

Converting RRSF from APPC to TCP/IP

New York RACF User's Group

November 1, 2011

Laurie Ward

LWard@us.ibm.com



Trademarks

- See url <http://www.ibm.com/legal/copytrade.shtml> for a list of trademarks.



Agenda

- Review of the RACF Remote Sharing Facility (RRSF)
- Simplified illustration of creating a TCP node
- Additional setup steps
 - Making the RACF subsystem address space a UNIX process
 - Trust policy (digital certificates)
 - Application Transparent Transport Layer Security (AT-TLS)
 - Protecting the RRSF listener port
 - SERVAUTH class considerations
- Protocol conversions
- TARGET command enhancements
- Considerations for Multi-System Nodes
- References



Why we did it

- Problem:
 - Clients do not have the APPC and VTAM skills required to setup and maintain an RRSF network.
- Solution
 - RRSF will support TCP/IP (IPv4 only) as an alternate transport protocol.
- Benefit / Value
 - Clients already have the skills necessary to maintain a TCP/IP network.
 - Stronger cipher algorithms are supported to protect the data while it crosses the network.



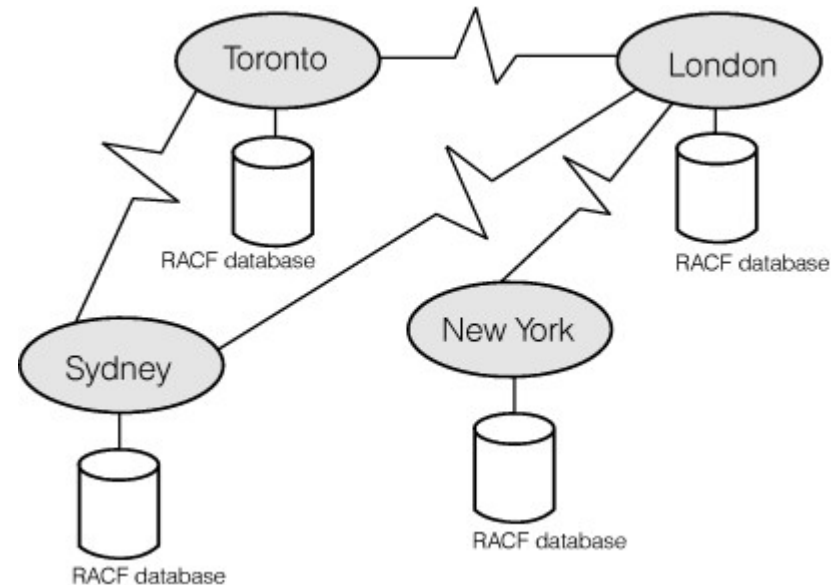
Review – What is RRSF?

- The RACF Remote Sharing Facility allows RACF to communicate with other z/OS systems that use RACF, allowing you to maintain remote RACF databases.
- Benefits of RRSF support for the security administrator include:
 - Administration from anywhere in the RRSF network.
 - User ID associations.
 - Automatic synchronization of databases.
- RRSF is designed in roughly three layers:
 - Application layer: Administrative commands and profiles
 - Presentation layer: command execution and return of command output and error and informational messages
 - Transport layer: Communication protocol used to transmit requests
 - The new function deals exclusively with the transport layer



Review – The RRSF network

- Consists of **nodes**
 - Local node: The one I'm logged on to at the moment
 - Remote nodes (all the others)
 - Local node can run in “local mode”, where there are no remote nodes
- The TARGET operator command is used to define, modify, and delete, and list nodes, as well as to de/activate them
- TARGET commands are contained within the RACF parameter library, and are executed automatically when the RACF subsystem starts
- The RACF parameter library member is specified in your started procedure JCL
- RACF parameter library members can be “chained together” using the SET INCLUDE(xx) command

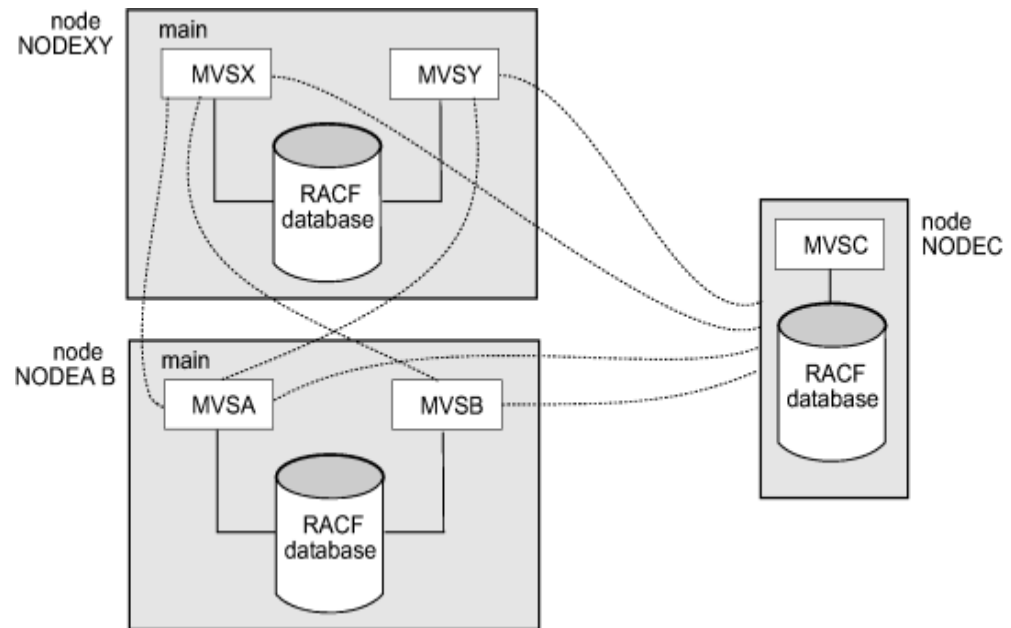


Sample RRSF network containing 4 nodes



Review – Multi-System Node (MSN)

- A set of systems sharing a RACF database (can be in a SYSPLEX, or simply on shared DASD)
- Managed with the TARGET command by specifying both NODE and SYSNAME
- All Single System Nodes (SSNs) send requests only to the MAIN system of a MSN
- All peer systems of an MSN send requests only to SSNs, and to the MAIN systems of remote MSNs
- Peer systems do not speak with each other, and do not speak with non-MAIN systems of remote MSNs



