order, regardless of which protocol becomes active first.

After the protocol conversion is complete, it is safe to remove the old protocol definitions from IRROPTxx. However, there is no requirement to remove the old protocol from the IRROPTxx file. It can remain indefinitely because it only causes some extra processing to happen each time the IRROPTxx member is reloaded. Keeping both definitions in place also makes it possible to share IRROPTxx files with earlier systems. It might take a long time to complete the conversion of the entire enterprise, depending on how many RRSF systems need to be converted.

One approach to performing a large-scale protocol conversion involves converting one single system connection at a time. The updates are added to IRROPTxx, then entered on the live system. When the conversion is complete, the old protocol definition for that connection is removed from IRROPTxx.

Another approach involves copying the old IRROPTxx files and changing them to specify the new protocol. Then use the **set include(xx)** command to convert the whole RRSF configuration at once. Any definitions that are not valid continue to use the old protocol until the new definitions are fixed. This process can get confusing quickly. If there are more than a few connection failures, the number of error messages to read and understand can be daunting.

This chapter approaches the conversion of the sample configuration in these phases:

- ▶ Define TCP as a supported protocol on all systems by adding it to the local nodes.
- ▶ Focus on the connections between all systems in one node (SSN or MSN) to all of the systems in another node.
- ▶ Move on to next node pair

## 4.13 Conversion of the sample configuration

This conversion assumes that the IRROPTxx files in 4.4, “Sample configuration IRROPTxx files” on page 85 are used in the example environment.

### 4.13.1 Preliminary setup

Add each TCP remote definition to a new IRROPTTP member and test it out by using the **set include(tp)** command. After the TCP definition is found to be valid, it is moved to the main IRROPTxx file. For now, create an IRROPTTP file in the RACF parameter library, and add a single target command so it is not empty (Figure 4-7).

```
TARGET LISTPROTOCOL
```

Figure 4-7 New IRROPTTP file on system SC75 adding the same file on PLEX76 and on PLEX81

Including the new IRROPTTP file from the existing IRROPTxx is an easy way to ensure that the new TCP definition is always part of the main configuration for recovery purposes. This configuration also allows flexibility in making changes and getting the protocol definition to work properly. The old protocol continues to process RRSF work until the new protocol is able to communicate properly.

To add the new IRROPTTP to the active configuration, include it at the bottom of IRROPTxx as shown in Figure 4-8.

```
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC80) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(ADCD) OPERATIVE
TARGET NODE(PLEX76) OPERATIVE
SET INCLUDE(TP)
```

Figure 4-8 PLEX75 IRROPTxx file changes to include IRROPTTP

### 4.13.2 Updating local node definitions to support TCP in addition to APPC

Updating the local node definitions requires the following steps.

#### Updating local node definitions in IRROPTxx

The first step in converting the sample configuration is to add TCP as a supported protocol to the local nodes in all of the IRROPTxx files. If the IRROPTxx files are shared across members of an MSN, add the new protocol to all of the local node definition statements in IRROPTxx.

Place the **target** statement that adds the new protocol before the **target** statement that makes the local node operative as shown in Figure 4-9 on page 97. Otherwise the addition of the new protocol will fail. It might be necessary to move the operative keyword to a new command, depending on how IRROPTxx is structured. If the local node is part of an MSN, be sure to add the TCP protocol to all local nodes which use the new protocol.

The local system does not need to have an IP address specified. Specifying **protocol(tcp)** is enough to make the local system accept TCP connections.

```

TARGET NODE(PLEX75) SYSNAME(SC75) MAIN LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC75)))
TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP)
TARGET NODE(PLEX75) SYSNAME(SC74) LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC74)))
TARGET NODE(PLEX75) SYSNAME(SC74) PROTOCOL(TCP)
....
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE

```

Figure 4-9 PLEX75 IRROPTxx file changes (bolded lines are new)

### Updating local node definition on running systems

The next step is to manually add the TCP protocol to the local node of all of the running systems that use TCP as shown in Example 4-13. When the system starts, the new entries in IRROPTxx add these definitions automatically. However, restarting the system now to pick up the IRROPTxx changes can lead to lost work.

Example 4-13 Commands to enter on system SC75

```

#TARGET NODE(PLEX75) SYSNAME(SC75) DORMANT
#TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP)
#TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE

```

### 4.13.3 Converting communications between PLEX75 and PLEX76 from APPC to TCP

Edit IRROPTTP on PLEX76 and add a definition for PLEX75 system SC75. Because SC75 is the MSN main system in PLEX75 when using APPC, it must also be defined as main in the TCP definition (Figure 4-10). Make any wanted changes to workspace file definitions at this time.

```

TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP(ADDRESS(9.12.4.72))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH5ST4) filesize(750)) MAIN
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE PROTOCOL(TCP)

```

Figure 4-10 New IRROPTTP on PLEX76 containing definition for system PLEX75 system SC75

Edit IRROPTTP on PLEX75 and add a definition for PLEX76 as shown in Figure 4-11.

```

TARGET NODE(PLEX76) PROTOCOL(TCP(ADDRESS(9.12.4.126))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH6ST2))
TARGET NODE(PLEX76) OPERATIVE PROTO(TCP)

```

Figure 4-11 New IRROPTTP on PLEX75 containing a definition for system PLEX76

Activate the new definitions by issuing the **set include(tp)** command on both PLEX76 and SC75 (Example 4-14). This command starts the protocol conversion from APPC to TCP.

Example 4-14 PLEX76 output for protocol conversion between systems PLEX76 and SC75

```

#SET INCLUDE(TP)
IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY MEMBER 088

```

```

        irropttp.
IRRC057I (#) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE
PLEX75 HAS BEEN INITIATED.
IRRQ001I (#) RACF REMOTE SHARING TCP CONNECTOR TASK STARTING FOR NODE
PLEX75 SYSNAME SC75 WITH HOST ADDRESS 9.12.4.72.
IRRI000I (#) LOCAL RACF NODE PLEX76 IS ATTEMPTING TO CONTACT PARTNER
RACF NODE PLEX75 SYSNAME SC75.
IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER 106
        irropttp IS COMPLETE.
IRRH004I (#) RACF SUBSYSTEM SET COMMAND HAS COMPLETED SUCCESSFULLY.
IRRN001I (#) RACF REMOTE SHARING TCP COMMUNICATOR TASK STARTING FOR
NODE PLEX75 SYSNAME SC75 WITH HOST ADDRESS 9.12.4.72.
IRRN021I (#) APPC RECEIVE AND WAIT ENDING FOR LU RRSFSC75 NODE PLEX7
SYSNAME SC75.
IRRN009I (#) RACF APPC RECEIVE TRANSACTION PROGRAM COMPLETED FOR LU
RRSFSC75 NODE PLEX75 SYSNAME SC75.
IRRI027I (#) RACF COMMUNICATION WITH TCP NODE PLEX75 SYSNAME SC75 HA
        BEEN SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35
        TLS_RSA_WITH_AES_256_CBC_SHA.
IRRC058I (#) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE
PLEX75 SYSNAME SC75 IS COMPLETE.

```

---

The output shows TCP communication begin and the APPC communication terminate. At the end, the message **IRRC058I** indicates that the conversion from APPC to TCP is complete between PLEX76 and SC75.

The protocol conversion might fail because of a typographical error or mistakes made in AT-TLS configuration. Fix the errors and run the **set include(tp)** command until the protocol conversion is successful. The old protocol continues to handle RRSF work indefinitely until the new protocol communicates successfully.

If a system restart occurs between messages IRRC057I and IRRC058I in Example 4-14 on page 97, the subsequent loading of IRROPTxx and IRROPTTP allows the protocol conversion to continue as though it were uninterrupted. The duration of the protocol conversion depends on how much work is present in the workspace files of the old protocol.

After the protocol conversion between PLEX76 and SC75 is successful, move the TCP definition from IRROPTTP to IRROPTxx on PLEX76. Replace the APPC definition of SC75 with the TCP definition, and remove the TCP definition from IRROPTTP (Figure 4-12). This is a simple change because PLEX76 is the only system that uses this specific IRROPTxx file.

```

...
TARGET NODE(PLEX81) SYSNAME(ADCD)      PREFIX(SYS1.RACF) -
        WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))
TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP(ADDRESS(9.12.4.72))) -
        PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH5ST4) FILESIZE(750)) MAIN
TARGET NODE(PLEX75) SYSNAME(SC74)      PREFIX(SYS1.RACF) -
        WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC74)))
...
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE
SET INCLUDE(TP)

```

Figure 4-12 Updated definition for system SC75 in IRROPTxx on PLEX76 (changes are in bold)



Convert the connection between SC74 and PLEX76. This process is similar to what was already done for SC75. The first step is to update IRROPTTP on PLEX76 to contain a TCP definition for SC74, and **set include(tp)** to load the new definition (Figure 4-13).

```
TARGET NODE(PLEX75) SYSNAME(SC74) PROTOCOL(TCP(ADDRESS(9.12.4.70))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH5ST4))
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE PROTOCOL(TCP)
```

Figure 4-13 IRROPTTP on PLEX76 containing a TCP definition for system SC74.

Because IRROPTTP on PLEX75 already contains the appropriate definition for PLEX76, issue **set include(tp)** on SC74. Watch for the conversion complete messages or use the **target listprotocol** to verify that the conversion was successful (Example 4-15).

Example 4-15 Running the TARGET LISTPROTOCOL command

```
TARGET LISTPROTOCOL
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC74 IS IN THE OPERATIVE
      ACTIVE STATE.
...
IRRM009I (#) REMOTE RRSF NODE PLEX76 PROTOCOL TCP IS IN THE OPERATIVE
      ACTIVE STATE.
...
```

If PLEX75 contained additional systems, repeat the previous process for each one:

- ▶ Add a TCP definition for remote system to IRROPTTP on PLEX76, then load it with **set include(tp)**.
- ▶ Use the **set include(tp)** command on the system PLEX75 matching the definition just added to PLEX76 IRROPTTP.
- ▶ When the conversion is complete, move the newest TCP definition from IRROPTTP to IRROPTxx on PLEX76.

All of the systems in PLEX75 are now using TCP to communicate with PLEX76. The definition for PLEX76 in the PLEX75 IRROPTxx file can be changed from APPC to TCP (Figure 4-14).

```
...
TARGET NODE(PLEX81) SYSNAME(ADCD)          PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))
TARGET NODE(PLEX76) PROTOCOL(TCP(ADDRESS(9.12.4.126))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH6ST2))
...
TARGET NODE(PLEX81) SYSNAME(ADCD) OPERATIVE
TARGET NODE(PLEX76)                OPERATIVE
SET INCLUDE(TP)
```

Figure 4-14 Updated definition for system PLEX76 in IRROPTxx on PLEX75 (changes are in bold)

## 4.14 Earlier system considerations

Now convert communications between PLEX76 and PLEX81. System ADCD is running z/OSV1R11 and does not support TCP communications. This is not a problem for RRSF, as you can use TCP for some communications and APPC for others.

The issue is what to do about the shared IRROPTxx file for node PLEX81. You can craft the file such that it continues to be shared by SC80, SC81, and ADCD. However, certain commands related to TCP fail on ADCD, resulting in error messages in the system log. When SC80 and SC81 connect to remote nodes, they first attempt to connect by using APPC. They then run a protocol conversion to TCP, resulting in some initial inefficiency.

The other option is to maintain one IRROPTxx for SC80 and SC81, and a separate IRROPTxx for system ADCD. Decide whether the cost of maintaining multiple files outweighs the presence of error messages and the extra processing of redundant protocol conversions.

To maintain a separate APPC-only IRROPTxx file for ADCD, the protocol conversion between PLEX76 and PLEX81 is similar to the previous section. Therefore, this example maintains a single, shared IRROPTxx file for all three systems in PLEX81.

To maintain compatibility with an earlier system, make sure that all commands that should only be run by uplevel systems contain the PROTOCOL(TCP) statement. This statement fails outright and cause no change to occur on the earlier system. Also, it helps to fully define and activate the old protocol before starting configuration of the new protocol.

### 4.14.1 Adding TCP to local node definition

The first step in protocol conversion was to update the statements in IRROPTxx to add the TCP protocol to the local node. Because of the way the sample IRROPTxx files were structured, it was easy to add TCP to the local node. However, it is not always as easy. Consider the IRROPTxx file in Figure 4-15, which is structured to have the local nodes defined and made operative on one command.

```
TARGET NODE(PLEX81) SYSNAME(SC81) MAIN LOCAL PREFIX(SYS1.RACF) -  
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC81))) OPER  
TARGET NODE(PLEX81) SYSNAME(ADCD)      LOCAL PREFIX(SYS1.RACF) -  
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD))) OPER
```

Figure 4-15 Sample IRROPTxx snippet that shows declaration of the local node for an MSN

Adding the TCP protocol to the local nodes as described in Figure 4-9 on page 97 requires the existing target commands be broken up. This process is required because the TCP protocol cannot be added to an operative local node. It is tempting to move the operative keyword to the new command which adds the TCP protocol as in Figure 4-16.

```
TARGET NODE(PLEX81) SYSNAME(SC81) MAIN LOCAL PREFIX(SYS1.RACF) -  
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC81)))  
TARGET NODE(PLEX81) SYSNAME(SC81) PROTOCOL(TCP) OPER  
TARGET NODE(PLEX81) SYSNAME(ADCD)      LOCAL PREFIX(SYS1.RACF) -  
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))  
TARGET NODE(PLEX81) SYSNAME(ADCD) PROTOCOL(TCP) OPER
```

Figure 4-16 WRONG way to add TCP protocol to local node in a IRROPTxx file

This process does not work because when the earlier system reads IRROPTxx, it fails on the TCP lines. This failure causes the operative keywords to be skipped, resulting in non-operative local nodes on the earlier system. The correct way is to separate the addition of the TCP protocol from the request to go operative as shown in Figure 4-17.

```
TARGET NODE(PLEX81) SYSNAME(SC81) MAIN LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC81)))
TARGET NODE(PLEX81) SYSNAME(SC81) PROTOCOL(TCP)
TARGET NODE(PLEX81) SYSNAME(SC81) OPER
TARGET NODE(PLEX81) SYSNAME(ADCD) LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))
TARGET NODE(PLEX81) SYSNAME(ADCD) PROTOCOL(TCP)
TARGET NODE(PLEX81) SYSNAME(ADCD) OPER
```

Figure 4-17 Correct way to add TCP protocol to local node.

## 4.14.2 Converting remote nodes

After the local node definition is updated, update communications from APPC to TCP between PLEX76 and SC80 and SC81 as for PLEX75. For brevity, the example converts SC80 and SC81 at the same time.

Update IRROPTTP on PLEX76 and the issue **set include(tp)** to activate it as shown in Figure 4-18.

```
TARGET NODE(PLEX81) SYSNAME(SC81) PROTOCOL(TCP(ADDRESS(9.12.4.47))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH8ST4)) MAIN
TARGET NODE(PLEX81) SYSNAME(SC80) PROTOCOL(TCP(ADDRESS(9.12.4.45))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH8ST2))
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE PROTOCOL(TCP)
TARGET NODE(PLEX81) SYSNAME(SC80) OPERATIVE PROTOCOL(TCP)
```

Figure 4-18 IRROPTTP on PLEX76 defines both SC80 and SC81 in MSN node PLEX81

Update IRROPTTP on PLEX81 and issue **set include(tp)** on both SC80 and SC81 as shown in Figure 4-19.

```
TARGET NODE(PLEX76) PROTOCOL(TCP(ADDRESS(9.12.4.126))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH5ST2))
TARGET NODE(PLEX76) PROTOCOL(TCP) OPERATIVE
```

Figure 4-19 IRROPTTP on PLEX81 defines system PLEX76

The TARGET LISTPROTOCOL command is used to verify that the conversion is complete as shown in Example 4-16.

