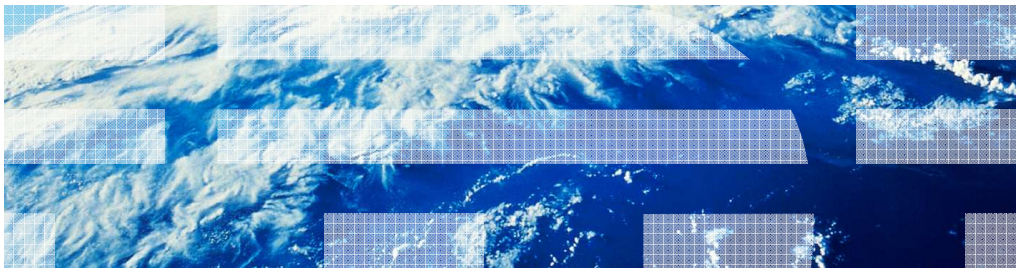
z/OS V1R13

RACF: RRSF TCP/IP support



Session objectives

- Overview of the RACF Remote Sharing Facility (RRSF)
- Simplified illustration of creating a TCP node
- Roadmap of setup steps
- Making the RACF subsystem address space a UNIX process
- Trust policy (digital certificates)
- Application Transparent Transport Layer Security (AT-TLS)
- Updates to the TARGET command
- Protocol conversions
- Diagnostics

Overview

- Problem Statement / Need Addressed
  - 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. In addition, stronger cipher algorithms are supported to protect the data while it crosses the network.

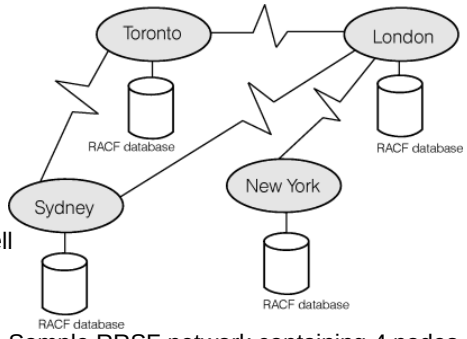
Overview – What is RRSF?

- 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

  - This line item deals exclusively with the transport layer

Overview – The RRSF network

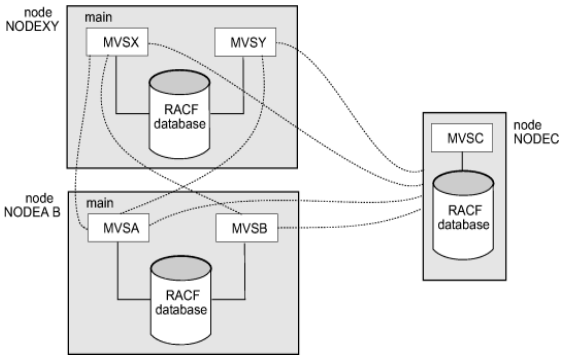
- Consists of **nodes**
  - Local node: The one I'm logged on to at the moment
  - Remote nodes (all the others)
  - Local node can run in “local mode”, where there are no remote nodes
- The TARGET operator command is used to define, modify, and delete, and list nodes, as well as to de/activate them
- 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

Overview – 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

Overview – Workspace data sets (that is, 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

The requests in the checkpoint files are masked using CDMF (Commercial Data Masking Facility), which is 40-bit DES. This is not industrial grade cryptography, but rather a protection against casual viewing. Although this line item introduced stronger encryption options than APPC as the data flows across the network, the dat at rest in the files is not changing for this line item.

Use the IRRBRW00 utility to browse the contents of the files.