Sample RRSF network containing two Multi-System Nodes and a Single System Node




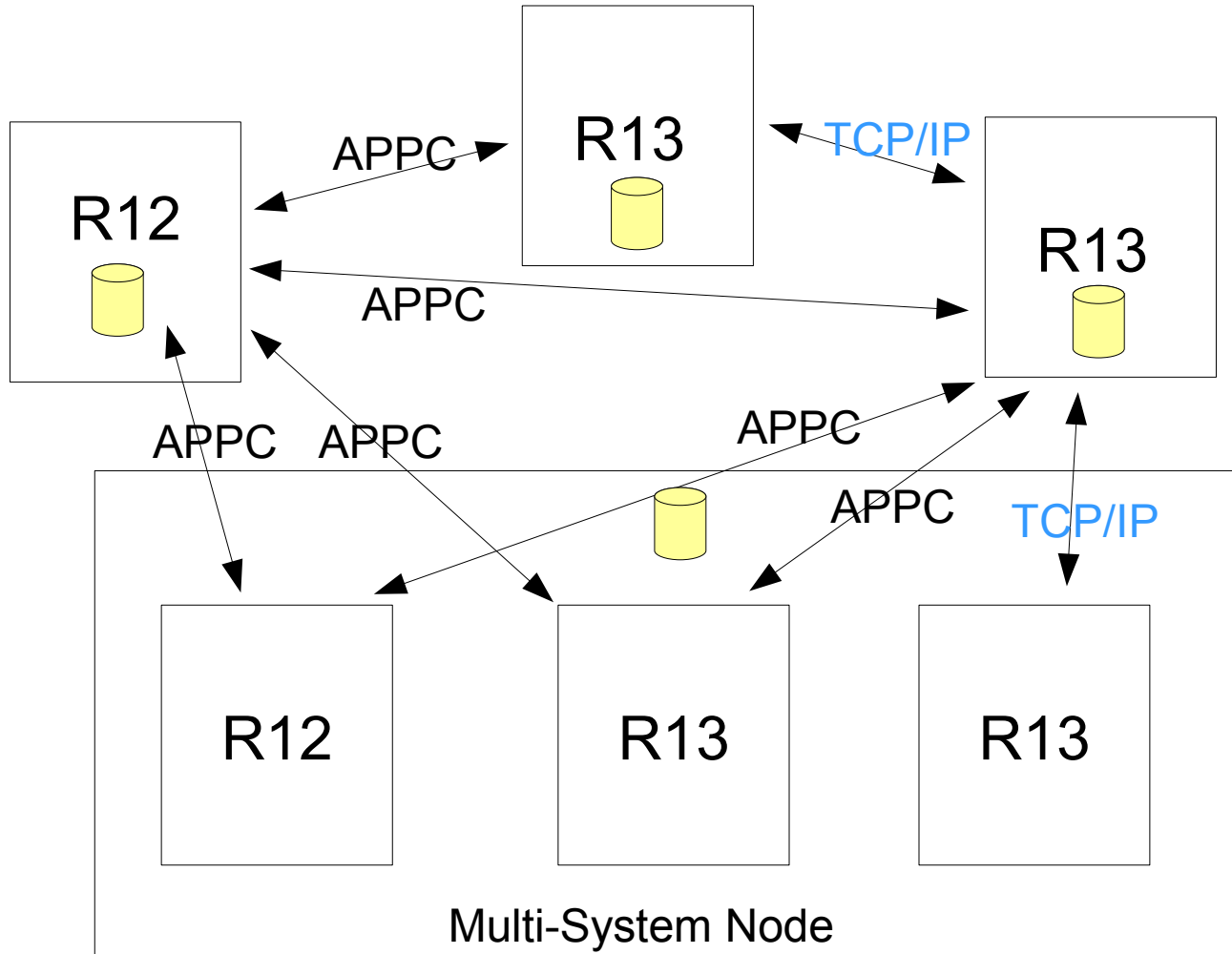
Review – Workspace data sets (i.e. checkpoint files)

- VSAM data sets that RACF uses to temporarily hold data that RACF is sending from one node to another.
- RACF deletes data from the workspace data sets when it receives confirmation that the data has been successfully processed at the receiving node.
- RACF uses two workspace data sets, the INMSG data set and the OUTMSG data set, for the local node and for each of its remote nodes.
 - The INMSG data set is used to temporarily hold requests that are being sent to the local node from itself or from a remote node (e.g. commands directed to the local node, or output from RACF commands, application updates, and password changes that were directed to a remote node)
 - The OUTMSG data set is used to temporarily hold requests that are being sent to a remote node (e.g. commands, application updates, and password changes directed from the local node, or output to be returned to a remote node)
- Requests are queued to the files while a connection is DORMANT. Queued work is sent when the connection becomes OPERATIVE ACTIVE.
- Requests are “casually encrypted” while checkpointed



Possible configurations

 = RACF parameter library



Defining a TCP/IP node and activating it using TARGET (Simplified)

- The only difference from APPC is the PROTOCOL information:
 - Define the local node with a socket listener

```
TARGET NODE (LOCAL1) LOCAL PROTOCOL (TCP)
PREFIX (SYS1.RRSF) WORKSPACE (VOLUME (VOL001)) OPERATIVE
```

```
IRRC054I (<) RACF REMOTE SHARING TCP LISTENER HAS BEEN SUCCESSFULLY
ESTABLISHED.
```

- Define the remote node and make it operative

```
TARGET NODE (REMOTE1) PROTOCOL (TCP (ADDRESS (remote.pok.ibm.com) ))
PREFIX (SYS1.RRSF) WORKSPACE (VOLUME (VOL001)) OPERATIVE
```

```
IRRI027I (<) RACF COMMUNICATION WITH TCP NODE REMOTE1 HAS BEEN
SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35
TLS_RSA_WITH_AES_256_CBC_SHA.
```

- Harden your TARGET commands in the RACF parameter library



TCP workspace naming convention

- For local node, nothing has changed:
prefix.sysname_or_wdsqual_or_nodename.INMSG|OUTMSG
SYS1.NODE1.INMSG
- For remote nodes, the current convention uses LU names as qualifiers.
prefix.local_luname.remote_luname_or_wdsqual.INMSG|OUTMSG
– This continues to be the convention for APPC nodes
SYS1.MF1AP001.MF2AP001.INMSG
- For remote TCP nodes, the new convention is
– *prefix.local-node-qualifier.wdsqual-or-nodename-or-sysname.INMSG|OUTMSG*
SYS1.NODE1.NODE2.INMSG
- This makes protocol conversions interesting. More later...