Example 4-16 TARGET LISTPROTOCOL issued on PLEX76 showing TCP protocol conversion

```
TARGET LISTPROTOCOL
IRRM009I (#) REMOTE RRSF NODE PLEX75 SYSNAME SC74 PROTOCOL TCP IS IN
...
IRRM009I (#) REMOTE RRSF NODE PLEX81 SYSNAME ADCD PROTOCOL APPC IS IN
  THE OPERATIVE ACTIVE STATE.
IRRM009I (#) REMOTE RRSF NODE PLEX81 SYSNAME SC80 PROTOCOL TCP IS IN
```

THE OPERATIVE ACTIVE STATE.  
**IRRM009I (#) REMOTE RRSF NODE PLEX81 SYSNAME SC81 PROTOCOL TCP (MAIN)**  
 IS IN THE OPERATIVE ACTIVE STATE.

---

Move the definitions for SC80 and SC81 into IRROPTxx on PLEX76 as shown in Figure 4-20.

```
TARGET NODE(PLEX76) LOCAL          PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH6ST1)) PROTOCOL(APPC(LUNAME(RRSFSC76)))
TARGET NODE(PLEX76) LOCAL PROTOCOL(TCP)
TARGET NODE(PLEX81) SYSNAME(SC81) PROTOCOL(TCP(ADDRESS(9.12.4.47))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH8ST4)) MAIN
TARGET NODE(PLEX81) SYSNAME(SC80) PROTOCOL(TCP(ADDRESS(9.12.4.45))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH8ST2))
TARGET NODE(PLEX81) SYSNAME(ADCD)    PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))
...
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC80) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(ADCD) OPERATIVE
SET INCLUDE(TP)
```

Figure 4-20 Updated IRROPTxx on PLEX76 with new TCP definitions for SC80 and SC81 highlighted.

All that is left is to add the TCP definition for PLEX76 to IRROPTxx on PLEX81 as shown in Figure 4-21. The APPC definition for PLEX76 needs to remain in this file indefinitely.

```
TARGET NODE(PLEX81) SYSNAME(SC81) MAIN LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC81)))
...
TARGET NODE(PLEX75) SYSNAME(SC74)          PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC74)))
1
TARGET NODE(PLEX76)          PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH6ST1)) PROTOCOL(APPC(LUNAME(RRSFSC76))) 5
2
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE
...
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE
3TARGET NODE(PLEX76)          OPERATIVE
4
SET INCLUDE(TP)
```

Figure 4-21 Excerpts from IRROPTxx on PLEX81 showing where to add new TCP definition

The only place in the previous file where the new TCP definition of PLEX76 cannot go is at or anywhere above **1**. If the TCP definition is placed at **1**, the result is a protocol conversion from TCP to APPC. This conversion occurs because APPC would be defined after TCP, which results in the TCP definition being deleted when APPC connects. For more information, see 4.9, “Order of protocol definition” on page 92.

A logical place to add the new definition is right after the APPC definition at position **2**. This configuration is preferable because it keeps both definitions for PLEX76 close to one another. To be consistent, insert the **target node(plex76) protocol(tcp) operative** command at

location **4**. Setting up the new protocol at location **2** and making it operative at **4** works fine on system ADCD. It fails both new commands, but SC80 and SC81 fail in a subtle way:

- The target operative command at location **3** fails because it does not specify a protocol, and PLEX76 has both APPC and TCP defined.
- Because command **3** failed, the APPC definition of PLEX76 is still in the initial state. You cannot make the new TCP protocol operative unless the old protocol is at least dormant. For more information, see 4.9, “Order of protocol definition” on page 92.

You can try to fix this adding **dormant** to the initial definition of the APPC protocol for PLEX76 at position **5** in Figure 4-21 on page 102. However, that fix will not work because at that point, the local node is not dormant or operative, so **dormant** for a remote node is not yet allowed.

The easiest way to fix this problem is to add **protocol(appc)** to command **3** (Figure 4-22).

```
TARGET NODE(PLEX81) SYSNAME(SC81) MAIN LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC81)))
...
TARGET NODE(PLEX75) SYSNAME(SC74) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC74)))
TARGET NODE(PLEX76) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH6ST1)) PROTOCOL(APPC(LUNAME(RRSFSC76)))
TARGET NODE(PLEX76) PROTOCOL(TCP(ADDRESS(9.12.4.126))) -
PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH5ST2))
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE
...
TARGET NODE(PLEX76) PROTOCOL(APPC) OPERATIVE
TARGET NODE(PLEX76) PROTOCOL(TCP) OPERATIVE
SET INCLUDE(TP)
```

Figure 4-22 Merging the TCP definition for PLEX76 into the IRROPTxx file on PLEX81

Although the placement of the TCP definition for PLEX76 is consistent with the structure of the IRROPTxx file, it took a few tries to get it correct. For example, an attempt was made to add **dormant** to location **5**, with the expectation that it would work, only to have it fail. There were a few deletion and redefinition cycles of the nodes on the test system before the changes were made properly. This aspect conflicts with the original intent of allowing protocol conversion to happen on live systems with no downtime.

The safest (most likely to work the first time) place to add the TCP definition for PLEX76 is at location **4** in Figure 4-21 on page 102. This location is after the APPC protocol is made operative. The old protocol is fully set up and operative. Because all of the new commands specify **protocol(tcp)**, they are ignored by z/OS V1R11 system ADCD. This simplicity and reduced need to test comes at a cost of a less organized IRROPTxx file.

### 4.14.3 Execution of shared IRROPTxx file on current and earlier systems

Either way, when IRROPTxx is run on sc80 and sc81, there is a brief protocol conversion from APPC to TCP each time. Because the APPC definition to SC80 and SC81 does not exist in IRROPTxx on PLEX76, the APPC connection fails each time (Example 4-17).

*Example 4-17 Excerpts from the output of set include(xx) on system SC81*

```
#SET INCLUDE(XX)
...
```

```
IRRC024I (#) RACF REMOTE SHARING CONNECTION TO NODE PLEX76 DID NOT 337
          COMPLETE SUCCESSFULLY. FAILURE OCCURRED WHEN APPC VERB
          ATBSEND WAS ISSUED. RETURN CODE = 22.
```

```
...
```

```
IRRC057I (#) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE
PLEX76 HAS BEEN INITIATED.
```

```
...
```

```
IRRN001I (#) RACF REMOTE SHARING TCP COMMUNICATOR TASK STARTING FOR
NODE PLEX76 WITH HOST ADDRESS 9.12.4.126.
```

```
...
```

```
IRRI027I (#) RACF COMMUNICATION WITH TCP NODE PLEX76 HAS BEEN 742
          SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35
          TLS_RSA_WITH_AES_256_CBC_SHA.
```

```
...
```

```
IRRC058I (#) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE
PLEX76 IS COMPLETE.
```

---

When system ADCD runs the same set **include(xx)**, a few commands fail, but everything connects properly by using APPC (Example 4-18).

*Example 4-18 SET include(xx)*

---

```
#SET INCLUDE(XX)
```

```
IRRG008I (#) RACF SUBSYSTEM IS PROCESSING PARAMETER LIBRARY MEMBER 184
          IRROPTAA.
```

```
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
```

```
...
```

```
IKJ56712I INVALID KEYWORD, TCP
```

```
...
```

```
IKJ56712I INVALID KEYWORD, TCP
```

```
..
```

```
IKJ56712I INVALID KEYWORD, TCP(ADDRESS(9.12.4.126) 191
```

```
IKJ56712I INVALID KEYWORD, )
```

```
IRRM003I (#) RACF SUBSYSTEM TARGET COMMAND ENDED IN ERROR.
```

```
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
```

```
...
```

```
IKJ56712I INVALID KEYWORD, TCP
```

```
IRRM003I (#) RACF SUBSYSTEM TARGET COMMAND ENDED IN ERROR.
```

```
IRRG010I (#) RACF SUBSYSTEM PROCESSING OF PARAMETER LIBRARY MEMBER
          IRROPTAA IS COMPLETE.
```

---

#### 4.14.4 Conversion of communications between MSN PLEX75 and MSN PLEX81

All that is left is to convert communications between PLEX75 and PLEX81. The procedure is similar to what has already been done. The only difference is that there are MSNs on both sides of the conversion. Again, multiple systems are converted at once in the example to save time and space. You can convert one system at a time if you want.

Start by updating IRROPTTP on PLEX75 and issuing **set include(tp)** on both sc75 and sc74 as shown in Figure 4-23.

```
TARGET NODE(PLEX81) SYSNAME(SC81) PROTOCOL(TCP(ADDRESS(9.12.4.47))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH8ST4)) MAIN
TARGET NODE(PLEX81) SYSNAME(SC80) PROTOCOL(TCP(ADDRESS(9.12.4.45))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH8ST4))
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE PROTO(TCP)
TARGET NODE(PLEX81) SYSNAME(SC80) OPERATIVE PROTO(TCP)
```

Figure 4-23 IRROPPTP on PLEX75 with definitions for SC80 and SC81

Update IRROPTTP on PLEX81 to contain definitions for SC74 and SC75 and issue **set include(tp)** from SC80 and SC81 as shown in Figure 4-24.

```
TARGET NODE(PLEX75) SYSNAME(SC75) MAIN PREFIX(RRSF.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(TCP(ADDRESS(9.12.4.72)))
TARGET NODE(PLEX75) SYSNAME(SC74) PREFIX(RRSF.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(TCP(ADDRESS(9.12.4.70)))
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE PROTOCOL(TCP)
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE PROTOCOL(TCP)
```

Figure 4-24 IRROPTTP on PLEX81 with definitions for SC80 and SC81

After communications are successfully established and the commands in IRROPTTP are found to be valid, merge IRROPTTP back into IRROPTxx as before (Figure 4-25). The final **set include(tp)** is removed because you are doing converting.

```
TARGET NODE(PLEX75) SYSNAME(SC75) MAIN LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC75)))
TARGET NODE(PLEX75) SYSNAME(SC75) PROTOCOL(TCP)
TARGET NODE(PLEX75) SYSNAME(SC74) LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC74)))
TARGET NODE(PLEX75) SYSNAME(SC74) PROTOCOL(TCP)
TARGET NODE(PLEX81) SYSNAME(SC81) PROTOCOL(TCP(ADDRESS(9.12.4.47))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH8ST4)) MAIN
TARGET NODE(PLEX81) SYSNAME(SC80) PROTOCOL(TCP(ADDRESS(9.12.4.45))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH8ST4))
TARGET NODE(PLEX81) SYSNAME(ADCD) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))
TARGET NODE(PLEX76) PROTOCOL(TCP(ADDRESS(9.12.4.126))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH6ST2))
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC80) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(ADCD) OPERATIVE
TARGET NODE(PLEX76) OPERATIVE
```

Figure 4-25 IRROPTxx on PLEX75 showing replacement of APPC definitions with TCP definitions

The updates to IRROPTxx on PLEX81 are similar to those made before (Figure 4-26). As was done with PLEX76, keep the APPC definitions for SC74 and SC75 because they are needed by APPC system ADCD.

```

TARGET NODE(PLEX81) SYSNAME(SC81) MAIN LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC81)))
TARGET NODE(PLEX81) SYSNAME(SC81) PROTOCOL(TCP)
TARGET NODE(PLEX81) SYSNAME(SC80) LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFSC80)))
TARGET NODE(PLEX81) SYSNAME(SC80) PROTOCOL(TCP)
TARGET NODE(PLEX81) SYSNAME(ADCD) LOCAL PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH8ST4)) PROTOCOL(APPC(LUNAME(RRSFADCD)))
TARGET NODE(PLEX75) SYSNAME(SC75) MAIN PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC75)))
TARGET NODE(PLEX75) SYSNAME(SC75) MAIN PREFIX(RRSF.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(TCP(ADDRESS(9.12.4.72)))
TARGET NODE(PLEX75) SYSNAME(SC74) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(APPC(LUNAME(RRSFSC74)))
TARGET NODE(PLEX75) SYSNAME(SC74) PREFIX(RRSF.RACF) -
  WORKSPACE(VOLUME(BH5ST2)) PROTOCOL(TCP(ADDRESS(9.12.4.70)))
TARGET NODE(PLEX76) PREFIX(SYS1.RACF) -
  WORKSPACE(VOLUME(BH6ST1)) PROTOCOL(APPC(LUNAME(RRSFSC76)))
TARGET NODE(PLEX76) PROTOCOL(TCP(ADDRESS(9.12.4.126))) -
  PREFIX(RRSF.RACF) WORKSPACE(VOLUME(BH5ST2))
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(SC80) OPERATIVE
TARGET NODE(PLEX81) SYSNAME(ADCD) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE PROTOCOL(APPC)
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE PROTOCOL(APPC)
TARGET NODE(PLEX76) PROTOCOL(APPC) OPERATIVE
TARGET NODE(PLEX75) SYSNAME(SC75) OPERATIVE PROTOCOL(TCP)
TARGET NODE(PLEX75) SYSNAME(SC74) OPERATIVE PROTOCOL(TCP)
TARGET NODE(PLEX76) PROTOCOL(TCP) OPERATIVE

```

Figure 4-26 Shared IRROPTxx on PLEX81 with both APPC and TCP definitions for all systems

The sample configuration is now fully converted to TCP communications where possible. Protocol conversion can be accomplished on a connection by connection basis. Be careful about editing IRROPTxx, especially when it is to be shared with earlier systems. Although the example shows a conversion from APPC to TCP, the process to convert from TCP to APPC is the same.





# Operations

RRSF is a complex product. Many things can cause failures between nodes, whether they are single-system nodes (SSN) or multi-system nodes (MSN). This chapter addresses the more common items to consider when part of an RRSF configuration fails. It includes nodes, revoked users (administrators and the RACF address space), RRSFLIST data sets, INMSG and OUTMSG data sets in extents, expired digital certificates, TCP/IP issues, and PAGENT problems. The chapter also addresses day to day operational items to keep an RRSF configuration operational.

This chapter includes the following sections:

- ▶ Administration
- ▶ RRSFDATA resources
- ▶ SETROPTS and SYSPLEX COMMUNICATION
- ▶ Revoked IDs or IDs revoked at different nodes
- ▶ Certificates and certificate renewal
- ▶ Diagnosing RRSF problems
- ▶ Maintenance and recovery: RRSF data sets

## 5.1 Administration

Administration of RRSF deals with two main commands: **SET** and **TARGET**. **SET** establishes what is allowed to flow, and **TARGET** establishes where the flows are to be directed. You can issue these commands as MVS commands. Remember to use the command prefix facility (CPF) prefix defined for your RACF instance. Or you can write these commands to a member in the RACFPARM data set and use a **SET INCLUDE(xx)** to have the RACF address space run them.

### 5.1.1 SET command

After RRSF is established, consider the whole RRSF network. What flows are established? What commands are you targeting to other nodes? What levels of **OUTPUT** and **NOTIFY** have you set? Use the **SET LIST** command to validate the commands and updates that RRSF propagates to remote nodes.

Example 5-1 allows COMMANDS such as ADDUSER, ALTUSER, and DELUSER **1** to flow out. ROLAND on PLEX75 and PLEX81 get the results in both RRSFLIST data set **2** and as a TSO SEND command **3**.

PASSWORD/PASSPHRASE and APPLICATION changes are not propagated **4**.

*Example 5-1 SET LIST output*

---

```
#SET LIST
IRRH005I (#) RACF SUBSYSTEM INFORMATION:
    TRACE OPTIONS                - NOIMAGE
                                - NOAPPC
                                - NOSYSTEMSSL
                                - NORRSF
                                - NORACROUTE

AUTOMATIC COMMAND DIRECTION IS ALLOWED 1
    OUTPUT IS IN EFFECT FOR:      2
        PLEX75.ROLAND             - ALWAYS
        PLEX81.ROLAND             - ALWAYS
    NOTIFY IS IN EFFECT FOR:      3
        PLEX75.ROLAND             - ALWAYS
        PLEX81.ROLAND             - ALWAYS
AUTOMATIC PASSWORD DIRECTION IS ALLOWED
    OUTPUT IS IN EFFECT FOR:
        PLEX75.ROLAND             - ALWAYS
        PLEX81.ROLAND             - ALWAYS
    NOTIFY IS IN EFFECT FOR:
        PLEX75.ROLAND             - ALWAYS
        PLEX81.ROLAND             - ALWAYS
PASSWORD SYNCHRONIZATION IS *NOT* ALLOWED 4
AUTOMATIC DIRECTION OF APPLICATION UPDATES IS *NOT* ALLOWED
```

---

**Caution:** The SET command allows you to modify the NOTIFY subparameter of the AUTODIRECT parameter. Use NOTIFY(&RACUID) only for those commands that RRSF propagates where the issuer is a TSO user. NOTIFY generates a TSO SEND command. Users without TSO access would create a build-up of undeliverable messages in SYS1.BROADCAST.