Usage and invocation (1 of 2)

- To define and activate a connection using the TCP protocol, you will:
  - Set up all that TLS stuff in Communication Server
  - Establish a socket listener on the local node

```
TARGET NODE(LOCNODE) PROTOCOL(TCP) OPERATIVE

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

  - Specify the remote host and make it operative

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

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


Not to minimize the effort involved under “Set up all that TLS stuff in Communication Server”, but this slide's focus is on the actual RRSF TARGET commands used to establish TCP connections under the assumption that the environmental stuff is already set up. By “all the TLS stuff”, I mean the digital certificate deployment, the SERVAUTH class definitions/permissions, and the AT-TLS policy enablement. If you are not currently using AT-TLS, you will also be setting up the Policy Agent infrastructure. Policy Agent setup is beyond the scope of this presentation.

Usage and invocation (2 of 2)

- The implementation steps are roughly the following
  - Add OMVS segment and UID to RACF subsystem user ID, and OMVS segment and GID to its default group
  - Deploy digital certificates/key rings which are used to authenticate RRSF servers to each other using the TLS protocol
  - Enable the AT-TLS policy required for RRSF connections (sample provided)
  - Permit the RACF subsystem identity to the necessary resources
  - Using the TARGET command, establish a socket listener on each RRSF node
  - Using the TARGET command, activate the connection from both sides of each connection
  - Harden the TARGET commands in your RACF parameter library

Setup: OMVS segment for the RACF subsystem

- 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 do not, 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!

Note that we've improved on the process used for existing RACF subsystem functions that require UNIX (password/phrase enveloping and Kerberos key generation). Those require a subsystem restart after you assign a UID/GID. With RRSF, if you attempt to establish the TCP listener without having performed the UNIX setup, you'll get an error message. After you assign the UID/GID, simply establish the listener again.

Setup: Digital certificates (1 of 3)

- 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 with whom I talk, and the people who I trust) are represented by digital certificates
- For a given application, I keep my 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").

The digital certificate setup **could** be considered outside the scope of this presentation, since it's standard SSL/TLS setup. However, we will spend some time discussing it because

1RRSF is not your typical client/server network application

2A simplified setup for client authentication is possible because you are your own certificate authority, and have complete control.

3We want you to understand and trust the protection you are establishing

Setup: Digital certificates (2 of 3)

- 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 (3 of 3)

- 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.

In the extremely simplified case, you could use a single self-signed certificate which acts as both the certificate authority and the server certificate for every RRSF instance. That is, all you have to do is define the certificate on one of the systems, add it to the key ring, and then 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 may violate some customers' security policy. But if you can live with this, life will be easy.

Setup: Digital certificates ... Addendum

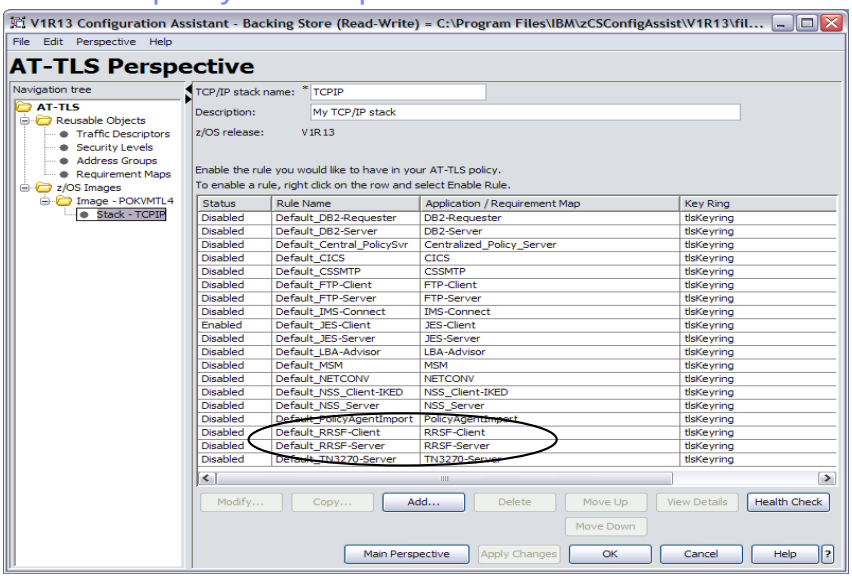
- Question: But what if my security policy forces me to obtain digital certificates from an external certificate authority?
- Answer: Patiently explain to the Powers That Be that you are configuring a RACF-to-RACF function between servers, and that the RACDCERT command has already been paid for.
- If 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

We don't think this should ever be a worry, but we have heard from at least one client that their security policy states that all certificates must be obtained from an external CA. In this case, we did not want to preclude the ability to use RRSF securely, so additional controls can be established in this case if it bothers you that you are not 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.