There's more to the TCP/IP setup than just the TARGET command

- First you must:
 - Add an OMVS segment with UID to the RACF subsystem user ID, and an OMVS segment with GID to its default group
 - Deploy digital certificates/key rings which are used to authenticate RRSF servers to each other using the TLS (Transport Layer Security) protocol
 - Enable the AT-TLS (Application Transparent Transport Layer Security) policy required for RRSF connections (samples provided). The policy identifies the name of the key ring to use (and much more)
 - Protect the RRSF listener port
 - Permit the RACF subsystem identity to the necessary resources
 - Even if it's TRUSTED! (more later)



Setup: OMVS segment for the RACF subsystem ID profile

- 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 using the ALTUSER command.

- Assign an OMVS segment with a GID to its default group using the ALTGROUP command.

- If you don't, you will see an error message when the socket listener attempts to start.
 - Just assign the OMVS segments, and make the local node OPERATIVE again
 - You do **not** have to restart the RACF subsystem!



Setup: Digital certificates

- Network entities authenticate to each other via the trust policy established by digital certificates
 - “I will believe you are who you say you are if someone I trust is vouching for your identity.”
 - “I have a list of the people I trust”
- Identities of the people (or servers) with whom I talk, and the people who I trust, are represented by digital certificates
- For a given application, I keep the server certificate, and those of the people I trust, in a container called a key ring
- The TLS standard requires the server to send its certificate to the client for validation
 - And optionally requires the client to send its certificate to the server for validation (a.k.a. “client authentication”).



Setup: Digital certificates ...

- Question: In an RRSF network, who is the client and who is the server?
- Answer: 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.
 - That is, this is not the traditional client/server model; Rather, it is a mesh of peers.
- So, we must enforce client authentication so that both sides of the conversation are authenticated to each other



Setup: Digital certificates ...

- Question: Because I trust the person vouching for your identity, does that make you an RRSF instance?
- Answer: No! Not necessarily...

- Question: If I knew that the sole purpose in life of somebody I trust is to sign RRSF server certificates, and I am presented with a server certificate signed by that person, then is that server an RRSF instance?
- Answer: Yes.

- Question: Whom do I need to trust to make sure that this is the case?
- Answer: You! (or your security/network administrator)



Setup: Digital certificates ... Bottom line

- On each system, the RACF address space must have access to a key ring containing
 - 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 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, 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!**
- See the Security Administrator's Guide for details.



Setup: Digital certificates ... Addendum

- Question: But what if my security policy forces me to obtain digital certificates from an external certificate authority?
- Answer: Patiently explain to the Powers That Be that you are configuring a RACF-to-RACF function between servers, and that the RACDCERT command has already been paid for.
- If this doesn't work, you can still set up RRSF securely, but it's going to take some more work:
 - Change your AT-TLS policy to specify a client authentication level of SAFCheck (more on AT-TLS policy shortly)
 - “Map” every server certificate to a RACF user ID on every other system (there are multiple ways of accomplishing this)
 - Grant the mapped user ID READ access to IRR.RRSF.CONNECT in the RRSFDATA class on each system



Setup: AT-TLS policy

- Question: Now that you have your certificates in place, how will the system know to use them?
- Answer: Your AT-TLS policy contains the name of the key ring.

- TCP/IP, using System SSL, will use the policy to perform the TLS handshake when one RRSF attempts to connect to another.
- Working policy samples are provided. Policy just needs to be enabled and installed into the Policy Agent.
 - Communication Server Configuration Assistant GUI provides policy
 - RACF also ships raw policy statements in
SYS1.SAMPLIB(IRRSRRSF)
- RRSF is a “TLS-aware application”. RRSF will refuse to connect or accept connections unless adequate policy is in effect.



Setup: AT-TLS policy ... Sample

AT-TLS Perspective

Navigation tree

- AT-TLS
 - Reusable Objects
 - Traffic Descriptors
 - Security Levels
 - Address Groups
 - Requirement Maps
 - z/OS Images
 - Image - POKVMTL4
 - Stack - TCPIP

TCP/IP stack name: * TCPIP

Description: My TCP/IP stack

z/OS release: V1R13

Enable the rule you would like to have in your AT-TLS policy.
To enable a rule, right click on the row and select Enable Rule.

Status	Rule Name	Application / Requirement Map	Key Ring
Disabled	Default_DB2-Requester	DB2-Requester	tsKeyring
Disabled	Default_DB2-Server	DB2-Server	tsKeyring
Disabled	Default_Central_PolicySvr	Centralized_Policy_Server	tsKeyring
Disabled	Default_CICS	CICS	tsKeyring
Disabled	Default_CSSMTP	CSSMTP	tsKeyring
Disabled	Default_FTP-Client	FTP-Client	tsKeyring
Disabled	Default_FTP-Server	FTP-Server	tsKeyring
Disabled	Default_IMS-Connect	IMS-Connect	tsKeyring
Enabled	Default_JES-Client	JES-Client	tsKeyring
Disabled	Default_JES-Server	JES-Server	tsKeyring
Disabled	Default_LBA-Advisor	LBA-Advisor	tsKeyring
Disabled	Default_MSM	MSM	tsKeyring
Disabled	Default_NETCONV	NETCONV	tsKeyring
Disabled	Default_NSS_Client-IKED	NSS_Client-IKED	tsKeyring
Disabled	Default_NSS_Server	NSS_Server	tsKeyring
Disabled	Default_PolicyAgentImport	PolicyAgentImport	tsKeyring
Disabled	Default_RRSF-Client	RRSF-Client	tsKeyring
Disabled	Default_RRSF-Server	RRSF-Server	tsKeyring
Disabled	Default_TN3270-Server	TN3270-Server	tsKeyring

Modify... Copy... Add... Delete Move Up View Details Health Check

Move Down

Main Perspective Apply Changes OK Cancel Help ?

Screen shot from the Communication Server Configuration Assistant GUI



Setup: AT-TLS policy ...

- The sample will satisfy RRSF, but in case you modify it, RRSF will enforce the following properties:
 - Policy is in effect for the connection
 - The TCP/IP stack is enabled for policy (TCPCONFIG TTLS)
 - A matching policy rule was found (match is based on target port number)
 - The rule is enabled
 - Policy specifies a client authentication level of at least “Required”
 - SSL V3, or TLS, is specified as the protocol level
 - Application-controlled attribute is OFF

- Note: A minimum encryption level is **not** enforced, though the sample specifies the strongest level currently available (AES-256), and the value is reported in the connection message and is displayed in TARGET LIST output.