The SET command is used to enable certain global RRSF functions for a node:

- AUTOAPPL** Specifies whether automatic direction of application updates is activated.  
**AUTODIRECT** Specifies whether automatic command direction is activated.  
**AUTOPWD** Specifies whether automatic password direction is activated.

The **#SET LIST** command shows the global options enabled at that node. It also shows if any notification output is to be generated and for whom at what node. For initial testing, use OUTPUT(ALWAYS) and NOTIFY(ALWAYS). For normal operations, use OUTPUT(FAIL(node.userid...)) for AUTOAPPL, AUTODIRECT, and AUTOPWD.

In Example 5-2, the **#SET LIST** output includes these parameters:

- ▶ AUTOMATIC COMMAND DIRECTION IS ALLOWED **1**
- ▶ AUTOMATIC PASSWORD DIRECTION IS ALLOWED **2**
- ▶ PASSWORD SYNCHRONIZATION IS ALLOWED **3**

*Example 5-2 SET LIST options output*

---

```
#SET LIST
IRRH005I (#) RACF SUBSYSTEM INFORMATION: 986
TRACE OPTIONS                - NOIMAGE

SUBSYSTEM USERID             - RACF
JESNODE (FOR TRANSMITS)      - WTSCPLX7
AUTOMATIC COMMAND DIRECTION IS ALLOWED 1
  OUTPUT IS IN EFFECT FOR:
    PLEX75.ROLAND            - ALWAYS
    PLEX81.ROLAND            - ALWAYS
  NOTIFY IS IN EFFECT FOR:
    PLEX75.ROLAND            - ALWAYS
    PLEX81.ROLAND            - ALWAYS
AUTOMATIC PASSWORD DIRECTION IS ALLOWED 2
  OUTPUT IS IN EFFECT FOR:
    PLEX75.ROLAND            - ALWAYS
    PLEX81.ROLAND            - ALWAYS
  NOTIFY IS IN EFFECT FOR:
    PLEX75.ROLAND            - ALWAYS
    PLEX81.ROLAND            - ALWAYS
PASSWORD SYNCHRONIZATION IS ALLOWED 3
  OUTPUT IS IN EFFECT FOR:
    ISSUER                    - ALWAYS
  NOTIFY IS IN EFFECT FOR:
    ISSUER                    - ALWAYS
AUTOMATIC DIRECTION OF APPLICATION UPDATES IS *NOT* ALLOWED 4
GENERICANCHOR:
  SYSTEM: COUNT(04)
  JOBNAME: <NONE SPECIFIED>
RACF STATUS INFORMATION:
  TEMPLATE VERSION            - HRF7780 00000150.00000020
  DYNAMIC PARSE VERSION       - HRF7780
```

---

## 5.1.2 TARGET commands (nodes)

**TARGET** either establishes the characteristics of the RACF address space in a system, or provides you with tools to list specifics of that RACF address.

**Tip:** When you issue **TARGET** as an operator command, remember to use the CPF. The example environment uses # as the prefix, but if not set it defaults to **RACF**. You can use the **D OPDATA** command to see what is registered.

Make sure the nodes that you expect are DEFINED and OPERATIVE ACTIVE. The **TARGET NODE LIST** command allows you to see the connections and listener tasks as well as their states. With RRSF over TCP/IP, you have the opportunity to match your NODE names to the SYSPLEXes they service. The individual SYSTEM names match the SYSNAMES under the related NODE names. For single system NODES, generally use the MONOPLEX name as the NODE name. Poorly chosen node names can affect the expansion or addition of new SYSTEMS to that node.

The **TARGET LISTPROTOCOL** command lists the current operational and protocol attributes of one or more nodes and lists the protocol name of each listed node. It is useful if you want to list protocol attributes with summary information for each remote node.

In an MSN, as shown in Example 5-3, the system from which the **#TARGET LISTP** is issued should be in **OPERATIVE ACTIVE** state **1**. Other systems at that node only need to be in the **DEFINED** state **2**. Remember, check the **REMOTE** nodes relative to each **LOCAL** node.

*Example 5-3 TARGET LISTP output*

---

```
#TARGET LISTP
IRRM009I (#) REMOTE RRSF NODE PLEX76 PROTOCOL TCP IS IN THE OPERATIVE
875
ACTIVE STATE.
IRRM009I (#) REMOTE RRSF NODE PLEX81 SYSNAME SC80 PROTOCOL TCP IS IN
THE DEFINED STATE. 2
IRRM009I (#) REMOTE RRSF NODE PLEX81 SYSNAME SC81 PROTOCOL TCP (MAIN)
IS IN THE OPERATIVE ACTIVE STATE. 1
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC74 IS IN THE OPERATIVE
ACTIVE STATE.
IRRM091I (#) - LOCAL NODE TCP LISTENER IS ACTIVE.
IRRM009I (#) LOCAL RRSF NODE PLEX75 SYSNAME SC75 (MAIN) IS IN THE
DEFINED STATE.
```

---

In Example 5-4, the NODE name is PLEX81, which is composed of three systems:

- ▶ ADCD (DEFINED STATE)
- ▶ SC80 (the system where you issued the command)
- ▶ SC81, the MAIN for this NODE

*Example 5-4 Display for PLEX81*

---

```
IRRM009I (#) LOCAL RRSF NODE PLEX81 SYSNAME ADCD IS IN THE DEFINED
STATE.2
IRRM009I (#) LOCAL RRSF NODE PLEX81 SYSNAME SC80 IS IN THE OPERATIVE
ACTIVE STATE. 1
IRRM091I (#) - LOCAL NODE APPC LISTENER IS ACTIVE.
IRRM091I (#) - LOCAL NODE TCP LISTENER IS ACTIVE.
IRRM009I (#) LOCAL RRSF NODE PLEX81 SYSNAME SC81 (MAIN) IS IN THE
DEFINED STATE.2
```

---

### 5.1.3 IP addresses and ports

Example 5-5 shows the **NETSTAT** command, which displays TCP/IP connections and the different ports used. Check your **IRROPTxx TARGET** statements to make sure that you have the correct IP addresses or domain name that resolve to the correct IP addresses. Unless you explicitly over-ride the LOCAL TCP/IP address, you would see RACF “LISTENING” on 0.0.0.0.18136 **1**.

Example 5-5 *NETSTAT output*

---

NETSTAT					
RACF	0000008B	9.12.4.126..1027	9.12.4.45..18136	Establish	
RACF	0000008C	9.12.4.126..1028	9.12.4.47..18136	Establish	
RACF	00000089	9.12.4.126..1026	9.12.4.70..18136	Establish	
RACF	00000088	0.0.0.0..18136	0.0.0.0..0	Listen	<b>1</b>
RACF	0000008D	9.12.4.126..18136	9.12.4.72..1816	Establish	

---

### 5.1.4 PAGENT and Certificates: AT-TLS

Check the **PAGENT** task. If the **PAGENT** task is not active, the RRSF connection fails at initialization or on a **RESTART CONNECTION**. This failure occurs because the Application Transparent Transport Layer Security (AT-TLS) protocol cannot be confirmed as shown in Example 5-6.

Example 5-6 *PAGENT task not active*

---

```
EZD1287I TTLS Error RC: 435 Initial Handshake 924
  LOCAL: 9.12.4.126..1037
  REMOTE: 9.12.4.72..18136
  JOBNAM: RACF RULE: Default_RRSF-Client 1
  USERID: RACF GRPID: 00000002 ENVID: 0000000E CONNID: 00000C59
IRRI031I (#) RRSF CONNECTION TO NODE PLEX75 SYSNAME SC75 HAS BEEN 925
              REJECTED BECAUSE RACF COULD NOT VERIFY AT-TLS POLICY.
              THE BPX1SEL SERVICE DETECTED A SOCKET EXCEPTION.
IRRQ010I (#) RACF REMOTE SHARING TCP CONNECTOR TASK TERMINATING FOR
927
              NODE PLEX75 SYSNAME SC75 WITH HOST ADDRESS 9.12.4.72.
BPXF024I (OMVSKERN) May 22 12:54:47 WTSC76 TTLS 50331704 : 08:54:47
926
TCP/IP    EZD1286I TTLS Error GRPID: 00000002 ENVID: 0000000E CONNID:
00000C59 LOCAL: 9.12.4.126..1037 REMOTE: 9.12.4.72..18136 JOBNAM:
RACF USERID: RACF RULE: Default_RRSF-Client 1 RC: 435 Initial
Handshake 00000000 7ED2FA18
```

---

Periodically verify that your AT-TLS policy is not compromised. The **#SET LIST** command shows you the encryption specification of any TCP/IP connections **1** (Example 5-7).

Example 5-7 *SET LIST output that shows encryption specification*

---

PROTOCOL	-	TCP	
	HOST ADDRESS	-	9.12.4.72
	LISTENER PORT	-	18136
	AT-TLS POLICY:		
	RULE_NAME	-	DEFAULT_RRSF-CLIENT~1
	CIPHER ALG	-	35 TLS_RSA_WITH_AES_256_CBC_SHA <b>1</b>

---

## 5.1.5 RESTART command

You can use the **#RESTART** command to establish a number of RRSF tasks to free up otherwise stopped processes. Example 5-8 provides the options.

*Example 5-8 RESTART output*

---

```
#RESTART
COMMAND
CONNECTION [ NODE(nodename | * ) [ SYSNAME(sysname | * ) ] ] 1
MESSAGE
OUTPUT
RACLINK
RECEIVE
SEND
```

---

Options for the **CONNECTION** are **NODE(nodename)** or **NODE(\*)**. **SYSNAME** is an optional parameter for an MSN with which you can qualify to a single **SYSNAME**. You can also qualify to all **OPERATIVE SYSNAMES** at the designated **NODE**.

## 5.2 RRSFDATA resources

This section addresses the general form for RRSF resources and the items they control. This is provided that the governing **#SET** options are also set. The profiles that cover these resources provide the required granularity so you can selectively control which items flow and who can use these flows. For more information, see *z/OS V1R13.0 Security Server RACF System Programmer's Guide SA22-7681* and *z/OS V1R13.0 Security Server RACF Security Administrator's Guide SA22-7683*.

Check the status of the RRSFDATA class to make sure that you have it **RACLISTED**. Check the access lists. If you enable **CLAUTH** for specific users, remember to include these IDs or groups in the appropriate access lists of the RRSFDATA profiles.

**Explanation:** The **#SET** command determines *what* events are to flow, whereas the **RRSFDATA** resources determine *whose* events are to flow.

### 5.2.1 RRSFDATA RESOURCES related to AUTODIRECT

Table 5-1 shows the general form for RRSF resources and the items they control, provided the governing **#SET** options are also set.

*Table 5-1 RRSFDATA class profiles in relation to RACF commands*

Direct RACF Commands		
ADDUSER	USER	AUTODIRECT.target-node.USER.ADDUSER
ALTUSER	USER	AUTODIRECT.target-node.USER.ALTUSER
CONNECT	USER	AUTODIRECT.target-node.USER.CONNECT

Direct RACF Commands		
DELUSER	USER	AUTODIRECT.target-node.USER.DELUSER
PASSWORD/PH RASE	USER	AUTODIRECT.target-node.USER.PASSWORD
REMOVE	USER	AUTODIRECT.target-node.USER.REMOVE
ADDGROUP	GROUP	AUTODIRECT.target-node.GROUP.ADDGROUP
ALTGROUP	GROUP	AUTODIRECT.target-node.GROUP.ALTGROUP
DELGROUP	GROUP	AUTODIRECT.target-node.GROUP.DELGROUP
ADDSD	DATASET	AUTODIRECT.target-node.DATASET.ADDSD
ALTDSD	DATASET	AUTODIRECT.target-node.DATASET.ALTDSD
DELDSD	DATASET	AUTODIRECT.target-node.DATASET.DELDSD
PERMIT	Resource class/DATASET	AUTODIRECT.target-node.class.PERMIT
RALTER	Resource class	AUTODIRECT.target-node.class.RALTER
RDEFINE	Resource class	AUTODIRECT.target-node.class.RDEFINE
RDELETE	Resource class	AUTODIRECT.target-node.class.RDELETE
RACMAP events <sup>a</sup>	IDIDMAP and USER	AUTODIRECT.target-node.IDIDMAP.APPL.
SETROPTS	RACF (although not a true class)	AUTODIRECT.target-node.RACF.SETROPTS
Password-related changes		
Password changes	USER	AUTODIRECT.target-node.USER.PWSYNC
Passphrase changes	USER	AUTODIRECT.target-node.USER.PHRSSYNC
Application derived changes		
Application Updates	Resource class	AUTODIRECT.target-node.classname.APPL
Application Updates	Modifying VOLSERS for DASD	AUTODASD.target-node.DATASET.APPL
Application updates	Modifying VOLSERS for TAPE	AUTOTAPE.target-node.DATASET.APPL

- a. The RACMAP command updates profiles in the USER and IDIDMAP classes. The RACMAP command itself is not eligible for command direction. It cannot be used with the AT or ONLY keyword, but it can be run on remote nodes by using other methods. Updates to profiles in these classes by the RACMAP command are eligible for automatic direction of application updates.

## 5.2.2 CLAUTH

Check your user community to see what users have CLAUTH authority and to which classes. Verify that these users have READ access to the appropriate **AUTODIRECT.target-node.CLAUTH-class.RDEFINE** profiles. This process is important for

help desks (CLAUTH to USER) and areas where you grant specific CLAUTH authority to IBM CICS®, IBM IMS™, IBM DB2®, or other administrators.

### 5.2.3 FIELD level access

When **FIELD** level access is active, consider the classes involved and grant READ access to the appropriate **AUTODIRECT.node.class.APPL** resource in the RRSFDATA class. There is no granularity at the RRSFDATA level, so the FIELD class profiles at all NODES should match when you have FIELD level access controls.

### 5.2.4 RRSF Considerations for Automatic ID Assignment

The IBM stated direction is that z/OS 1.13 will be the last z/OS release to support **BPX.DEFAULT.USER**. You need to consider two key issues with **BPX.NEXT.USER** and **BPX.UNIQUE.USER** profiles.

For **BPX.NEXT.USER**, set up a *unique* range at each **NODE** (shared RACF database) so that these ranges do not overlap. The format of the **BPX.NEXT.USER** profile is shown in Example 5-9.

*Example 5-9 BPX.NEXT.USER format*

---

```
RDEFINE FACILITY BPX.NEXT.USER
  APPLDATA('10000-19999/10000-19999') 1
  ONLYAT(.MYID)
```

---

**APPLDATA('UIDstart–UIDend/GIDstart–GIDend')** 1 sets the range of **UIDS** that are to be used first followed by the range of **GIDS** to be used. Example 5-10 shows where AUTOMATIC UID 1 and GID 2 assignment is allowed.

*Example 5-10 Profile definitions*

---

```
SET AUTOAPPL
RDEFINE RRSFDATA AUTODIRECT.*.USER.APPL UACC(READ) 1
RDEFINE RRSFDATA AUTODIRECT.*.GROUP.APPL UACC(READ) 2
```

---

The **#SET** command enables automatic application updates. The two RRSFDATA profiles allow users to automatically acquire the same **UID/GID** as is set when they first use a function that requires a *dubbed* process.

**Guideline:** Assign a unique UID for each user and a unique GID for each group that needs access to z/OS UNIX functions and resources. Avoid assigning shared or default UNIX identities.

When you have many existing users who need access to UNIX services such as FTP, use the **BPX.UNIQUE.USER** so that RACF assigns a unique UID/GID from **BPX.NEXT.USER**.