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.
- A working sample is provided. It just needs to be enabled and installed into the Policy Agent.
- 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



Screen capture from the Communication Server Configuration Assistant GUI

This screen shot of the Windows-based Configuration Assistant GUI shows the sample policy shipped with Communication Server in R13. Note that the Configuration Assistant is also available as a z/OSMF plug-in.

The default key ring name for all samples is tlsKeyring. Note that RACF has also shipped sample policy statements in the IRRSRRSF member of SYS1.SAMPLIB, and in these statements, a default key ring name of IRR.RRSF.KEYRING is used. Regardless of which sample you use, you can change the key ring name to anything you'd like. RRSF has no idea what the key ring name is.

Setup: AT-TLS policy ... (1 of 2)

- The sample will satisfy RRSF once (copied and) enabled. Notes of interest:
  - It consists of two rules: one when RRSF acts as the client, and one when it acts as a server
  - Client authentication is specified as “Required”
  - The AES-256 cipher algorithm is specified (strongest currently supported algorithm)
  - A listening port number of 18136 is specified. This number has been reserved with the Internet Assigned Numbers Authority (IANA)
- The policy can optionally be made more specific by adding IP addresses, and user ID or job name, of the endpoints.
  - Would force a “no policy” error rather than a “handshake error” (which will be easier to debug, if it is a bug)

Fun fact:

You can remember the listening port (not that you have to) by using the digits as an index into the alphabet:

18 = R

1 = A

3 = C

6 = F

Purely a coincidence, I assure you.

Setup: AT-TLS policy ... (2 of 2)

- 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 (let the discussion begin!), 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.

We debated enforcing some sort of minimum encryption level. For example, Triple-DES, as is used by APPC. However, we decided against it, thinking that some clients would already be using link-layer encryption (IP Sec), or communicating across LPARs using hypersockets, and thus would not want redundant or unnecessary encryption overhead.

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 PIVILEGED attributes
- As a result, the subsystem user will need to be permitted to whatever SERVAUTH class profiles are protecting resources it is accessing
  - But do **not** permit it to the stack initialization resource (INITSTACK) or remote connections may fail during IPL
- Permission will also be required to open the key ring
  - And, if the server's private key is stored in ICSF, then the appropriate CSFKEYS/CSFSERV profiles

We recommend you run the RACF subsystem TRUSTED, since RRSF needs to access unpredictably named VSAM INMSG and OUTMSG files. However, this causes a couple of problems with the SERVAUTH class:

1RRSF initialization at IPL is actually thwarted if RACF has access to the stack initialization resource (EZB.INITSTACK.**).

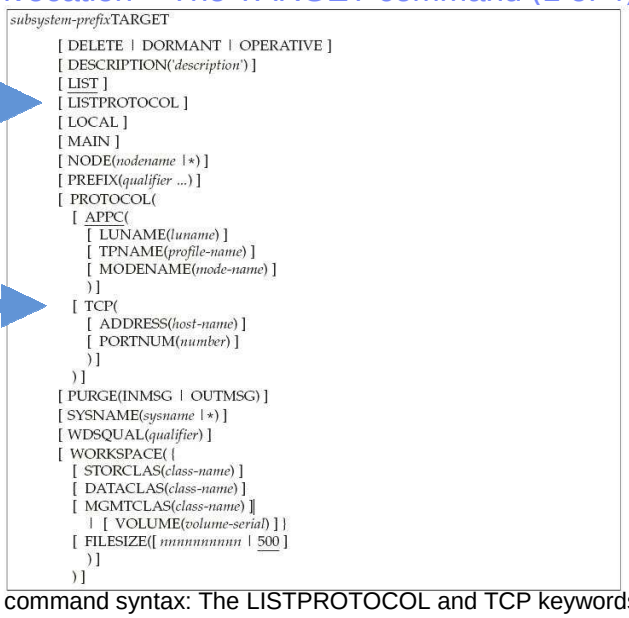
2RRSF misconfigurations could actually subvert protections established for other applications (e.g. You could assign RRSF a port that has been reserved for another application)

There is a small amount of “PERMIT creep” associated, in that RACF will need explicit permission to read its key ring, and access its keys, but it shouldn't be too bad.

Usage and invocation – The TARGET command (1 of 4)

New

New



TARGET command syntax: The LISTPROTOCOL and TCP keywords are new

This slide shows the syntax of the TARGET command, and points out that the only enhancements are a new PROTOCOL value of TCP (with its suboperands: ADDRESS and PORTNUM and), and a LISTPROTOCOL option which displays and alternative LIST output format.

Usage and invocation – The TARGET command (2 of 4)

- Now that you have all that TLS stuff set up, the actual RRSF/RACF setup is almost as trivial as alluded to previously
  - Establish the listener on the local node, which must first be dormant