Setup: Protect the RRSF Listener Port

- RRSF Listener Port = 18136

- Reserved with Internet Assigned Numbers Authority (IANA)

- Assign a SAF name in the TCP/IP profile for the listener port for each node

```
PORT 18136 TCP * SAF RRSF ; Listener port for RACF Remote Sharing
```

- Protect it with a SERVAUTH profile

- Resource name

```
EZB.PORTACCESS.sysname.stack-name.SAF-name
```

- Examples

```
RDEFINE SERVAUTH EZB.PORTACCESS.NODE1.TCPIP.RRSF
```

Or

```
RDEFINE SERVAUTH EZB.PORTACCESS.*.*.RRSF
```

- Give the RACF subsystem access to it

```
PERMIT EZB.PORTACCESS.NODE1.TCPIP.RRSF CLASS (SERVAUTH)
ID(RACFSUB) ACCESS (READ)
```



Setup: Resource permissions for the RACF address space user ID

- Generally, you run the RACF started task with the TRUSTED attribute, and automatic access is granted to RACF-protected resources
 - However, this does not play well with the SERVAUTH class
- So, the RRSF tasks that perform TCP/IP communication run under a task-level security environment (ACEE) without the TRUSTED or PRIVILEGED attributes
- As a result, the subsystem user will need to be permitted to whatever SERVAUTH class profiles are protecting resources it is accessing
 - Example

```
PERMIT EZB.STACKACCESS.NODE1.TCPIP CLASS (SERVAUTH) ID (RACFSUB)
ACCESS (READ)
```
 - But do **not** permit it to the stack initialization resource or remote connections may fail during IPL
 - Ignore ICH408I message for `EZB.INITSTACK.sysname.tcpname`



Protocol conversions – the pitfalls

- Because of the different workspace file naming conventions, TARGET commands for one protocol cannot derive the names used by the other, and there is no persistent memory across restart/IPL
- So when defining the new protocol for a given node, a new set of files will be allocated
- The following problems must be avoided:
 - We cannot lose whatever work may be queued in the old files
 - We cannot let requests run out of order
 - We cannot impose a “quiet time” on the customer to allow the old files to drain before queuing work to the new files.
 - If there **is** a disruption (subsystem restart), we must continue where we left off when the subsystem resumes
 - The conversion process should work in either direction

Guideline: Perform conversion when workspace data sets are close to empty and not receiving many updates



Protocol conversions – some new terminology

- Protocol instance – a set of protocol information for a particular transport mechanism
 - Local node – an instance is an attribute of the single logical representation of the local node
 - Remote node – an instance is a separate logical representation of the connection to a remote node. It contains its own workspace files, prefix, description, etc.
- Multi-Protocol node – A node which has more than one protocol instance. A remote node is multi-protocol throughout the conversion process. The local node is multi-protocol for as long as you have a mixed-protocol network.
- Protocol specification – the act of specifying protocol information using the TARGET command
- Protocol qualification – the act of identifying the protocol instance to modify using the TARGET command



Protocol conversions – what happens during the conversion

- Communication will continue uninterrupted using the old protocol until the new protocol instance establishes a connection
- The new protocol instance will assume ownership of the old protocol instance's files
- The old protocol instance will be automatically deleted
- New requests will be queued to the new instance's files while the old instance's files are draining
- When the old instance's files have drained, they will automatically be deallocated and deleted
- It will now appear as though the new protocol instance is the only one that ever existed



Protocol conversions – How to do it

- Before you begin:
 - Read procedure in RACF System Programmer's Guide
 - Ensure you have done all setup for RRSF to use TCP/IP
 - Ensure you have sufficient DASD space for 2 sets of workspace data sets on each node being converted
- 1) On local node, create RACF parameter member (e.g. IRROPTRR) which will contain TARGET command for remote node using TCP/IP
 - Just put TARGET LIST command in it for now
 - At end of current parmlib member, add SET INCLUDE(RR) for new member
- 2) Specify protocol information for the local node if you haven't already

```
TARGET NODE (LOCAL1) DORMANT
TARGET NODE (LOCAL1) LOCAL PROTOCOL (TCP)
PREFIX (SYS1.RRSF) WORKSPACE (VOLUME (VOL001)) OPERATIVE
```

```
IRRC054I (<) RACF REMOTE SHARING TCP LISTENER HAS BEEN SUCCESSFULLY
ESTABLISHED.
```



Protocol conversions – How To do it...

3) On local node, update existing RRSF parmlib member

- Add TARGET LOCAL command for local node with TCP keyword (from Step 2) directly after existing TARGET LOCAL command for local node.
- Remove OPERATIVE keyword from TARGET LOCAL command with APPC

4) Repeat steps 1 to 3 on remote system

5) On both systems, copy existing TARGET command for remote node into new parmlib member IRROPTRR and edit as follows:

- Replace PROTOCOL(APPC(...)) with PROTOCOL(TCP(...))
- Adjust workspace data set names if you wish. Keep same PREFIX. Make sure new data sets will be protected by RACF profiles.

Note: The TARGET command should look as though you are defining the node from scratch. Nothing will be copied from the existing protocol.



Protocol conversions – How To do it...

6) On first system, issue new TARGET command from console and debug any problems.

- If you make changes to the TARGET command, remember to change the copy of that command in IRROPTRR
- Expect the first TARGET command to fail

On NODE1:

```
<target node(node2) prefix(sys1.rrsf) workspace(volume(temp01))  
  protocol(tcp(address(alps4260.pok.ibm.com))) operative
```

```
IRRC057I (<) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE  
  NODE2 HAS BEEN INITIATED.
```

```
IRRM002I (<) RSWK SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
```

```
IRRI016I (<) ERROR: LOCAL NODE NODE1 AND PARTNER NODE NODE2 HAVE CONFLICTING  
  TARGET STATEMENTS WITH LOCAL SYSTEM. REASON CODE 5.
```