## 5.3 SETROPTS and SYSPLEX COMMUNICATION

Based on your RRSFDATA profiles and the **#SET** options, RRSF might automatically propagate new class definitions and the associated **SETR RACLIST** or **SETR RACLIST(classname) REFRESH** to a remote node. However, that command runs only at the



MAIN identified at the REMOTE node. Therefore, SYSPLEX COMMUNICATION needs to be enabled for MSNs. Without SYSPLEX COMMUNICATION, some of the benefit of RRSF is lost.

Also, establish SYSPLEX COMMUNICATION through your **ICHRDSNT (RACF DATA SET NAME TABLE)**. An **RVARY LIST** shows you the status as shown in Example 5-11.

*Example 5-11 RVARY LIST output*

---

```
RVARY LIST
IRRA011I (#) OUTPUT FROM RVARY:
ICH15013I RACF DATABASE STATUS:
ACTIVE USE  NUM VOLUME  DATASET
-----
YES  PRIM   1 BH8CAT   SYS1.RACFESA
YES  BACK   1 BH8ST1   SYS1.RACFBK
MEMBER SC81      IS SYSPLEX COMMUNICATIONS ENABLED & IN NON-DATA
SHARING MODE.1
ICH15020I RVARY COMMAND HAS FINISHED PROCESSING.
```

---

Example 5-11 shows that you have **SYSPLEX COMMUNICATIONS ENABLED 1**.

## 5.4 Revoked IDs or IDs revoked at different nodes

Although the RACF address space is often defined as TRUSTED, most communication tasks do not inherit this TRUST attribute. Check the status of the RACF user ID to confirm that it is not REVOKED. The LU command for the specific user would show this status. Remember to check at each node to confirm that it is not REVOKED.

Check your ADMINISTRATOR IDs when you enable AUTOMATIC COMMAND DIRECTION. Confirm that these ids have like attributes (SPECIAL/OPERATIONS/AUDITOR) at each of the nodes in your RRSF network.

If you enabled automatic PASSWORD and PASSPHRASE DIRECTION, verify that the IDs are not revoked for those IDs with this capability enabled. Revoked IDs cause the automatic updates to fail at those nodes where the associated IDs exist.

Example 5-12 shows a failed command where the ID at the REMOTE node is in a REVOKED state **1**.

*Example 5-12 ID at remote node is revoked*

---

```
SETR issued at 13:41:30 on 05/21/12 was *not* processed at PLEX75.ROLAND

COMMAND ISSUED: SETR      LIST

ERROR INFORMATION:
IRRC010I UNABLE TO ESTABLISH RACF ENVIRONMENT FOR COMMAND SETR.
IRRC021I ACCESS HAS BEEN REVOKED FOR USER ID ROLAND. 1
=====
```

---

## 5.5 Certificates and certificate renewal

When you implement RRSF over TCP/IP, you must implement a trust policy based on digital certificates to allow TCP/IP communication to take place between RRSF nodes. RRSF node connections that use TCP/IP are protected by using AT-TLS. This trust policy is based on the requirements of AT-TLS. It requires that you create a RACF key ring for each node and one or more signed server certificates. These items were created in 3.4, “Creating digital certificates and key rings” on page 46.

Various certificates can be used to define the encryption used between RRSF nodes:

<b>Server certificates</b>	Each RRSF server has its own server certificate. With an MSN, local members (RRSF systems) can share a certificate and key ring.
<b>CA certificate</b>	The Certificate Authority (CA) certificate that contains the public key of the CA must be stored in the RRSF server key ring.

**Remember:** All times used in the RACDCERT commands are stored in the certificates as Coordinated Universal Time values.

Eventually, the default RACF certificates in use by your RRSF nodes will expire. At that point, you can delete and redefine them, or re-sign them in place with new expiration dates. There are fewer steps to re-sign them than to delete and redefine them. This topic addresses how to re-sign them.

Various scenarios are provided to help you identify SSL connection issues that occur when certificates expire. This section addresses the following scenarios:

1. How to renew an RRSF server certificate before it expires.

This scenario includes what failures you get, and the implementation steps to correct these failures after an RRSF server certificate has expired:

- Symptoms
- Error messages
- SSL return codes, error codes, and reason codes

2. Rekeying a certificate

3. How to renew CA AUTH certificate, and re-sign all RRSF server certificates after the CA certificate used to sign them expires (if using an internal CA).

This scenario includes what failures you get, and what steps to implement after a CA certificate expires:

- Symptoms
- Error messages
- SSL return codes, error codes, and reason codes

## 5.5.1 Renewing an expiring RRSF server certificate issued by a local CA

Perform these steps to renew an expiring certificate by using a private key. Employ the same private key used when the certificate was generated by RACF and issued by a local CA. In this example, the expiring certificate was signed by a CERTAUTH certificate labeled Local RACF CA 1.

1. Identify the certificate used by your node.
2. Create a certificate request based on the expiring certificate and store it in the MVS data set SYSADM.CERT.REQ 2 by running the following command:

```
RACDCERT ID(RACF) GENREQ(LABEL('Local RACF CA')) 1  
          DSN('SYSADM.CERT.REQ') 2
```

3. Renew and replace the existing certificate by running the following command:

```
RACDCERT ID(RACF) GENCERT('SYSADM.CERT.REQ') 2  
          SIGNWITH(CERTAUTH LABEL('Local RACF CA')) 1
```

You have now renewed a certificate that was signed by a local certificate authority by using the same private key. All information in the certificate is updated to reflect the renewal, including the key ring connection information.

## 5.5.2 Renewing an expired RRSF server certificate

This section addresses the symptoms and corrective action when an RRSF server certificate is expired. In this scenario, the RRSF server is on the same node as the internal CA. The RRSF server certificate was generated by an internal CA, and the internal CA (RACF) is on the same node as the RRSF server with the expired certificate.

This section addresses these issues:

- ▶ Both the local node and remote node symptoms and associated messages
- ▶ The SSL return and reason codes
- ▶ The steps required to renew the certificate with the changed NOTAFTER date

Example 5-13 shows the key ring and expired certificate. List the key ring with the name IRR.RRSF.KEYRING 1 to see the user certificate 2.

*Example 5-13 List KEYRING*

---

```
RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING) 1
```

Digital ring information for user RACF:

Ring:

>IRR.RRSF.KEYRING<

Certificate Label Name	Cert Owner	USAGE	DEFAULT
-----	-----	-----	-----
RRSF CERTAUTH CERT	CERTAUTH	CERTAUTH	NO
SC75 RRSF SERVER CERTIFICATE	ID(RACF)	PERSONAL	NO
SC75 SOON RRSF SERVER CERT	ID(RACF)	PERSONAL	YES 2

---

List the certificates by using the CERTIFICATE LABEL NAME to get the end date of the named certificate as shown in Example 5-14.

- ▶ SC75 SOON RRSF SERVER CERT **3** is the default certificate for SC75, with an End date of 2012/05/16 19:30:00 (expired) **4**
- ▶ RRSF CERTAUTH CERT **5** is the CERTAUTH certificate, with an End Date of 2018/05/14 16:30:00 **6**

*Example 5-14 Certificate information*

---

```
RACDCERT ID(RACF) LIST(LABEL('SC75 SOON RRSF SERVER CERT')) 3
```

Digital certificate information for user RACF:

```
Label: SC75 SOON RRSF SERVER CERT
...
Start Date: 2012/01/01 00:00:00
End Date: 2012/05/16 19:30:00 4
Serial Number:
...
Ring Associations:
  Ring Owner: RACF
  Ring:
    >IRR.RRSF.KEYRING<
```

---

```
RACDCERT CERTAUTH LIST(LABEL('RRSF CERTAUTH CERT')) 5
```

Digital certificate information for CERTAUTH:

```
Label: RRSF CERTAUTH CERT
...
Start Date: 2012/01/01 00:00:00
End Date: 2018/05/14 16:30:00 6
Serial Number:
  >10<
Issuer's Name:
...
Private Key: YES
Ring Associations:
  Ring Owner: RACF
  Ring:
```

---

On an RRSF start or a RESTART CONNECTION, you see the output shown in Example 5-15.

*Example 5-15 Connection attempt with expired certificate*

---

```
....
....
IRRB020I (#) IRRTCP00 TASK HAS BEEN RESTARTED. 944
...
IRRJ000I (#) RACF RACF LOCAL NODE TRANSACTION PROGRAM STARTING
IRRC054I (#) RACF REMOTE SHARING TCP LISTENER HAS BEEN SUCCESSFULLY
ESTABLISHED.
```

```

IRRQ001I (#) RACF REMOTE SHARING TCP CONNECTOR TASK STARTING FOR NODE PLEX76
WITH HOST ADDRESS 9.12.4.126.
IRRI000I (#) LOCAL RACF NODE PLEX75 SYSNAME SC75 IS ATTEMPTING TO 950
CONTACT PARTNER RACF NODE PLEX76.
...
IRRI031I (#) RRSF CONNECTION TO NODE PLEX76 HAS BEEN REJECTED BECAUSE
969
RACF COULD NOT VERIFY AT-TLS POLICY. THE BPX1SEL SERVICE
DETECTED A SOCKET EXCEPTION.
IRRI031I (#) RRSF CONNECTION TO NODE PLEX81 SYSNAME SC81 HAS BEEN 970
REJECTED BECAUSE RACF COULD NOT VERIFY AT-TLS POLICY.
THE BPX1SEL SERVICE DETECTED A SOCKET EXCEPTION.
IRRQ010I (#) RACF REMOTE SHARING TCP CONNECTOR TASK TERMINATING FOR
NODE PLEX76 WITH HOST ADDRESS 9.12.4.126.
IRRQ010I (#) RACF REMOTE SHARING TCP CONNECTOR TASK TERMINATING FOR
NODE PLEX81 SYSNAME SC81 WITH HOST ADDRESS 9.12.4.47.
BPXF024I (TCP/IP) May 17 12:23:12 TTLSY84017879~: 08:23:12 TCP/IP 972
EZD1286I TTLS Error GRPID: 0000000C ENVID: 0000002F CONNID: 00006D13
LOCAL: 9.12.4.72..1638 REMOTE: 9.12.4.45..18136 JOBNAME: RACF USERID:
RACF RULE: Default_RRSF-Client~1 RC: 503 Initial Handshake 00000000
7EC60018
BPXF024I (TCP/IP) May 17 12:23:12 TTLSY84017879~: 08:23:12 TCP/IP 973
EZD1286I TTLS Error GRPID: 0000000C ENVID: 0000002F CONNID: 00006D12
LOCAL: 9.12.4.72..1637 REMOTE: 9.12.4.126..18136 JOBNAME: RACF
USERID: RACF RULE: Default_RRSF-Client~1 RC: 7 Initial Handshake
00000000 7EC60018
BPXF024I (TCP/IP) May 17 12:23:12 TTLSY84017879~: 08:23:12 TCP/IP 974
EZD1286I TTLS Error GRPID: 0000000C ENVID: 0000002F CONNID: 00006D14
LOCAL: 9.12.4.72..1639 REMOTE: 9.12.4.47..18136 JOBNAME: RACF USERID:
RACF RULE: Default_RRSF-Client~1 RC: 503 Initial Handshake 00000000
7EC60018

```

---

On a REMOTE MAIN NODE, you see output similar to that in Example 5-16.

*Example 5-16 REMOTE MAIN NODE with expired certificate*

---

```

IRRC062I (#) RACF REMOTE SHARING CONNECTION TO NODE PLEX75 SYSNAME SC75 CLOSED
BY REQUEST OF THE REMOTE NODE.
IRRC062I (#) RACF REMOTE SHARING CONNECTION TO NODE PLEX75 SYSNAME SC75 CLOSED
BY REQUEST OF THE REMOTE NODE.
IRRN010I (#) RACF REMOTE SHARING TCP COMMUNICATOR TASK TERMINATING
FOR NODE PLEX75 SYSNAME SC75 WITH HOST ADDRESS 9.12.4.72.
IRRN010I (#) RACF REMOTE SHARING TCP COMMUNICATOR TASK TERMINATING
FOR NODE PLEX75 SYSNAME SC75 WITH HOST ADDRESS 9.12.4.72.
IRRI031I (#) RRSF CONNECTION FROM PEER 9.12.4.72:1638 HAS BEEN 561
REJECTED BECAUSE RACF COULD NOT VERIFY AT-TLS POLICY.
THE BPX1SEL SERVICE DETECTED A SOCKET EXCEPTION.
BPXF024I (SYSLOGD) May 17 12:23:12 WTSC81 TTLSY84018984~: 08:23:12 441
TCP/IP EZD1281I TTLS Map CONNID: 000070C0 LOCAL: 9.12.4.47..18136
REMOTE: 9.12.4.72..1639 JOBNAME: RACF USERID: RACF TYPE: InBound
STATUS: Enabled RULE: Default_RRSF-Server~2 ACTIONS: gAct1
eAct2~RRSF-Server cAct2~RRSF-Server
EZD1287I TTLS Error RC: 403 Initial Handshake 560
LOCAL: 9.12.4.45..18136
REMOTE: 9.12.4.72..1638
JOBNAME: RACF RULE: Default_RRSF-Server~2

```

```

        USERID: RACF GRPID: 0000000C ENVID: 00000029 CONNID: 000637B6
EZD1287I TTLS Error RC: 403 Initial Handshake 442
LOCAL: 9.12.4.47..18136
REMOTE: 9.12.4.72..1639
JOBNAME: RACF RULE: Default_RRSF-Server~2
        USERID: RACF GRPID: 00000003 ENVID: 00000025 CONNID: 000070C0
IRRI031I (#) RRSF CONNECTION FROM PEER 9.12.4.72:1639 HAS BEEN 443
        REJECTED BECAUSE RACF COULD NOT VERIFY AT-TLS POLICY.
        THE BPX1SEL SERVICE DETECTED A SOCKET EXCEPTION.
BPXF024I (SYSLOGD) May 17 12:23:12 WTSC81 TTLSY84018984": 08:23:12 444
TCP/IP EZD1283I TTLS Event GRPID: 00000003 ENVID: 00000025 CONNID:
000070C0 RC: 403 Initial Handshake 00000000 7E835318
BPXF024I (SYSLOGD) May 17 12:23:12 WTSC81 TTLSY84018984": 08:23:12 445
TCP/IP EZD1286I TTLS Error GRPID: 00000003 ENVID: 00000025 CONNID:
000070C0 LOCAL: 9.12.4.47..18136 REMOTE: 9.12.4.72..1639 JOBNAME:
RACF USERID: RACF RULE: Default_RRSF-Server~2 RC: 403 Initial
Handshake 00000000 7E835318

```

---

On PLEX76, a single system node, the output is shown in Example 5-17.

*Example 5-17 Single System Node Messages for expired certificate*

```

IRRI031I (#) RRSF CONNECTION FROM PEER 9.12.4.72:1637 HAS BEEN 042
        REJECTED BECAUSE RACF COULD NOT VERIFY AT-TLS POLICY.
        THE BPX1SEL SERVICE DETECTED A SOCKET EXCEPTION.
EZD1287I TTLS Error RC: 403 Initial Handshake 041
LOCAL: 9.12.4.126..18136
REMOTE: 9.12.4.72..1637
JOBNAME: RACF RULE: Default_RRSF-Server~2
        USERID: RACF GRPID: 00000005 ENVID: 00000022 CONNID: 000188BF
BPXF024I (IBMUSER) May 17 12:23:12 WTSC76 TTLSY50332005": 08:23:12 043
TCP/IP EZD1286I TTLS Error GRPID: 00000005 ENVID: 00000022 CONNID:
000188BF LOCAL: 9.12.4.126..18136 REMOTE: 9.12.4.72..1637 JOBNAME:
RACF USERID: RACF RULE: Default_RRSF-Server~2 RC: 403 Initial
Handshake 00000000 7EB48F18

```

---

The expired certification causes these symptoms:

- ▶ From SC75 (local) to SC81 (MSN remote): handshake failure rc 503
- ▶ From SC75 (local) to SC76 (Single node remote): handshake failure rc 7
- ▶ From SC81 (local) to SC75 (MSN remote): handshake failure rc 403

Renew the certificate to a non-expired future date by performing these steps:

1. Create a certificate request:

```

RACDCERT ID(RACF) GENREQ(LABEL('SC75 OK RRSF SERVER CERTIFICATE')) +
        DSN('RRSF.SC75.REQ.SOONDATE') FORMAT(CERTB64)

```

2. Create a certificate with a non-expired date as shown in Example 5-18.

*Example 5-18 Creation of new certificate with non-expired date.*

```

RACDCERT ID(RACF) GENCERT('RRSF.SC75.REQ.SOONDATE') -
        NOTBEFORE(DATE(2012-01-01)) -
        NOTAFTER(DATE(2017-05-16) TIME(19:30:00)) -
        WITHLABEL('SC75 OK RRSF SERVER CERT') -
        SIGNWITH(CERTAUTH LABEL('RRSF CERTAUTH CERT'))

```

---

The **NOTBEFORE**(DATE(yy-yy-mm-dd) TIME(hh:mm:ss)) command specifies the local date and time from which the certificate is valid. If DATE is not specified, it defaults to the current local date. If TIME is not specified, it defaults to TIME(00:00:00).