```
TARGET NODE(LOCNODE) DORMANT
TARGET NODE(LOCNODE) PROTOCOL(TCP) OPERATIVE
```

  - Define remote node.

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

  - You'll get a handshaking error message because you haven't performed these steps on the remote system yet. When you have, the connection will complete.
- Now put these TARGET commands in the RACF parameter library so they are automatically executed on the IPL or restart of the subsystem

The handshaking message you see will be because you have not yet defined the node on the remote system. The remote listener will trust that this system is an RRSF node trying to connect, but will not be able to find the definition of the node name it is asserting itself to be.

Usage and invocation – The TARGET command (3 of 4)

- Notes:
  - You never need to specify the port number unless you are **not** using the default (18136). We recommend you use the default.
  - For the local node, the default host address value is INADDR_ANY (0.0.0.0), so you don't need to specify this either
  - Remote nodes must always have an ADDRESS value specified
    - This can be expressed either as an IP address, or a host name

Usage and invocation – The TARGET command (4 of 4)

- Notes:
  - You may only specify one protocol per TARGET command. So, if you have APPC nodes also (and you will if you wish to communicate with an R12 or older system), then do the following in the RACF parameter library for the local node:
    - Remove the OPERATIVE keyword from the existing statement with APPC protocol information
    - Add your TARGET command for TCP after the APPC command and specify OPERATIVE on this command.
  - E.G. Before:

```
 TARGET NODE(NODE1) PROTOCOL(APPC(LUNAME(MF1AP001))) PREFIX(NODE1.WORK) -                  WORKSPACE(VOLUME(TEMP01)
FILESIZE(500)) LOCAL OPERATIVE
```

  - E.G. After:

```
TARGET NODE(NODE1) PROTOCOL(APPC(LUNAME(MF1AP001))) PREFIX(NODE1.WORK) -
  WORKSPACE(VOLUME(TEMP01) FILESIZE(500)) LOCAL

TARGET NODE(NODE1) PROTOCOL(TCP) OPERATIVE
```

Usage and invocation – 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

A listener status of INITIALIZING means that it is attempting to open a listening socket. This process is usually more or less instant, but the listener may be waiting for TCP/IP to initialize during system IPL, for example.

Usage and invocation – TARGET LISTPROTOCOL

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

```
- 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 PROTOCOL TCP IS IN THE OPERATIVE ACTIVE STATE.
```

- Comes in handy when displaying a mixed-protocol network

Imagine you are half way through the process of converting 20 connections from APPC to TCP. LISTPROTOCOL is a nice way to summarize which protocol is being used by which connections. We didn't want to automatically display the protocol when using LIST, in case the message format change would break existing automation.

Usage and invocation – TARGET LIST: detailed version (1 of 2)

- 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)
```


Note that only a TARGET LIST of the local node can show multiple PROTOCOL sections for a single node definition. A remote node will only ever have one protocol defined, though as we will see during the protocol conversion discussion, there may actually be two separate node definitions (one for each protocol type) for a single connection as you are preparing for a conversion.