In the midst of a conversion, TARGET LIST shows both protocols by default

?target node(node1) list

– IRRM010I (?) RSFX SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE1 **PROTOCOL APPC:**

```

- STATE      - OPERATIVE ACTIVE
- DESCRIPTION - <NOT SPECIFIED>
- PROTOCOL  - APPC
-   LU NAME   - NODE1LU
-   TP PROFILE NAME - IRRRACF
-   MODENAME  - <NOT SPECIFIED>
-   TIME OF LAST TRANSMISSION TO - <NONE>
-   TIME OF LAST TRANSMISSION FROM - <NONE>
-   WORKSPACE FILE SPECIFICATION
-     PREFIX   - "RRSF1"
-     WDSQUAL  - <NOT SPECIFIED>
-     FILESIZE - 500
-     VOLUME   - TEMP01
-   FILE USAGE
-     "RRSF1.MF2AP001.NODE1LU.INMSG"
-       - CONTAINS 0 RECORD(S)
-       - OCCUPIES 1 EXTENT(S)
-     "RRSF1.MF2AP001.NODE1LU.OUTMSG"
-       - CONTAINS 0 RECORD(S)
-       - OCCUPIES 1 EXTENT(S)

```

– IRRM010I (?) RSFX SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE1 **PROTOCOL TCP:**

```

- STATE      - OPERATIVE PENDING CONNECTION
- DESCRIPTION - <NOT SPECIFIED>
- PROTOCOL  - TCP
-   HOST ADDRESS - ALPS4167.POK.IBM.COM
-   LISTENER PORT - 18136
-   TIME OF LAST TRANSMISSION TO - <NONE>
-   TIME OF LAST TRANSMISSION FROM - <NONE>
-   WORKSPACE FILE SPECIFICATION
-     PREFIX   - "RRSF1"
-     WDSQUAL  - <NOT SPECIFIED>
-     FILESIZE - 500
-     VOLUME   - TEMP01
-   FILE USAGE
-     "RRSF1.NODE1.NODE2.INMSG"
-       - CONTAINS 0 RECORD(S)
-       - OCCUPIES 1 EXTENT(S)
-     "RRSF1.NODE1.NODE2.OUTMSG"
-       - CONTAINS 0 RECORD(S)
-       - OCCUPIES 1 EXTENT(S)

```



Protocol qualification

- When 2 protocols are defined for a given remote node, the `protocol` keyword must be specified on subsequent `TARGET` commands to further **qualify** which instance of the node is being manipulated.

```
— TARGET node(node1) protocol(TCP) description('remote node node1')
```

- If the protocol is not qualified, an error results

```
— target node(node1) description('remote node node1')  
— IRRM087I (?) RSFX SUBSYSTEM TARGET COMMAND REQUIRES THAT A  
  PROTOCOL BE SPECIFIED FOR NODE NODE1 TO IDENTIFY THE INTENDED  
  PROTOCOL INSTANCE.
```



Protocol conversions – draining the old files

- After the old instance is deleted, but before the conversion completes, TARGET LIST will show that the node owns two sets of workspace files

```
<target list node(node2)
IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2:
STATE          - OPERATIVE ACTIVE
...
WORKSPACE FILE SPECIFICATION
PREFIX          - "SYS1.RRSF"
WDSQUAL        - <NOT SPECIFIED>
FILESIZE       - 500
VOLUME        - TEMP01
FILE USAGE
    "SYS1.RRSF.NODE1.NODE2.INMSG"
        - CONTAINS 0 RECORD(S)
        - OCCUPIES 1 EXTENT(S)
    "SYS1.RRSF.NODE1.NODE2.OUTMSG"
        - CONTAINS 0 RECORD(S)
        - OCCUPIES 1 EXTENT(S)

CONVERSION FILE
    INMSG WORKSPACE FILE NOT ALLOCATED
    "SYS1.RRSF.MF1AP001.MF2AP001.OUTMSG"
        - CONTAINS 5 RECORD(S)
        - OCCUPIES 1 EXTENT(S)
```



Protocol conversions – How To do it...

- (still part of Step 6) Issue new TARGET command on second system
 - Protocol mismatch will be resolved and conversion process should start

On NODE2:

```
>target node(node1) prefix(sys1.rrsf) workspace(volume(temp01))  
  protocol(tcp(address(alps4242.pok.ibm.com))) operative
```

```
IRRC057I (>) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE  
  NODE1 HAS BEEN INITIATED.
```

```
IRRM002I (>) RSWL SUBSYSTEM TARGET COMMAND HAS COMPLETED SUCCESSFULLY.
```

```
IRRI027I (>) RACF COMMUNICATION WITH TCP NODE NODE1 HAS BEEN  
  SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35  
  TLS_RSA_WITH_AES_256_CBC_SHA.
```

```
IRRC058I (>) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE  
  NODE1 IS COMPLETE.
```

On NODE1:

```
IRRI027I (<) RACF COMMUNICATION WITH TCP NODE NODE2 HAS BEEN  
  SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM 35  
  TLS_RSA_WITH_AES_256_CBC_SHA.
```

```
IRRC058I (<) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE  
  NODE2 IS COMPLETE.
```



Protocol conversions – How To do it...

- After the conversion completes, there is no trace left of the APPC instance:

```
<target list node(node2) protocol (appc)
```

```
IRRM005I (<) RSWJ SUBSYSTEM TARGET COMMAND WAS UNABLE TO FIND  
DEFINITION OF NODE NODE2 PROTOCOL APPC.
```

```
IRRM003I (<) RSWJ SUBSYSTEM TARGET COMMAND ENDED IN ERROR.
```

```
<target list node(node2)
```

```
IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2:  
STATE          - OPERATIVE ACTIVE
```

```
...
```

```
...
```

FILE USAGE

```
"SYS1.RRSF.NODE1.NODE2.INMSG"
```

```
- CONTAINS 0 RECORD(S)
```

```
- OCCUPIES 1 EXTENT(S)
```

```
"SYS1.RRSF.NODE1.NODE2.OUTMSG"
```

```
- CONTAINS 0 RECORD(S)
```

```
- OCCUPIES 1 EXTENT(S)
```



Protocol conversions – How To do it...

- 7) Wait for conversion to complete successfully on BOTH systems.
 - Delete TARGET command for APPC for the remote node from original parmlib members
 - If you leave the APPC statements in the parmlib members, a protocol conversion will occur during each IPL (ok, but 'noisy')
- To change protocol for connections to other remote nodes, repeat steps 5 to 7 for each connection
- If you change the protocol for all remote connections, you can remove TARGET LOCAL command with the old protocol information from the old parmlib member. Then you can optionally combine the two parmlib members and delete the SET INCLUDE(RR) command from the original member.