The **NOTAFTER**(DATE(yy-yy-mm-dd) TIME(hh:mm:ss)) command specifies the local date and time after which the certificate is no longer valid. If RACDCERT GENCERT DATE is not specified, it defaults to one year from the NOTBEFORE date value. If TIME is not specified, it defaults to TIME(23:59:59). The NOTBEFORE value must be earlier than the NOTAFTER value or an informational message is issued.

If you do not specify NOTBEFORE and NOTAFTER, by default the new extension is for one year only. You will need to repeat this process next year.

**Caution:** The LABEL is limited to 32 characters. If your label has more than 32 characters, you get the following error messages:

```
INVALID STRING, 'SC75 SOON RRSF SERVER CERTIFICATE'
MISSING LABEL NAME IN QUOTES+
MISSING The Label Name for the new Digital Certificate, in quotes
```

3. Add this certificate to the keyring as the default certificate. List the key ring and certificate as shown in Example 5-19.

Example 5-19 Verification of renewed certificates

```
RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)
RACDCERT ID(RACF) LIST
RACDCERT ID(RACF) LIST(LABEL('SC75 OK RRSF SERVER CERT'))
```

Command outputs are displayed as shown in Example 5-20. The key values are highlighted in bold.

Example 5-20 Partial RACDCERT output

```
Ring:
      >IRR.RRSF.KEYRING<
Certificate Label Name      Cert Owner      USAGE      DEFAULT
-----
...
SC75 OK RRSF SERVER CERT      ID(RACF)      PERSONAL      YES
```

```
RACDCERT ID(RACF) LIST
```

```
Digital certificate information for user RACF:
...
Label: SC75 OK RRSF SERVER CERT
Certificate ID: 2QTZwcPG4sP39UDW0kDZ2eLGQOLF2eXF2UDDxdnj
Status: TRUST
Start Date: 2012/01/01 00:00:00
End Date: 2017/05/16 19:30:00
```

4. Even after renewing or re-creating a new certificate, the RRSF connections are not in the active state, so the connections must be restarted.

On SC75 (the local MAIN node), issue the **#TARGET OPERATIVE** command either by manually issuing it or having it issued through the RACF parmlib member. In this example, make the node operative by issuing the **#SET INCLUDE(01) command**. This process causes

RACF to process the 01 member of the RACF parmlib data set, which includes the **#TARGET OPERATIVE** command. Validate your certificate extension as in Example 5-21.

*Example 5-21 Using IRROPT01 to activate a node after certificate renewal*

```
...  
IRRI027I (#) RACF COMMUNICATION WITH TCP NODE PLEX76 HAS BEEN 194  
SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35  
TLS_RSA_WITH_AES_256_CBC_SHA.  
IRRN001I (#) RACF REMOTE SHARING TCP COMMUNICATOR TASK STARTING FOR NODE  
PLEX81 SYSNAME SC80 WITH HOST ADDRESS 9.12.4.45.  
IRRI027I (#) RACF COMMUNICATION WITH TCP NODE PLEX81 SYSNAME SC80 HAS BEEN  
SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35  
TLS_RSA_WITH_AES_256_CBC_SHA.  
IRRN001I (#) RACF REMOTE SHARING TCP COMMUNICATOR TASK STARTING FOR NODE  
PLEX81 SYSNAME SC81 WITH HOST ADDRESS 9.12.4.47.  
IRRI027I (#) RACF COMMUNICATION WITH TCP NODE PLEX81 SYSNAME SC81 HAS BEEN  
SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35  
TLS_RSA_WITH_AES_256_CBC_SHA.
```

Issue the appropriate **#TARGET LISTP** commands at each node to verify that the systems are in following states:

- MSN MAIN systems are in the OPERATIVE state
- All Non MAIN systems are in the DEFINED state
- All single nodes are in the OPERATIVE state

This is the normal mode of operations.

**Consideration:** Using **#SET INCLUDE(01)** with the original IRROPT01 member to rebuild the connections simplifies the whole process. However, it causes many error messages like the following to be issued:

```
IRRM016I (#) RACF SUBSYSTEM TARGET COMMAND CANNOT CHANGE THE PROTOCOL  
INFORMATION FOR NODE PLEX81 SYSNAME SC81 WHILE IT IS  
OPERATIVE.  
IRRM003I (#) RACF SUBSYSTEM TARGET COMMAND ENDED IN ERROR.
```

You can disregard these error messages, or you can use the **#RESTART CONNECTION** command.

### 5.5.3 RRSF remote node server certificate expired

This section deals with when an RRSF server certificate on a remote node (the internal CA certificate is on a different RACF database) is expired.

It addresses the symptoms at both the local node and any remote nodes:

- ▶ Messages
- ▶ SSL return and reason codes

It then identifies the steps necessary to renew the certificate and specify a new **NOTAFTER** date.

Recycle the policy agent (PAGENT) so that its policy is refreshed, including the key ring and SSL information that it maintains in storage. This might be the case when a remote system is IPLed or otherwise recycles its PAGENT address space.



When you restart the connections, you see output as shown in Example 5-22.

*Example 5-22 Restart of PAGENT at REMOTE NODE*

---

```
#RESTART CONNECTION
...
IRRBO20I (#) IRRDDM00 TASK HAS BEEN RESTARTED. 622
IRRC051I (#) RACF REMOTE SHARING TCP LISTENER TASK STARTING. 623
...
IRRI000I (#) LOCAL RACF NODE PLEX76 IS ATTEMPTING TO CONTACT PARTNER
RACF NODE PLEX75 SYSNAME SC75.
IRRI031I (#) RRSF CONNECTION TO NODE PLEX81 SYSNAME SC80 HAS BEEN REJECTED
BECAUSE RACF COULD NOT VERIFY AT-TLS POLICY.
THE BPX1SEL SERVICE DETECTED A SOCKET EXCEPTION.
EZD1287I TTLS Error RC: 503 Initial Handshake 650
LOCAL: 9.12.4.126..2893
REMOTE: 9.12.4.45..18136
  JOBNAME: RACF RULE: Default_RRSF-Client"1
  USERID: RACF GRPID: 00000006 ENVID: 00000036 CONNID: 00018BDE
  IRRQ010I (#) RACF REMOTE SHARING TCP CONNECTOR TASK TERMINATING FOR NODE PLEX81
  SYSNAME SC80 WITH HOST ADDRESS 9.12.4.45.
..
```

---

The expired certification causes these symptoms:

- ▶ From SC76 (single node) to SC75 (MAIN MSN remote): handshake failure rc 503
- ▶ From SC76 (single node) to SC765 (non-MAIN MSN remote): handshake failure rc 503
- ▶ From SC76 (single node) to SC81 (MAIN MSN remote): handshake failure rc 7
- ▶ From SC76 (single node) to SC80 (non-MAIN MSN remote): handshake failure rc 503

To renew the certificate to a non-expired future date, perform these steps:

**Remember:** The steps are different than in the previous scenario. The remote RRSF server is not on the same node as the internal CA that issued the certificate. The certificate request must be sent to the system where the internal CA is.

1. Create a certificate request

```
RACDCERT ID(RACF) GENREQ(LABEL('SC76 RRSF NOOK CERTIFICATE')) -
  DSN('RRSF.SC76.REQ.OKDATE') FORMAT(CERTB64)
/*
```

2. Export this certificate request to the CA Host(SC75) in ASCII.

3. Sign the certificate request and extend its validity to a new date:

```
RACDCERT ID(RACF) GENCERT('RRSF.SC76.REQ.OKDATE') -
  NOTBEFORE(DATE(2012-01-01)) -
  NOTAFTER(DATE(2017-05-17)) -
  WITHLABEL('SC76 RRSF OK') -
  SIGNWITH(CERTAUTH LABEL('RRSF CERTAUTH CERT'))
```

4. Still on the CA host (SC75), export this newly signed certificate to a new data set:

```
RACDCERT ID(RACF) -
  EXPORT(LABEL('SC76 RRSF OK')) -
  DSN('RRSF.SC76.OK.CERT') FORMAT(PKCS7B64)
```

```
/* NOW DELETE THIS CERTIFICATE AS NO LONGER NEEDED */
```

```
RACDCERT ID(RACF) -  
DELETE(LABEL('SC76 RRSF OK'))
```

5. Copy this data set to the remote host (SC76) in ASCII mode.

6. On the remote node SC76, add the renewed certificate:

```
RACDCERT ID(RACF) -  
ADD('RRSF.SC76.OK.CERT')  
SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH
```

7. Add the certificate to the keyring as the default.

8. List the certificate and key ring to verify the new **NOTAFTER** date.

9. Finally, restart the RRSF connections from SC76 as shown in Example 5-23.

---

*Example 5-23 RESTART of SC76*

---

```
#RESTART CONNECTION  
...  
...  
IRRC054I (#) RACF REMOTE SHARING TCP LISTENER HAS BEEN SUCCESSFULLY  
ESTABLISHED.  
  
#TARGET LISTP  
IRRM009I (#) REMOTE RRSF NODE PLEX75 SYSNAME SC74 PROTOCOL TCP IS IN  
THE OPERATIVE PENDING VERIFICATION STATE.
```

---

As the connections are still in a pending state, issue the **TARGET ACTIVE** commands again from IRROPT01 as shown in Example 5-24.

---

*Example 5-24 SET INCLUDE(01)*

---

```
#SET INCLUDE(01)  
...  
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.  
IEF196I IGD103I SMS ALLOCATED TO DDNAME SYS00121  
IRRQ001I (#) RACF REMOTE SHARING TCP CONNECTOR TASK STARTING FOR NODE PLEX75  
SYSNAME SC75 WITH HOST ADDRESS 9.12.4.72.  
IRRI000I (#) LOCAL RACF NODE PLEX76 IS ATTEMPTING TO CONTACT PARTNER  
RACF NODE PLEX75 SYSNAME SC75.  
IEF196I IGD104I HFS FILE WAS RETAINED, DDNAME IS (SYS00121)  
IEF196I FILENAME IS (/etc/resolv.conf)  
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.  
IEF196I IGD103I SMS ALLOCATED TO DDNAME SYS00124  
IRRQ001I (#) RACF REMOTE SHARING TCP CONNECTOR TASK STARTING FOR NODE PLEX75  
SYSNAME SC74 WITH HOST ADDRESS 9.12.4.70.  
IRRI000I (#) LOCAL RACF NODE PLEX76 IS ATTEMPTING TO CONTACT PARTNER  
RACF NODE PLEX75 SYSNAME SC74.  
IRRM002I (#) RACF SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.  
IRRC055I (#) RACF REMOTE SHARING TCP LISTENER IS TERMINATING. 002  
IRRJ001I (#) RACF RACF LOCAL NODE TRANSACTION PROGRAM COMPLETED UNDER USER ID  
RACF GROUP SYS1.  
IRRC054I (#) RACF REMOTE SHARING TCP LISTENER HAS BEEN SUCCESSFULLY  
ESTABLISHED.
```

---

## 5.5.4 Renewing (rekeying) a certificate with a new private key

When you renew a certificate by using a new private key, retire the private key and replace it with a new one. This process is called *certificate rekeying* or *key rollover*. This option prevents the overuse of a private key. The more a key is used, the more susceptible it is to being *hacked*.

All information in the renewed certificate is updated to reflect the renewal, including the key ring connection information. After you retire and replace the old certificate, you can begin to use the new certificate and its private key. You can continue to use the old, retired certificate until it expires to verify previously generated signatures. However, you cannot use the retired certificate to sign new certificates. Additionally, do not connect the retired certificate to any key rings as the default certificate.

Rekey and rollover a private key by using the REKEY and ROLLOVER operands of the RACDCERT command. The REKEY operand makes a self-signed copy of the original certificate with a new public-private key pair. The ROLLOVER operand finalizes the rekey operation by replacing the use of the original certificate with the new certificate. It does so in every key ring to which the original certificate is connected. It also deletes the original private key and copies over the information about its serial number base in case the certificate was being used to sign new certificates.

In the following procedures, the local certificate authority issued the expiring certificate. Replace the private key with a new one.

Rekey the certificate associated with the user ID RACF (our RRSF server) with label 'SC75 OK RRSF SERVER CERTKR'. The certificate was issued by the CERTAUTH certificate with label 'RRSF CERTAUTH CERT1' that was generated by RACF. To rekey a certificate issued by a local CA and replace the private key, perform these steps

1. List the RRSF server certificate and CERTAUTH certificates:

```
RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)
RACDCERT ID(RACF) LIST
RACDCERT CERTAUTH LIST
```

Validate the output to confirm that you will manipulate the expected certificate.

2. Initiate the rekeying by running the following RACF command:

```
RACDCERT ID(RACF) REKEY(LABEL('SC75 OK RRSF SERVER CERTKR')) -
  WITHLABEL('SC75 OK RRSF SERVER CERT')
```

- **SC75 OK RRSF SERVER CERT** is the new certificate label.
- **SC75 OK RRSF SERVER CERTKR** is the current certificate label you are rekeying.

3. Create a certificate request based on the new self-signed certificate and store it in an MVS data set 'RRSF.SC75.KEYR' by running the following command:

```
RACDCERT ID(RACF) GENREQ(LABEL('SC75 OK RRSF SERVER CERT')) +
  DSN('RRSF.SC75.KEYR') FORMAT(CERTB64)
```

4. Sign the new certificate by running the following command:

```
RACDCERT ID(RACF) GENCERT('RRSF.SC75.KEYR') -
  SIGNWITH(CERTAUTH LABEL('RRSF CERTAUTH CERT1'))
```

Issuing a RACDCERT ID(RACF) LIST command shows that a new certificate named SC75 OK RRSF SERVER CERT has been created as shown in Example 5-25. However, it has no keyring associated with it.

*Example 5-25 RACDCERT ID(RACF) LIST*

---

```
READY
RACDCERT ID(RACF) LIST

Digital certificate information for user RACF:

  Label: SC75 OK RRSF SERVER CERTKR
  ...
  Ring Associations:
    Ring Owner: RACF
    Ring:
      >IRR.RRSF.KEYRING<

READY

  Label: SC75 OK RRSF SERVER CERT
  Subject's Name:
    >CN=RRSF SERVER.OU=IBM ITSO POK.O=IBM.C=US<
  Key Type: RSA
  Key Size: 1024
  Private Key: YES
  Ring Associations:
    *** No rings associated ***
```

---

5. Retire the original certificate and stop all use of the original private key. The original certificate and its private key exist in RACF with label 'SC75 OK RRSF SERVER CERTKR'. The new certificate and its private key exist in a separate entry in RACF with label 'SC75 OK RRSF SERVER CERT'.

6. Finalize the rollover by running the following command:

```
RACDCERT ID(RACF) ROLLOVER(LABEL('SC75 OK RRSF SERVER CERTKR')) -
  NEWLABEL('SC75 OK RRSF SERVER CERT')
```

You have now renewed a certificate that was signed by a local certificate authority by using a new private key. All information in the certificate is updated to reflect the renewal, including the key ring connection information. You have also retired and replaced the old certificate. You can continue to use the old certificate for signature verification purposes until it expires. However, you cannot use the old certificate to sign new certificates.

Run the following RACF commands:

```
RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)
RACDCERT ID(RACF) LIST
```

The output shown in Example 5-26 is displayed.

*Example 5-26 RACDCERT output*

---

```
RACDCERT ID(RACF) LISTRING(IRR.RRSF.KEYRING)
Digital ring information for user RACF:

  Ring:
    >IRR.RRSF.KEYRING<
```

Certificate Label Name	Cert Owner	USAGE	DEFAULT
-----	-----	-----	-----
RRSF CERTAUTH CERT1	CERTAUTH	CERTAUTH	NO
SC75 OK RRSF SERVER CERT	ID(RACF)	PERSONAL	YES

---

#### RACDCERT ID(RACF) LIST

Digital certificate information for user RACF:

Label: SC75 OK RRSF SERVER CERTKR  
Certificate ID: 2QTZwcPG4sP39UDW0kDZ2eLGQOLF2eXF2UDDxdnj0t1A  
Status: TRUST

...

Private Key: NO

Ring Associations:

\*\*\* No rings associated \*\*\*

Label: SC75 OK RRSF SERVER CERT

Certificate ID: 2QTZwcPG4sP39UDW0kDZ2eLGQOLF2eXF2UDDxdnj

Status: TRUST

Start Date: 2012/05/22 00:00:00

End Date: 2013/05/22 23:59:59

Ring Associations:

Ring Owner: RACF

Ring:

>IRR.RRSF.KEYRING<

---

The old certificate is still defined, but is no longer connected to the server keyring. The new certificate is now active and connected to the keyring.

## 5.5.5 Considerations for CA certificate management

A certificate authority might need to be renewed for any of the following reasons:

- ▶ Certificate authority validity periods
- ▶ Change in the policy of the certificates issued by the CA
- ▶ Expiration of the CA that issued the certificate

Every certificate issued by a CA has a validity period. This validity period is the time when the certificate can be accepted as an authoritative credential of the identity of the subject of the certificate. This period assumes that the certificate is not revoked before the validity period ends, and that the issuing CA is trusted. The primary purpose for the validity period is to limit the time span in which a certificate might be compromised.

A CA is another entity that has been issued a certificate. It is issued either by itself (root CA) or by a parent (subordinate CA). Every CA has a built-in expiration date that is determined by the end of the validity period in its CA certificate. This date does not imply that the life of a CA is equivalent to the validity period of its CA certificate. It implies that the CA cannot issue certificates if it does not have a valid certificate of its own.

## Life of a CA includes the validity periods of all its CA certificates

With these considerations in mind, an organization must plan for the renewal of every certificate that is issued by a CA in the certification hierarchy. This process maintains the existing trust relationships and extends the life of CAs.

## The validity periods of a CA and the certificates it issues

Before addressing the renewal of a CA certificate, you must understand how the end of the validity period of a CA affects the validity period of the certificates it issues. Certificate Services enforces a rule that a CA can never issue a certificate to be valid beyond the expiration date of its own certificate. Therefore, when a CA certificate reaches the end of its validity period, all certificates it has issued also expire. If the CA is not renewed and reaches the end of its lifetime, all the certificates the now-expired CA has issued can no longer be used. In other words, there are no *orphaned* certificates still within their validity period that have been issued by a CA that has expired.

Because a CA that is approaching the end of its own validity period issues certificates valid for shorter periods of time, renew the CA well before it expires. This procedure avoids issuing certificates with a short or non-existent validity period.

An RRSF server certificate can have a validity period beyond the validity period of the CA used to sign it. In this case, discard the certificate and acquire a new certificate whose validity date does not extend beyond the period of the CA used to sign it.

**Tip:** This last situation is unlikely to happen. In this case, the certificate is stored with the NOTRUST status. However, you can change the NOTRUST status to TRUST status with the RACDCERT ALTER command:

```
RACDCERT ID(RACF) ALTER(LABEL('RRSF SERVER CERTIFICATE')) -  
TRUST
```

## 5.5.6 Renewing a local CERTAUTH certificate

To renew a local CERTAUTH certificate, perform these steps:

1. Identify all the certificates used by your nodes.
2. Re-sign the CERTAUTH certificate with its own private key, specifying a new expiration date.
3. Use the re-signed CERTAUTH certificate to re-sign each of the PERSONAL certificates used by your RRSF nodes, specifying a new expiration date.
4. List the certificates to confirm the new dates.
5. Restart the RRSF connections to read in the re-signed certificates.

## 5.5.7 Renewing an expired CERTAUTH certificate

This example deals with an expired CERTAUTH certificate. When the CERTAUTH certificate expires, error messages are generated. The following examples are from the example environment.

On the node (PLEX75) where the CA HOST (SC75) is located, expect the error messages shown in Example 5-27. These error messages are for the MAIN SC75 on node PLEX75 for its connection to SC76 (remote node PLEX76) after a **RESTART COMMAND**.

*Example 5-27 RESTART CONNECTION*

---

```
#RESTART CONNECTION
...
IRRBO20I (#) IRRDDM00 TASK HAS BEEN RESTARTED. 480
IRRJ000I (#) RACF RACF LOCAL NODE TRANSACTION PROGRAM STARTING UNDER USER ID
RACF GROUP STCGROUP.
...
BPXF024I (TCP/IP) May 23 08:44:00 TTLSŸ84018028~: 04:44:00 TCP/IP 494
EZD1286I TTLS Error GRPID: 0000000F ENVID: 00000078 CONNID: 0000C990
LOCAL: 9.12.4.72..1840 REMOTE: 9.12.4.126..18136 JOBNAME: RACF
USERID: RACF RULE: Default_RRSF-Client"1 RC: 8 Initial Handshake
00000000 7EC62418
IRRI031I (#) RRSF CONNECTION TO NODE PLEX76 HAS BEEN REJECTED BECAUSE RACF
COULD NOT VERIFY AT-TLS POLICY. THE BPX1SEL SERVICE
DETECTED A SOCKET EXCEPTION.
IRRQ010I (#) RACF REMOTE SHARING TCP CONNECTOR TASK TERMINATING FOR
NODE PLEX76 WITH HOST ADDRESS 9.12.4.126.
```

---

For remote node (PLEX76) NOT on the same host as the CA host, the error messages are shown in Example 5-28.

*Example 5-28 Errors on PLEX76*

---

```
EZD1287I TTLS Error RC: 414 Initial Handshake 954
LOCAL: 9.12.4.126..18136
REMOTE: 9.12.4.72..1840
JOBNAME: RACF RULE: Default_RRSF-Server"2
USERID: RACF GRPID: 00000003 ENVID: 00000028 CONNID: 00001981
IRRI031I (#) RRSF CONNECTION FROM PEER 9.12.4.72:1840 HAS BEEN 955
REJECTED BECAUSE RACF COULD NOT VERIFY AT-TLS POLICY.
THE BPX1SEL SERVICE DETECTED A SOCKET EXCEPTION.
```

---

The connection to SC74 (non-MAIN system) from PLEX75 where the CA host is generates the errors shown in Example 5-29.

*Example 5-29 Errors on PLEX75*

---

```
EZD1287I TTLS Error RC: 8 Initial Handshake 019
LOCAL: 9.12.4.126..1062
REMOTE: 9.12.4.70..18136
JOBNAME: RACF RULE: Default_RRSF-Client"1
USERID: RACF GRPID: 00000003 ENVID: 0000002A CONNID: 0000199E
IRRI031I (#) RRSF CONNECTION TO NODE PLEX75 SYSNAME SC74 HAS BEEN 020
...
BPXF024I (OMVSKERN) May 23 08:53:16 WTSC76 TTLSŸ33554722~: 04:53:16
TCP/IP EZD1286I TTLS Error GRPID: 00000003 ENVID: 0000002A CONNID:
0000199E LOCAL: 9.12.4.126..1062 REMOTE: 9.12.4.70..18136 JOBNAME:
RACF USERID: RACF RULE: Default_RRSF-Client"1 RC: 8 Initial
Handshake 00000000 7ED2C118
```

---

SSL return codes 8 and 414 correspond are shown in Example 5-30.

*Example 5-30 SSF functions return codes*

---

#define GSK_ERR_CERT_VALIDATION	8
#define GSK_ERR_BAD_CERT	414

---

Display the certificate to see that the CERTAUTH certificate has expired (End Date: 2012/05/20 23:59:59). Similar information is displayed at the remote node when you display this CERTAUTH certificate there.

**Label1: RRSF CERTAUTH CERT2**

Certificate ID: 2QiJmZmDhZmjgdnZ4sZAw8XZ48Hk48hAw8XZ4/JA  
Status: TRUST  
Start Date: 2012/01/01 00:00:00  
**End Date: 2012/05/20 23:59:59**

Renew the CERTAUTH certificate by rekeying the internal CA self-signed certificate by using a new private key. Perform this function by using the following RACF command:

```
RACDCERT CERTAUTH REKEY(LABEL('RRSF CERTAUTH CERT2')) -  
  WITHLABEL('RRSF CERTAUTH CERT') -  
  NOTBEFORE(DATE(2012-01-01)) -  
  NOTAFTER(DATE(2020-05-20))
```

There is a new CERTAUTH **RRSF CERTAUTH CERT** that is not connected to any keyring. You are now ready to retire the original CA certificate. You must stop all use of the original private key.

The original certificate and its private key now exist in RACF under label RRSF CERTAUTH CERT2. This new certificate and its private key exist as a separate entity in RACF with label RRSF CERTAUTH CERT. Roll over the key with the following command:

```
RACDCERT CERTAUTH ROLLOVER(LABEL('RRSF CERTAUTH CERT2')) -  
  NEWLABEL('RRSF CERTAUTH CERT')
```

This command automatically connects the CERTAUTH certificate RRSF CERTAUTH CERT to the RRSF keyring. Because you rekeyed the certificate ID, its ID has changed, and the certificate serial number is also changed.

**Tip:** You can re-sign the CERTAUTH certificate with a new expiration date. Re-signing a certificate does not change the PRIVATE KEY, whereas re-keying does.

Create a certificate request from the CERTAUTH certificate listed on the previous page. The certificate is written to a new data set allocated by RACF.

```
RACDCERT CERTAUTH GENREQ(LABEL('RRSF CERTAUTH CERT')) DSN('RRSF.CERTREQ.DATASET')
```

Sign the certificate request data set with the CERTAUTH certificate itself, re-specifying the original start date, and a new end date. Make the CERTAUTH certificate good for another four years. The signed certificate request automatically replaces the original CERTAUTH certificate in RACF.

```
RACDCERT CERTAUTH GENCERT('RRSF.CERTREQ.DATASET') -  
  NOTBEFORE(DATE(2012-01-01) TIME(00:00:00)) -  
  NOTAFTER(DATE(2015-12-31) TIME(23:59:59)) -  
  SIGNWITH(CERTAUTH LABEL('RRSF CERTAUTH CERT'))
```



On the local host where your CA host is located, re-sign the RRSF local node server PERSONAL certificate with a new expiration date. For more information about creating the certificates for RRSF, see 3.4, “Creating digital certificates and key rings” on page 46.

Also, perform the steps required to EXPORT/IMPORT the newly signed Personal Certificates for the remote node systems. Those procedures are taken from z/OS V1R13.0 Security Server RACF Security Administrator's Guide SA22-7683.

## 5.6 Diagnosing RRSF problems

This section describes how to diagnose RRSF-related problems.

### 5.6.1 Error messages from components

RRSF uses a number of system components as shown in Figure 5-1.

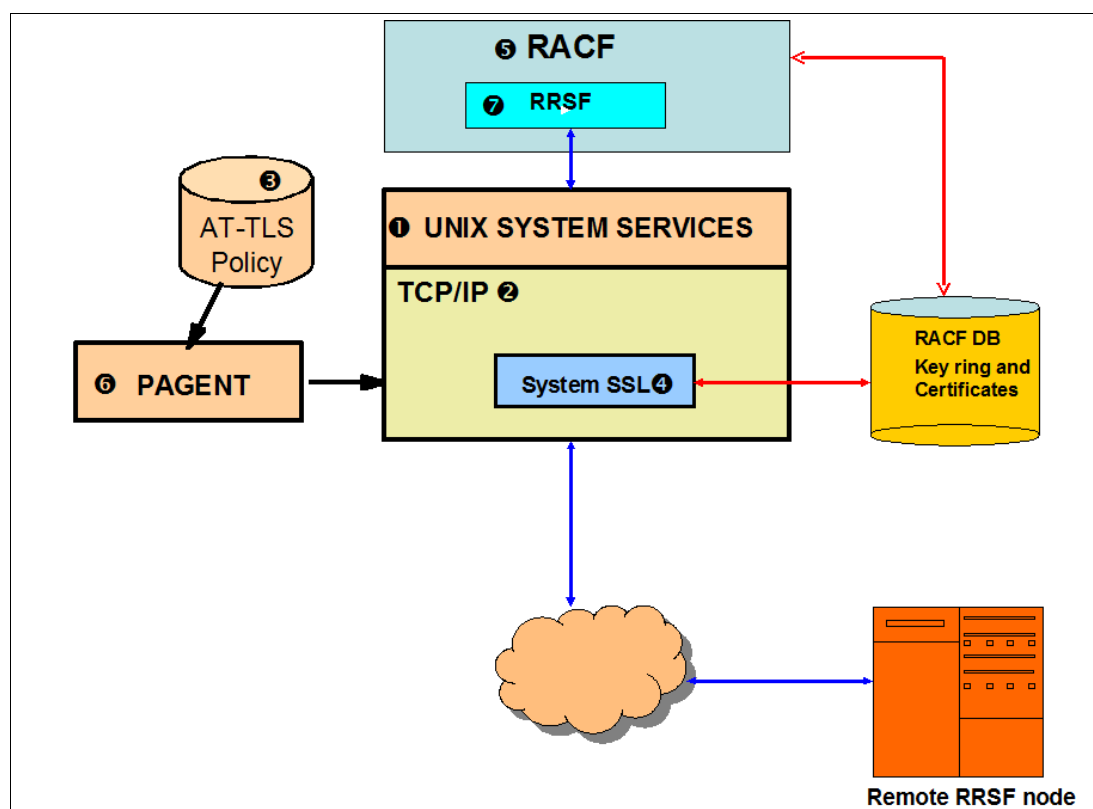


Figure 5-1 System components used by RRSF

The diagram shows these aspects:

- ▶ **1** z/OS UNIX System Services: RRSF makes UNIX System Service socket API calls to TCP/IP
- ▶ **2** TCP/IP: The communication vehicle for exchanging data between the RRSF nodes
- ▶ **3** AT-TLS: The traffic between the RRSF nodes is protected by AT-TLS
- ▶ **4** Secure Sockets Layer (SSL): AT-TLS uses SSL to authenticate the RRSF nodes
- ▶ **5** RACF: The resources used for RRSF are protected by RACF profiles

- ▶ **6** Policy Agent (PAGENT): The AT-TLS policy is enforced by the PAGENT
- ▶ **7** RRSF itself

You can get an error message from any of these components if something goes wrong. The following can go wrong from the time the system is IPLed to the point where the RRSF nodes establish a secure connection and start transmitting data:

### 1. **System IPL**

### 2. **RACF subsystem initialization starts**

RACF attempts to establish its TCP listener for RRSF. If the TCP/IP address space is not up, the connection attempt fails and the message IRRRC050I is issued. The listener periodically tries again for approximately 30 minutes, then issues message IRRRC063I and stops trying.

If the TCP/IP address space is up and running, but the PAGENT is not up yet, RACF attempts to establish its TCP listener for RRSF. This process fails again with ICH408I error INSUFFICIENT ACCESS AUTHORITY from EZB.INITSTACK.\*.\* (G) profile. The default behavior of TCP/IP is to reject such a request before the PAGENT has initialized and established the security policy. The exception is when the user ID has access to a stack initialization resource EZB.INITSTACK.\*.\* (G) profile. RACF is not given access to this profile to avoid opening any RRSF port before security is in place. This is normal, and such messages can be ignored.