Syntax of the TARGET command

New



New



```

subsystem-prefixTARGET
  [ DELETE | DORMANT | OPERATIVE ]
  [ DESCRIPTION('description') ]
  [ LIST ]
  [ LISTPROTOCOL ]
  [ LOCAL ]
  [ MAIN ]
  [ NODE(nodename |*) ]
  [ PREFIX(qualifier ...) ]
  [ PROTOCOL(
    [ APPC(
      [ LUNAME(luname) ]
      [ TPNAME(profile-name) ]
      [ MODENAME(mode-name) ]
    ) ]
    [ TCP(
      [ ADDRESS(host-name) ]
      [ PORTNUM(number) ]
    ) ]
  ) ]
  [ PURGE(INMSG | OUTMSG) ]
  [ SYSNAME(sysname |*) ]
  [ WDSQUAL(qualifier) ]
  [ WORKSPACE( {
    [ STORCLAS(class-name) ]
    [ DATACLAS(class-name) ]
    [ MGMTCLAS(class-name) ]
    | [ VOLUME(volume-serial) ] }
  [ FILESIZE([ nnnnnnnnnnn | 500 ]
  ) ]
  ) ]
  ) ]
  ) ]

```

TARGET command syntax: The LISTPROTOCOL and TCP keywords are new

TARGET LIST: summary version

- A new message line, prefixed with IRRM091I, indicates the status of each protocol listener defined to the local node.

```
- NODE1 <target list
- NODE1 IRRM009I (<) LOCAL RRSF NODE NODE1 IS IN THE OPERATIVE ACTIVE
- STATE.
- IRRM091I (<) - LOCAL NODE APPC LISTENER IS ACTIVE.
- IRRM091I (<) - LOCAL NODE TCP LISTENER IS ACTIVE.
- IRRM009I (<) REMOTE RRSF NODE NODE2 IS IN THE OPERATIVE ACTIVE STATE.
```

- Status values are ACTIVE, INACTIVE, and INITIALIZING



TARGET LISTPROTOCOL

- LISTPROTOCOL is a new keyword that displays the protocol in IRRM009I for remote nodes

```
- NODE1 <target listprotocol
- NODE1 IRRM009I (<) LOCAL RRSF NODE NODE1 IS IN THE OPERATIVE ACTIVE STATE.
- IRRM091I (<)          - LOCAL NODE APPC LISTENER IS ACTIVE.
- IRRM091I (<)          - LOCAL NODE TCP LISTENER IS ACTIVE.
- IRRM009I (<) REMOTE RRSF NODE NODE2 PROTOCOL TCP IS IN THE OPERATIVE ACTIVE STATE
- IRRM009I (<) REMOTE RRSF NODE NODE3 PROTOCOL TCP IS IN THE OPERATIVE ACTIVE STATE
- IRRM009I (<) REMOTE RRSF NODE NODE4 PROTOCOL APPC IS IN THE OPERATIVE ACTIVE STATE
- IRRM009I (<) REMOTE RRSF NODE NODE5 PROTOCOL TCP IS IN THE OPERATIVE ACTIVE STATE
- IRRM009I (<) REMOTE RRSF NODE NODE6 PROTOCOL APPC IS IN THE OPERATIVE ACTIVE STATE
```

- Comes in handy when displaying a mixed-protocol network



TARGET LIST: detailed version

- For the local node, shows protocol information for all defined protocols

```

NODE1 <target list node(node1)
NODE1 IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF LOCAL RRSF NODE NODE1:
  STATE          - OPERATIVE ACTIVE
  DESCRIPTION    - <NOT SPECIFIED>
  PROTOCOL      - APPC
                  LU NAME          - MF1AP001
                  TP PROFILE NAME   - IRRRACF
                  MODENAME         - <NOT SPECIFIED>
                  LISTENER STATUS   - ACTIVE
  PROTOCOL      - TCP
                  HOST ADDRESS      - 0.0.0.0
                  IP ADDRESS        - 9.57.1.243
                  LISTENER PORT     - 18136
                  LISTENER STATUS   - ACTIVE
  TIME OF LAST TRANSMISSION TO     - <NONE>
  TIME OF LAST TRANSMISSION FROM   - <NONE>
  WORKSPACE FILE SPECIFICATION
    PREFIX          - "NODE1.WORK"
    WDSQUAL        - <NOT SPECIFIED>
    FILESIZE       - 500
    VOLUME         - TEMP01
    FILE USAGE
      "NODE1.WORK.NODE1.INMSG"
        - CONTAINS 0 RECORD(S)
        - OCCUPIES 1 EXTENT(S)
      "NODE1.WORK.NODE1.OUTMSG"
        - CONTAINS 0 RECORD(S)
        - OCCUPIES 1 EXTENT(S)

```

TARGET LIST: detailed version

- For a **connected** remote node, shows some AT-TLS information
 - Much more AT-TLS info is available with the NETSTAT command