If host names are used instead of IP addresses in TARGET command definitions, the resolver address space must be up before it can process a TARGET command. An external name server might have to be up as well. Otherwise the TARGET command might fail.

### 3. **TCP/IP and PAGENT are initialized**

Now the system is ready to process any TARGET commands.

### 4. **TARGET command issued**

RRSF uses z/OS UNIX System Services APIs to start socket services to establish a connection with a remote RRSF node.

If the RACF subsystem user ID does not have access to the RRSF port, it fails. This failure generates an ICH408 error INSUFFICIENT ACCESS AUTHORITY to SERVAUTH for the RACF subsystem user ID on EZB.PORTACCESS.\*.\*.RRSF.

TCP/IP then runs the TLS handshake with the assistance of System SSL. The handshake process determines the AT-TLS policy rule used to protect the connection on each system and applies that policy. The policy identifies the RACF key ring that contains the digital certificates required to authenticate each server to the other. This connection might fail for one of these reasons:

- The RACF Key ring specified in the policy rule cannot be found. Remember that the key ring name is case-sensitive.
- The RACF user ID does not have access to the key ring.
- The key ring does not contain a digital certificate for the RRSF server (the RACF subsystem user ID) as the default.
- The key ring does not have the RRSF signing certificate defined as TRUSTED.
- The RACF subsystem user ID does not have authority to read its own key rings. Reading key rings is generally accomplished by granting READ access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class. This permission is required even if the RACF subsystem started task definition specifies TRUSTED or PRIVILEGED.

- A certificate has expired.
- The key type associated with the certificate is not valid for the cipher algorithm requested in the policy.
- The certificate private key is stored in the ICSF PKDS, but ICSF was not started.
- There is a logical inconsistency between the policy statements for the “client” and “server” portions of the policy.

The TLS handshake is run on both systems, and the error might have occurred on only one of the systems. Therefore, be sure to look in the trace log on the remote system if there is no helpful information about the system you are currently logged on to.

**Consideration:** If the TCP socket listener encounters an error during initialization, or if a remote connection cannot be established, a console message is issued. The attempt is periodically tried again. RRSF tracks the failing service name, including its return and reason code,. It does not issue an additional error message if it encounters the same error on a subsequent attempt.

Whenever an error is encountered, a message is issued to the console. You need to look in z/OS UNIX System Services Messages and Codes for the explanation of the error.

**Requirement:** After correcting any key ring problems, including an authorization problem, make sure that the policy agent reads the contents of the key ring again. Change the **EnvironmentUserInstance** value in the policy rule. If you are using Configuration Assistant, click **Re-access Key Rings** under image level settings, for an image. After the policy is updated, refresh the policy agent by issuing the following command from the console:

```
F PAGENT,UPDATE
```

After correcting an ICSF problem, you must change the **GroupUserInstance** value, and then update the policy agent as shown previously. ICSF problems include starting ICSF after an RRSF connection has failed because ICSF was not available. Your system configuration must start ICSF earlier in the IPL sequence (before the policy agent starts) to avoid this problem on the next IPL.

These keywords might not be in your policy. Specifically, if using the sample policy, the **GroupUserInstance** keyword is not specified. For information about where to add this statement in your policy, see the AT-TLS policy statement section of the *z/OS Communications Server: IP Configuration Reference*, SC31-8776-21.

To diagnose a problem, you need to identify the component that is producing the error first. The next few sections address each component, including these considerations:

- How to identify an error from the component
- Where to look for an explanation of the error
- Where to look for possible causes of error

Most problems generate errors from more than one component.

## 5.6.2 Obtaining information about RRSF connections

The z/OS Communications Server NETSTAT command is frequently used to display information about TCP/IP connections. On a busy system, the output can be long. To help identify RRSF sockets, RACF associates application data with each of the sockets it uses.

Each socket is tagged with the string IRRRRSF to identify it as an RRSF socket. You can issue the NETSTAT command that specifies application data to restrict the output to RRSF connections:

```
NETSTAT ALLCON (APPLD IRRRRSF)
```

The output of this command is shown in Example 5-31.

*Example 5-31 NETSTAT ALLCON (APPLD IRRRRSF)*

---

```

NETSTAT ALLCON (APPLD IRRRRSF)
MVS TCP/IP NETSTAT CS V1R13      TCP/IP Name: TCP/IP      16:52:01
User Id Conn      Local Socket      Foreign Socket      State
-----
RACF      0000BEC9 9.12.4.72..1832      9.12.4.45..18136    EstablsH
Application Data: IRRRRSF
RACF      0000BEC8 9.12.4.72..18136      0.0.0.0..0          Listen
Application Data: IRRRRSF
RACF      0000BECB 9.12.4.72..1833      9.12.4.126..18136    EstablsH
Application Data: IRRRRSF
RACF      0000BECA 9.12.4.72..1834      9.12.4.47..18136     EstablsH
EZZ2591I APPLICATION DATA: IRRRRSF

```

---

For more information about the **NETSTAT** command, see *z/OS Communications Server: IP System Administrator's Commands*.

### 5.6.3 z/OS UNIX System Services errors

RRSF uses z/OS UNIX System Services socket calls to get TCP/IP services. Therefore you can get UNIX System Services errors if there is a problem. These errors normally are followed by a TCP/IP error.

Most of the z/OS UNIX System Services errors have a prefix of **BPX**, and provide a return code and a reason code.

#### Where to look for explanation of the error

- ▶ **Return codes**, also known as **errno**s, are generated by the system in response to either an action or lack of action. For more information about the various z/OS UNIX return codes you might receive, see *z/OS UNIX System Services Messages and Codes*. This manual describes reason codes, listing them by hexadecimal value and describing actions to correct the error.
- ▶ **Reason codes** are sometimes called **errnojrs** or as **errno2** values. The reason code is made up of 4 bytes in the following format:  

```
cccc rrrr
```

  - **cccc** is a halfword reason code qualifier. Generally this qualifier is used to identify the issuing module, and represents a module ID.
  - **rrrr** is the halfword reason code described in this documentation. Only this part of the reason code is intended as an interface for programmers.

If the contents of the two high-order bytes are within the range of X'0000' – X'20FF', look up the error in *z/OS UNIX System Services Messages and Codes*.

The two high-order bytes of the reason codes returned contain a value that is used to qualify the contents of the two low-order bytes.

**Tip:** For z/OS UNIX, zFS, TCP/IP, and IBM Language Environment® reason codes, use either the BPXMTEXT TSO/E command or the `bpxmtext` shell command to display the meaning and needed action for a reason code. For more information, see *z/OS UNIX System Services Command Reference*.

### Where to look for possible causes of error

Most problems are caused by errors in AT-TLS setup. See 5.6.5, “AT-TLS errors” on page 135.

## 5.6.4 TCP/IP errors

TCP/IP Messages start with prefix **EZZ**. The message also gives an explanation of the error.

### Where to look for explanation of the error

For more information, see the *z/OS Communications Server IP Messages* manual.

### Where to look for possible causes of error

Most problems are caused by errors in AT-TLS setup. See 5.6.5, “AT-TLS errors” on page 135.

## 5.6.5 AT-TLS errors

Most AT-TLS errors happen during the SSL handshake. You get RRSF error **IRRI031I**. AT-TLS tracing, by default, logs errors and provides an error code. However, the TLS handshake is run on both systems, and the error might have occurred on only one of the systems. Therefore, be sure to look in the trace log on the remote system if there is no helpful information about the system you are currently logged on to.

### Where to look for explanation of the error

For more information about AT-TLS errors, see *z/OS Communications Server: IP Diagnosis Guide*.

### Where to look for possible causes of error

TLS handshake errors are usually caused by certificate or key ring setup errors. This cause might not always be obvious from the error code description. The following checks generally identify the problem:

- ▶ On each system, make sure that the key ring specified in the AT-TLS policy exists and is defined for the RACF subsystem user ID.

**Remember:** The key ring names are case-sensitive

- ▶ Verify that the key ring contains a digital certificate for the RACF subsystem user ID as the default.
- ▶ Verify that the key ring contains the RRSF signing certificate and that it is trusted.
- ▶ Verify that the RACF subsystem user ID has authority to read its own key rings. You can accomplish this by granting READ access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class. This permission is required even if the RACF subsystem started task definition specifies TRUSTED or PRIVILEGED.

- ▶ Verify that a certificate has not expired.
- ▶ Verify that the key type associated with the certificate is not valid for the cipher algorithm requested in the policy.
- ▶ If the certificate private key is stored in the ICSF PKDS, verify that ICSF is started when RRSF connections were attempted.

**Attention:** Correct any key ring or certificate problems, including an authorization problem. Then make sure that the policy agent reads the contents of the key ring again by adding/changing the **EnvironmentUserInstance** value in the policy rule and refreshing PAGENT. If you are using the sample policy supplied in SYS1.PARMLIB, the GroupUserInstance keyword is not specified. You must add it in the policy.

You must go through the same process after correcting any ICSF problems

- ▶ Verify that both systems have the same policy statements for RRSF, both for the client and server portions of the policy.
- ▶ Make sure that ICSF starts earlier in the IPL sequence than the PAGENT.

**Requirement:** If you use ICSF to store your certificate keys, you must start ICSF before PAGENT so that the correct information is available for PAGENT and your policy.

## 5.6.6 Secure Sockets Layer (SSL)

The SSL errors are likely caused by an error in the certificates. You get an SSL error code as part of UNIX System Services message when there are any errors in the SSL handshake.

### Where to look for explanation of the error

*z/OS Cryptographic Services System SSL Programming* describes the SSL errors. You can also look in SYS1.SIEAHDR.H(GSKSSL) for an explanation of SSL error codes.

### Notes on SSL return codes

Browse SYS1.SIEAHDR.H(GSKSSL) to find the meanings of the SSL error and return codes as shown in Example 5-32.

*Example 5-32 GSKSSL in SYS1.SIEAHDR showing SSL return codes*

```
VIEW          SYS1.SIEAHDR.H(GSKSSL) - 01.00
Command ==>
***** Top of Data
000001 ??=if defined(__COMPILER_VER__)
000002    ??=pragma filetag("IBM-1047")
000003    ??=pragma nomargins nosequence
000004 ??=endif
...
/*****
/*  SSL function return values
*****/

#define GSK_OK                                0
#define GSK_INVALID_HANDLE                   1
#define GSK_API_NOT_AVAILABLE               2
#define GSK_INTERNAL_ERROR                   3
```

```

#define GSK_INSUFFICIENT_STORAGE          4
#define GSK_INVALID_STATE                 5
#define GSK_KEY_LABEL_NOT_FOUND           6
#define GSK_CERTIFICATE_NOT_AVAILABLE     7
...
#define GSK_CONNECTION_ACTIVE             302
#define GSK_ERR_BAD_DATE                   401
#define GSK_ERR_NO_CIPHERS                 402
#define GSK_ERR_NO_CERTIFICATE             403
#define GSK_ERR_BAD_CERTIFICATE            404
#define GSK_ERR_UNSUPPORTED_CERTIFICATE_TYPE 405
...

```

---

## 5.6.7 RACF errors

RACF errors are prefixed by **ICH** or **IRR**. The ICH messages are related to RACF commands, and the IRR messages are related to other aspects of RACF:

- ▶ DATABASE initialization(IRR8xxxy)
- ▶ RACF subsystem initialization (IRRAxxxy)
- ▶ RRSF Send (IRRFxxxy)
- ▶ RRSF PARMLIB and initialization (IRRPxxxy)
- ▶ RRSF SET (IRRHxxxy)
- ▶ RRSF Handshaking (IRRIxxxy)
- ▶ RRSF Local Transaction Connection (IRRGxxxy)
- ▶ RRSF TARGET (IRRMxxxy)
- ▶ RRSF Connection RECEIVE(IRRNxxxy)
- ▶ RRSF Connection SEND (IRROxxxy)
- ▶ RRSF General (IRRPxxxy)
- ▶ RRSF Connection task(IRRQxxxy)
- ▶ RRSF Output(IRRRxxxy)
- ▶ RACLINK/RRSF output (IRRTxxxy)
- ▶ File allocation(IRRUxxxy)
- ▶ RRSF Operational(IRRXxxxy)

### Where to look for explanation of the error

The RACF errors are described in *z/OS Security Server RACF Messages and Codes*.

### Where to look for possible causes of error

Most of the errors are caused by insufficient access to the resources used by RRSF. Verify that the various RACF profiles are defined and have the correct authorizations. Some of the things to look for are

- ▶ Verify that the RACF system ID has access to the RRSF port controlled by the profile **EZB.PORTACCESS.\*.\*.RRSF** in class **SERVAUTH**
- ▶ Verify that you defined the PAGENT user ID with an OMVS segment
- ▶ Verify that the RACF subsystem user ID has access to the profile BPX.DAEMON in the FACILITY class.

## 5.6.8 PAGENT errors

PAGENT errors have a prefix of **EZD**.

## Where to look for explanation of the error

For more information, see the *z/OS Communications Server IP Messages* manual.

## Where to look for possible causes of error

Most PAGENT errors are caused by problems in the PAGENT setup. Review the section on PAGENT setup in 3.2, “Setting up the PAGENT-started task” on page 28.

You can also dump the current PAGENT policy to verify that it has not changed. This process is important if you had other definitions active as well. See Example 5-33 for a sample job to list the current active policy in PAGENT.

The example environment uses DDNAME **STDPARM** ❹ to handle more than 100 characters. The command **pasearch -t** ❶ is issued by UID 0 ❷ as a **shell** command ❸. The output is written to DDNAME **STDOUT** ❹.

*Example 5-33 List PAGENT policy*

---

```
//ROLAND1 JOB (1234), 'ROLAND', CLASS=A, MSGCLASS=X, MSGLEVEL=(1,1),  
//          NOTIFY=&SYSUID  
//S1 EXEC PGM=BPXBATCH  
//STDOUT DD SYSOUT=* ❹  
//STDERR DD SYSOUT=*  
//STDPARM DD * ❹  
sh ❸ echo pasearch -t ❶ | su ❷  
//STDIN DD *  
//*
```

---

## 5.6.9 RRSF errors

RRSF errors are caused by errors in any of the components used by RRSF. RRSF errors have a prefix of **IRR**.

## Where to look for explanation of the error

RRSF errors are described in the *z/OS Security Server RACF Messages and Codes* manual.

## 5.6.10 Useful commands for problem diagnosis

These commands are useful for diagnosing problems:

- ▶ NETSTAT ALLCON (APPLD IRRRRSF)
- ▶ NETSTAT TTLS CONN XX
- ▶ NETSTAT TTLS CONN XX DETAIL
- ▶ TARGET LIST
- ▶ TARGET LISTP
- ▶ NETSTAT ALL
- ▶ NETSTAT ALLCONN
- ▶ NETSTAT TTLS
- ▶ pasearch -t



## 5.7 Maintenance and recovery: RRSF data sets

When planning for maintenance on a system within your RRSF network, consider not only the node that you are scheduling for maintenance, but also the effect this system has on the other nodes. Different actions must be considered depending on whether the system is part of an MSN or an SSN.

RRSF uses two main types of data sets:

- ▶ The workspace data sets are Virtual Storage Access Method (VSAM) data sets in use by the RACF subsystem for message and command flows
- ▶ The individual user RRSFLIST data sets

### 5.7.1 RRSFLIST data sets

These data sets are user data sets of the form <userid>.RRSFLIST. RRSF allocates these data sets as required, or you can pre-allocate them. These data sets are used to receive information from ONLYAT and AT directed commands as well as the OUTPUT settings established with the SET COMMAND.

Each entry in an RRSF data set has a time stamp **1** of the event, and the system and user **2** where the event took place (Example 5-34).

*Example 5-34 Sample listing from an RRSFLIST data set*

---

```
=====
SETR issued at 13:36:19 on 05/14/121 was processed at PLEX81.ROLAND2 on 05/14/12
at 14:25:24

COMMAND ISSUED: SETR      LIST

COMMAND OUTPUT:
ATTRIBUTES = INITSTATS WHEN(PROGRAM -- BASIC) SAUDIT CMDVIOL OPERAUDIT
STATISTICS = DATASET DASDVOL GDASDVOL NODES NODMBR TAPEVOL
AUDIT CLASSES = USER FACILITY FSSEC GXFACILI IDIDMAP RRSFDATA XFACILIT
ACTIVE CLASSES = DATASET USER GROUP ACCTNUM ACICSPCT AIMS APPCLU APPL
                  BCICSPCT CBIND CCICSCMD CDT CONSOLE CRYPTOZ CSFKEYS CSFSERV
                  DASDVOL DCICSDCT DIGTCERT DIGTCRIT DIGTNMAP DIGTRING
                  DSNADM DSNR ECICSDCT EJBROLE FACILITY FCICSFCT FIELD
```

---

After these data sets are filled, RRSF switches to **TSO TRANSMIT** to process command responses and command outputs. When command results are not displayed in the RRSFLIST data sets, check the allocations. Determine whether they need to be deleted and redefined, or if you need archive and refresh them. Manage these data sets as you would other log data sets by archiving or trimming old data from the beginning. Example 5-35 shows a sample from RRSFLIST.

RACF indicates that the RRSFLIST data set is full and sends the response as a **TSO TRANSMIT** **1**. A **RECEIVE** **2** command retrieves the response to display to the requestor **3**.

*Example 5-35 TRANSMIT when RRSFLIST is full*

---

```
+IRRR011I SETR was successful at node PLEX76. Output was sent via TSO TRANSMIT.
RACF 1
TSO RECEIVE 2
Dataset ** MESSAGE ** from RACF on ???????? 3
```

IRRR001I RACF command output transmitted because user data set ROLAND.RRSFLIST is full.

=====

SETR issued at 13:48:24 on 05/16/12 was processed at PLEX76.ROLAND on 05/16/12 at 13:48:24

COMMAND ISSUED: SETR LIST

COMMAND OUTPUT:

ATTRIBUTES = INITSTATS WHEN(PROGRAM -- BASIC) SAUDIT CMDVIOL OPERAUDIT  
STATISTICS = DATASET DASDVOL GDASDVOL TAPEVOL  
AUDIT CLASSES = RRSFDATA  
ACTIVE CLASSES = DATASET USER GROUP #SNOMISC ACCTNUM ACICSPCT AIMS APPCLU

---

Verify that any RRSFLIST data sets (from the OUTPUT display) are not currently being manipulated by their respective owners. Open RRSFLIST data sets can cause INMSG and OUTMSG backups.

The following sample IEBGENER job, Example 5-36, prints the contents (archives) of an RRSFLIST data set and then clears it for reuse.

*Example 5-36 Clearing an RRSFLIST data set*

---

```
//ROLAND1 JOB (1234),'ROLAND',CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),
//          NOTIFY=&SYSUID
//PRINTIT EXEC PGM=IEBGENER,COND=(0,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DISP=SHR,DSN=ROLAND.RRSFLIST
//SYSUT2 DD SYSOUT=*,
//          DCB=(RECFM=FB,LRECL=80)
//SYSIN DD DUMMY
//*
//CLEARIT EXEC PGM=IEBGENER,COND=(0,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD *
//SYSUT2 DD DISP=OLD,DSN=ROLAND.RRSFLIST
//SYSIN DD DUMMY
//*
```

---

The IEBGENER job provides relief for RRSFLIST data sets out of space and the ability to look at the contents without causing excessive RRSF backlogs.

## 5.7.2 Moving/resizing/renaming the RRSF workspace data sets

You might need to move the RRSF data sets. This move might be needed when the volumes selected for them in the initial planning for RRSF are to be decommissioned or otherwise removed from your environment.

**Restriction:** The following process should ONLY be done during a period of low RACF activity, particularly RRSF flow eligible work. The #TARGET DELETE command stops any local activity from being eligible to be shipped to any REMOTE node. The data involved will be lost.

Move the RRSF data sets by performing these steps:

1. Make the system DORMANT, then DELETE that system.

If this system is the MAIN for that NODE, perform these steps:

- a. TARGET DORMANT all other systems first (LOCAL)
- b. TARGET DORMANT the MAIN(LOCAL)
- c. TARGET DELETE the LOCAL systems
- d. TARGET DELETE the MAIN(LOCAL)

When the system in question is not the LOCAL main, perform these steps:

- a. TARGET DORMANT the system
- b. TARGET DELETE the system

2. Change the allocation statements in the TARGET statement (if required).
3. Make the system OPERATIVE as shown in Example 5-37.

*Example 5-37 Making the system operative*

---

```
TARGET NODE(PLEX81) SYSNAME(SC81) PROTOCOL(TCP(ADDRESS(9.12.4.47))) -  
  PREFIX(SYS1.RACF) WORKSPACE(VOLUME(BH8ST4)FILESIZE(1000)) MAIN  
TARGET NODE(PLEX81) SYSNAME(SC81) OPERATIVE PROTO(TCP)
```

---

4. If the INMSG and OUTMSG data sets are not empty, DEFINE a new CLUSTER by using the existing INMSG/OUTMSG data set as a model **1** (Example 5-38).
  - a. REPRO from the old CLUSTER to a temporary new CLUSTER.
  - b. Delete the old cluster.
  - c. Rename the temporary cluster to the correct name **2** as shown in Example 5-38.

*Example 5-38 Defining the cluster*

---

```
DEFINE CLUSTER(  
  NAME('SYS1.RACF.TEMP.INMSG')  
  MODEL('SYS1.RACF.SC80.SC74.INMSG') 1  
  RECORDS(1000 750))  
REPRO IDS('SYS1.RACF.SC80.SC74.INMSG') ODS('SYS1.RACF.TEMP.INMSG')  
DELETE 'SYS1.RACF.SC80.SC74.INMSG'  
ALTER 'SYS1.RACF.TEMP.INMSG' NEWNAME('SYS1.RACF.SC80.SC74.INMSG') 2  
ALTER 'SYS1.RACF.TEMP.INMSG.*' NEWNAME('SYS1.RACF.SC80.SC74.INMSG.*') 2
```

---

You can also consider the **#STOP** command to stop the RACF address space gracefully. You must quiesce the work on your LOCAL system (drained initiators, stopped most started tasks, limited TSO usage). **#STOP** might be the fastest way to terminate the RRSF work to and from this system. This process might be faster than making all LOCALS DORMANT and then DELETING all LOCALS.

To rename/relocate these data sets, update your IRROPTxx member with the new High Level Qualifier, modify the SMS constructs, or change the VOLSERS. Now DEFINE and make the LOCAL node OPERATIVE. Remember, DEFINE the LOCAL MAIN first, followed by the LOCAL non-main, then your REMOTES.

For the LOCAL INMSG and OUTMSG data sets, such as PREFIX.node.INMSG or PREFIX.node.OUTMSG, check to see whether there is any RACLINK activity. You can do so by verifying that these data sets have a zero record count **1** as shown in Example 5-39.

*Example 5-39 Verifying zero record count*

---

```
FILE USAGE
      "SYS1.RACF.SC81.INMSG"
          - CONTAINS 0 RECORD(S) 1
          - OCCUPIES 1 EXTENT(S)
      "SYS1.RACF.SC81.OUTMSG"
          - CONTAINS 0 RECORD(S) 1
          - OCCUPIES 1 EXTENT(S)
```

---

You can now **#STOP** RACF, which frees up these files. Modify the definitions as explained previously, and then restart the RACF address space (**S RACF,OPT=xx,SUB=MSTR**).

### 5.7.3 INMSG and OUTMSG data set names

When the RRSF protocol is APPC, the default names are:

```
PREFIX.LU1.LU2.INMSG
PREFIX.LU1.LU2.OUTMSG
```

Local RACLINK definitions and self targeted ONLYAT commands:

```
PREFIX.LU1.INMSG
PREFIX.LU1.OUTMSG
```

When the RRSF protocol is TCP/IP, the default names are:

- ▶ Single system for RACLINK definitions and self targeted ONLYAT commands:

```
PREFIX.sysname.INMSG
PREFIX.sysname.OUTMSG
```

- ▶ Single system to single system:

```
PREFIX.sysname.NODE2.INMSG
PREFIX.sysname.NODE2.OUTMSG
```

- ▶ Single system to multi-system:

```
PREFIX.sysname1.sysname2.INMSG
PREFIX.sysname1.sysname2.OUTMSG
```

- ▶ Multi-system to single system:

```
PREFIX.sysname1.NODE2.INMSG
PREFIX.sysname1.NODE2.OUTMSG
```

- ▶ Multi-system to multi-system:

```
PREFIX.sysname1.sysname2.INMSG
PREFIX.sysname1.sysname2.OUTMSG
```

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *RACF Version 2 Release 2 Installation and Implementation*, SG24-4580
- ▶ *RACF Version 2 Release 2 Technical Presentation Guide*, GG24-2539

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS Security Server RACF Command Language Reference*, SA22-7687
- ▶ *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- ▶ *z/OS Security Server RACF System Programmer's Guide*, SA22-7681

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)



# Index

## A

- administration of RRSF 108
  - checking the PAGENT task 111
  - RESTART command 112
  - SET command 108
    - options 109
  - TARGET command 110
  - using the NETSTAT command 111
- AT-TLS errors 135

## C

- certificate management issues 116
  - renewing certificates 128
- configure RRSF using TCP/IP 27
  - configuration steps
    - create digital certificates and key rings 46
    - define an AT-TLS policy 32
    - set up Policy Agent (PAGENT) 28
    - update RACF profiles 69
    - update TCP/IP to enable AT-TLS 68
    - verify the RRSF setup 71
  - create digital certificates and key rings 46
    - digital certificates 46
    - trust policy 46
    - using an external CA 66
    - using the same, self-signed certificate for all RRSF nodes 48
  - define an AT-TLS policy 32
    - authentication between RRSF nodes 33
    - creating a policy using the configuration assistant tool of z/OSMF 34
      - sample policy 43
      - steps and detailed walkthrough 34
    - policy rules 32
      - RRSF client traffic 33
      - RRSF server traffic 33
    - specify the cipher suite 34
  - overview of steps 28
  - set up the Policy Agent (PAGENT) 28
    - permit the policy agent access to EZB.INITSTACK 32
    - RACF requirements 30
      - authorize users to control policy agent task 30
      - define user ID with an OMVS segment 30
      - JCL example 31
      - permit to BPX.DAEMON facility if defined 30
    - restrict access to psearch 31
    - started task 29
    - steps 28
    - verify policy agent setup 32
  - update RACF profiles 69
    - add omvs segment for RACF 69
      - example JCL 69
    - enable CSFSERV resources if using cryptograph-

- ic hardware 70
  - protect the TCP/IP stack 70
- update TCP/IP to enable AT-TLS 68
  - define the RRSF port 68
  - enable AT-TLS 68
  - verify the RRSF setup 71–72
- conversion from APPC to TCP/IP 83
  - IP address 94
  - local node considerations 92
  - multisystem node considerations 92
  - order of protocol definitions 92
  - protocol conversion 84
  - protocol conversion process 85
  - RACF parameter library considerations 95
  - sample conversion walkthrough 95
  - sample environment 84
  - sample RACF parameter library members 86–87
  - simple conversion walkthrough 87
  - workspace data set considerations 93
  - z/OS R13 support for multiple protocols 89

## M

- moving RRSF workspace data sets 140

## N

- NETSTAT command example 134

## P

- PAGENT errors 137
- planning
  - configuration worksheet 15
  - deciding on RRSF functions 16
  - example configuration 14
  - main system of a multisystem node 15
    - considerations 15
  - migration from APPC to TCP/IP 23
    - earlier systems 25
    - example configuration 24
    - network considerations 26
    - RACF parameter library 25
    - workspace data set naming convention when using APPC 25
  - multiple RACF environment considerations 17
    - dynamic parse and template versions 19
  - profiles 21
    - tools available to compare RACF databases 21
  - RACF address space 17
    - example 17
    - RACF address space STARTED profile 18
  - RACF classes 21
  - RACF installation exits 19
  - RACF parameter library 19

- example 19
- RACF release level 17
- RACF subsystem 17
  - example 17
- SETOPTS options 20
  - Enhanced Generic Data Set name (EGN) settings 20
  - GENCMD(DATASET) 20
  - password options 20
- node names 14
- required reading 13
- RRSF configuration diagram 16
- TCP/IP networking 22
  - IP addresses 22
  - security 23
    - protection of TCP/IP listener port using SERV-AUTH profile 23
    - protection of TCP/IP Stack using RACF SERVAUTH profile 23
    - trust policy based on digital certificates 23
- workspace data sets 21
  - INMSG data set 21
  - naming convention in a TCP/IP configuration 22
  - OUTMSG data set 21
  - usage of data sets by RRSF 21

## R

- RACF errors 137
- RACF Remote Sharing Facility (RRSF) 1
  - administering 8
    - RRSFDATA class 9
    - SET command 8
      - functions 9
      - operator command 9
    - TARGET command 9
      - function 9
      - operator command 9
  - command process example 8
  - components 5
    - RACF subsystem address space 5
    - RRSFdata data set 5
    - TCPIP with z/OS V1R13 5
    - workspace data sets 5
      - INMSG data sets 5
      - OUTMSG data sets 5
  - functions 1, 6
    - automatic command direction 7
    - automatic password direction 6
    - command direction 6
    - password synchronization 6
  - local mode 2
  - main system of a MSN 3
  - managed user ID association 4
  - multi-system node (MSN) 3
  - network 2
  - node 2
  - peer systems in a MSN 3
  - peer user ID association 4
  - RACLINK command 4
  - remote mode 2

- Set up over TCP/IP 10
  - single system node (SSN) 3
  - support for TCP/IP 8
  - user ID association 4
- Redbooks website 143
  - Contact us x
- rekey or rollover a certificate 125
- renew a certificate
  - expired CERTAUTH certificate 128
  - expired certificate 117, 122
  - expiring CERTAUTH certificate 128
  - expiring certificate 117
- revoked IDs 115
- RRSF errors 138
- RRSFdata class 112
  - CLAUTH authority 113
  - considerations for Automatic ID Assignment 114
  - FIELD level access 114
  - RRSF resources 112
- RRSFdata data sets 139
- RRSFdata datasets
  - archiving example 140
  - content example 139
  - out of space 139

## S

- SETOPTS and Sysplex communication 114
- SSL errors 136

## T

- TCP/IP errors 135
- troubleshooting 131
  - AT-TLS errors 135
  - displaying socket connections 134
  - from IPL to RRSF activation 132
  - Policy Agent (PAGENT) errors 137
  - RACF errors 137
  - RRSF errors 138
  - SSL errors 136
  - TCP/IP errors 135
  - useful commands 138
  - z/OS Unix System Services errors 134

## Z

- z/OS Unix System Services errors 134









# RACF Remote Sharing Facility over TCP/IP



## Implementing RRSF over TCP/IP with no APPC

## Migrating from RRSF APPC to TCP/IP

## Using detailed planning and implementation scenarios

The IBM RACF remote sharing facility (RRSF) allows RACF to communicate with other IBM z/OS systems that use RACF, allowing you to maintain remote RACF databases. RRSF support for the security administrator provides these benefits:

- ▶ Administration of RACF databases from anywhere in the RRSF network
- ▶ Creation of User ID associations for password and password phrase synchronization
- ▶ Automatic synchronization of databases

Before to z/OS V1R13, RRSF only supported the APPC protocol. With z/OS release V1R13, TCP/IP can be used to extend the RACF Remote Sharing Facility (RRSF) functionality to a network of RRSF nodes capable of communicating over the TCP/IP protocol. Using TCP/IP connections for RRSF nodes provides advantages over APPC such as improved security, including stronger encryption levels.

This IBM Redbooks publication addresses the issue of implementing a new RRSF network using the TCP/IP protocol. It covers planning, implementation, and operational issues for deploying RRSF using TCP/IP. In addition, It addresses migration of an RRSF network from APPC to TCP/IP, including in-depth examples of the migration process.

## INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

## BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)