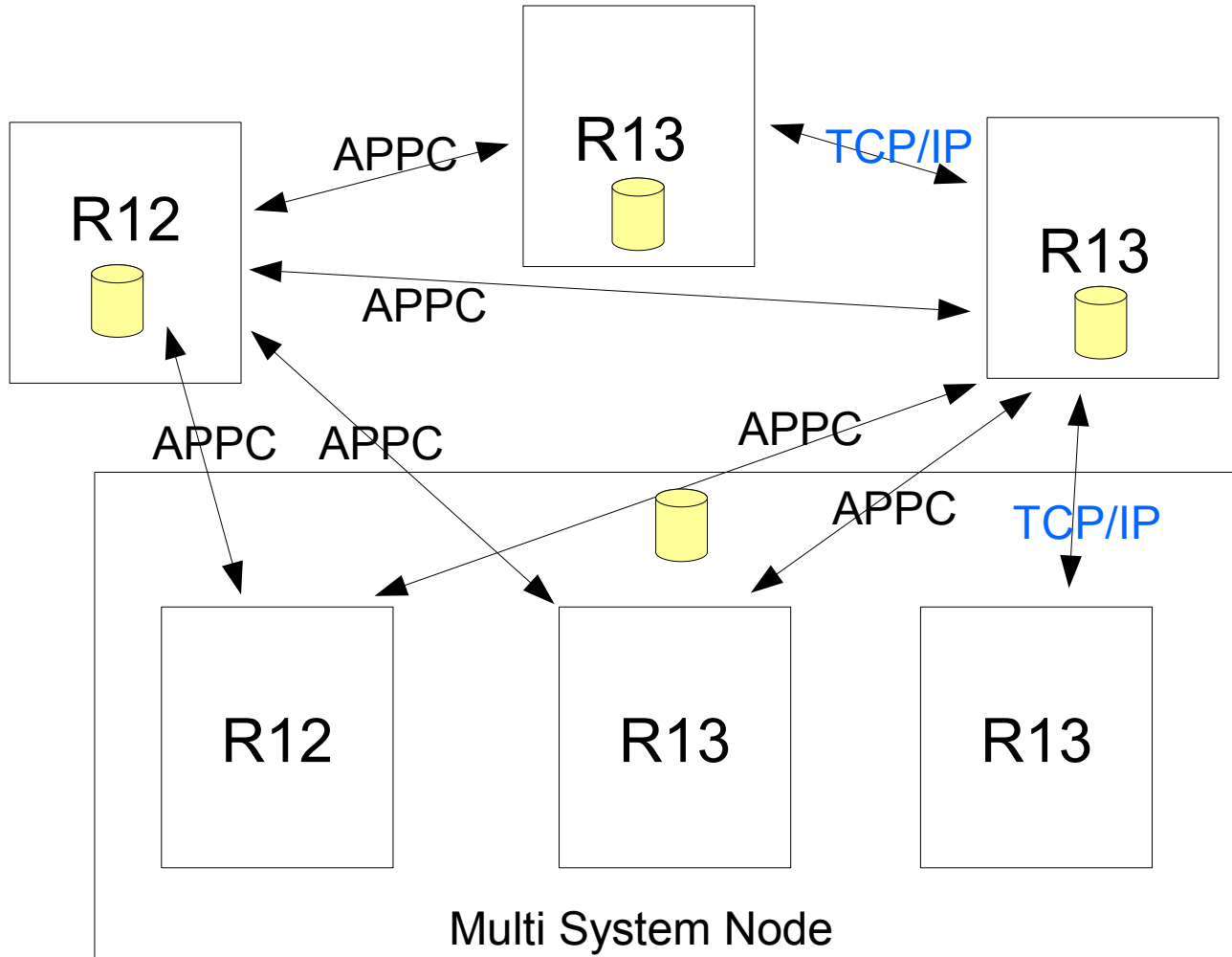
```

NODE1 <target list node(node2)
NODE1 IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2:
STATE          - OPERATIVE ACTIVE
DESCRIPTION    - <NOT SPECIFIED>
PROTOCOL       - TCP
                HOST ADDRESS      - ALPS4012.POK.IBM.COM
                IP ADDRESS        - 9.57.1.13
                LISTENER PORT     - 18136
AT-TLS POLICY:
                RULE_NAME        - RRSF-CLIENT
                CIPHER ALG     - 35 TLS_RSA_WITH_AES_256_CBC_SHA
                CLIENT AUTH    - REQUIRED
TIME OF LAST TRANSMISSION TO - <NONE>
TIME OF LAST TRANSMISSION FROM - <NONE>
WORKSPACE FILE SPECIFICATION
  PREFIX          - "RSFJ.BRUCE"
  WDSQUAL         - <NOT SPECIFIED>
  FILESIZE        - 500
  VOLUME          - TEMP01
  FILE USAGE
    "RSFJ.BRUCE.NODE1.NODE2.INMSG"
    - CONTAINS 0 RECORD(S)
    - OCCUPIES 1 EXTENT(S)
    "RSFJ.BRUCE.NODE1.NODE2.OUTMSG"
    - CONTAINS 0 RECORD(S)
    - OCCUPIES 1 EXTENT(S)

```


Possible configurations – Multi-System Nodes

= RACF parameter library



- R12 systems can only speak APPC
- R13 systems can only speak APPC to R12 systems
- R13 systems can speak TCP/IP or APPC to other R13 systems



Multi-system Node Considerations

- Multi-system Nodes (MSNs) can consist of mixed levels where communication is constrained
- MSNs can continue to use a shared RACF parameter library
 - Two TARGET commands will be required for each remote system. The TCP/IP command must follow the APPC command (assuming that's your preferred protocol)
 - The TCP/IP command will harmlessly fail when executed on R12
 - The TCP/IP command will trigger a conversion when executed on R13
- Conversion procedure in RACF System Programmer's Guide has MSN considerations for several steps



Session Summary

- Review of the RACF Remote Sharing Facility (RRSF)
- Simplified illustration of creating a TCP node
- Additional setup steps
 - Making the RACF subsystem address space a UNIX process
 - Trust policy (digital certificates)
 - Application Transparent Transport Layer Security (AT-TLS)
 - Protecting the RRSF listener port
 - SERVAUTH class considerations
- Protocol conversions
- TARGET command enhancements
- Considerations for Multi-System Nodes
- References



Appendix

- RACF: System Programmer's Guide (SA22-7681-13)
- RACF: Command Language Reference (SA22-7687-16)
- RACF: Security Administrator's Guide (SA22-7683-15)
- RACF: Diagnosis Guide (GA22-7689-14)
- UNIX System Services: Messages and Codes (SA22-7807-12)
- UNIX System Services Programming: Assembler Callable Services Reference (SA22-7803-14)
- Communication Server: IP Configuration Guide (SC31-8775-18)
- Communication Server: IP Diagnosis Guide (GC31-8782-12)
- Communication Server: IP System Administrator's Commands (SC31-8781-11)



Appendix ...

- Communication Server web site
 - <http://www-01.ibm.com/software/network/commserver/zos/>
- Red book: IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking (SG24-7801-00)
- Comm Server Configuration Assistant download from IBM support
 - <http://www-01.ibm.com/support/docview.wss?uid=swg24013160>
- AT-TLS education assistant
 - http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/topic/com.ibm.iea.commserv_v1/commserv/1.7z/security/AT_TLS.pdf
- Search SHARE web site for AT-TLS presentations



Backup slides

- It's way too early to be thinking about diagnostics, but here is some additional information for when you get around to implementing this function.
- This is all covered in the RACF Diagnosis Guide



New status messages for TCP communication

- Listener status (successful initialization and termination, and error message when failing to start) are issued to the console
- Connection status (successful or failed connection, and successful or failed termination) are issued to the console
- On failure, attempts are periodically retried (except for hand shaking errors), but duplicate error messages will not be issued
 - If unsuccessful after about 30 minutes, a console message is issued and no more retries are attempted
- Subtask start and end messages are issued to SYSLOG only
- There will not be one-to-one correspondence with messages issued for APPC (and we feel this is an improvement)



Diagnostics

- UNIX System Services interfaces are used. Failing service, return code and reason code are displayed in error messages
 - Look in UNIX Messages and Codes for return code (errno) descriptions
- For AT-TLS return codes, see Communication Server IP Diagnosis Reference for AT-TLS return codes
- For AT-TLS failures
 - RRSF issues an explicit message if there's something it doesn't like about the policy
 - But actual TLS handshake failures (happening under RRSF's radar) usually result in
 - 'socket exception' message. E.G.

```
IRRI031I (<) RRSF CONNECTION FROM PEER 9.57.1.13:1231 HAS BEEN  
REJECTED BECAUSE RACF COULD NOT VERIFY AT-TLS POLICY.  
THE BPX1SEL SERVICE DETECTED A SOCKET EXCEPTION.
```
 - AT-TLS trace messages on the console

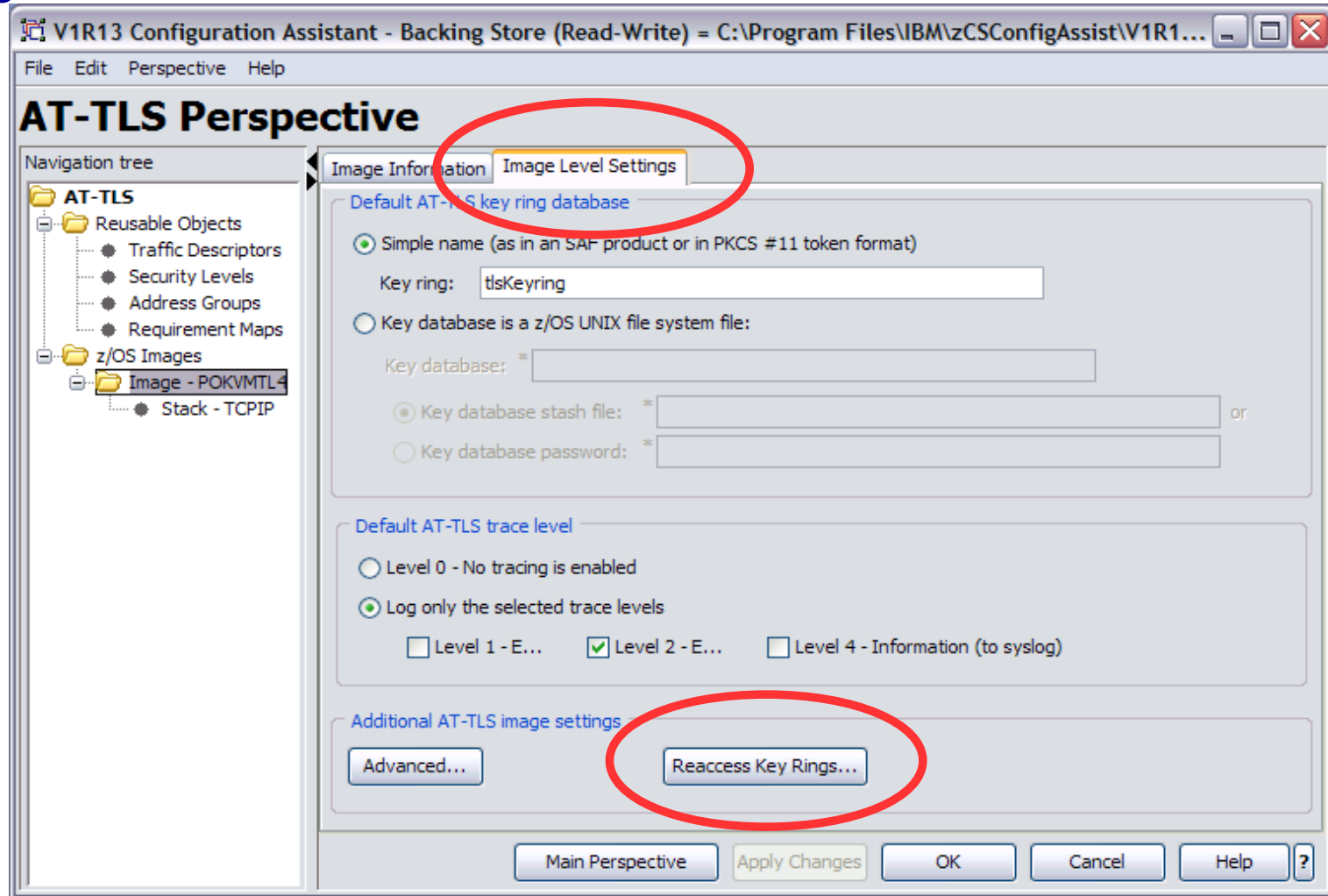


Diagnostics ...

- TLS handshake errors are almost always caused by digital certificate setup errors. On both sides of the connection, check that:
 - The key ring specified in AT-TLS policy is defined for the subsystem user ID (case matters)
 - The node's server certificate (or self-signed) is connected as the default
 - The signing CA cert (if not using self-signed server cert) is connected
 - The RACF subsystem user has authority to read its key ring (usually implemented with READ access to IRR.DIGTCERT.LISTRING in the FACILITY class)
- After making any ring changes (including authority to read it), the policy agent must be notified
 - Change EnvironmentUserInstance value in policy file (or use “Reaccess Key Rings...” button in GUI. See next slide.)
 - Refresh policy agent (“f pagent,refresh” command from console)



Diagnostics ...



Configuration Assistant screen shot showing the "Reaccess Key Rings" function

Diagnostics ...

- New SET TRACE(RRSF) operator command
- More comprehensive than existing SET TRACE(APPC), and no overlap with it
- Use at direction of RACF Level 2 when trying to debug a problem
- Don't run this way pro-actively all the time, because it will generate a **lot** of GTF records!



Diagnostics ...

- Network connectivity errors will generally be debugged using Communication Server commands and traces
- Note that TARGET LIST will display what RRSF thinks is the LAST resolved IP address for local and remote nodes
- Addresses resolved when connection (inbound or outbound) is established (true for local node also)

```
<TARGET LIST NODE(NODE2)
IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2:
STATE          - DORMANT REMOTE
DESCRIPTION    - <NOT SPECIFIED>
PROTOCOL       - TCP
                HOST ADDRESS      - ALPS4012.POK.IBM.COM
                IP ADDRESS       - 9.57.1.13
                LISTENER PORT     - 18136
TIME OF LAST TRANSMISSION TO     - <NONE>
TIME OF LAST TRANSMISSION FROM  - <NONE>
WORKSPACE FILE SPECIFICATION
```

...

...