Usage and invocation – TARGET LIST: detailed version (2 of 2)

- 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      - DEFAULT_RRSF-CLIENT 1
                   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)
```


RRSF extracts AT-TLS policy information and displays it for nodes which are connected.

Use NETSTAT CONN to obtain the connection ID for a given connection, and use that in a NETSTAT TTLS CONN *xx* DETAIL command to get all the gory detail.

See the Diagnosis Guide for more details.

Usage and invocation – TCP workspace naming convention

- For local node, nothing has changed:
  - *prefix.sysname_or_wdsqual*.INMSG|OUTMSG
- 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
- For remote TCP nodes, the new convention is
  - *prefix.local-node-qualifier.wdsqual-or-nodename-or-sysname*.INMSG|OUTMSG
- This makes protocol conversions interesting.

The existing workspace data set naming convention used APPC LU names as qualifiers. This would look exceedingly dumb for TCP connections, and in fact would make the TARGET command extremely complicated if we had to provide a way for your TCP connections to derive the old APPC names. So, we've established a new naming convention for TCP that does not use protocol information.

Note that APPC continues to use the old naming convention, and that the names used by the local node are unaffected.

Usage and invocation – Protocol conversions (1 of 5)

- 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 impose a “quiet time” on the customer to allow the old files to drain before queuing work to the new files.
  - We cannot let requests run out of order
  - 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

We hope you think the conversion process is as cool as we think it is. It took a lot of effort to design and implement. In addition to solving all the problems shown on this slide, it also performs some (not all) cleanup automatically to reduce manual steps.

Usage and invocation – Protocol conversions (2 of 5)

- Here is the approach
  - Define protocol information for the local node
  - For a remote node, enter a TARGET command as though you are defining the node from scratch, specifying the new protocol information (nothing will be copied from the existing protocol)
    - Communication will continue uninterrupted using the old protocol until the new protocol establishes a connection
    - The new protocol will assume ownership of the old protocol's files
    - The old protocol instance will be automatically deleted
    - New requests will be queued to the new files while the old files are draining
    - When the old files have drained, they will automatically be deallocated and deleted
    - It will now appear as though the new protocol is the only one that ever existed

Usage and invocation – Protocol conversions (3 of 5)

- Example: From NODE2, convert NODE1's protocol from APPC to TCP
- Assumptions: TCP listeners have already been established on both systems, and NODE1 has already issued its remote node command

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

IRRC057I (>) RRSF PROTOCOL CONVERSION FROM APPC TO TCP FOR NODE
 NODE1 HAS BEEN INITIATED.
IRRM002I (>) RSWK 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.
```

- To make the conversion bullet-proof, harden the commands in the parameter library prior to issuing them on the console (see SPG for conversion procedure)

The conversion procedure has you adding/saving your new TCP-based TARGET commands in the RACF parameter library **before** issuing them on the console. This is a safeguard in case the conversion is interrupted by a RACF subsystem STOP. In this situation, when the subsystem is restarted, both protocols are still defined for the node, and the conversion is resumed.

The one annoyance is that the console command line is not very large, and yet individual TARGET commands can span lines in the parameter library. So be careful that your commands are correct in parmlib before issuing them (or some abbreviated version of them) on the console.

Usage and invocation – Protocol conversions (4 of 5)

- 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)
```


After the TCP connection is established (it is in the OPERATIVE ACTIVE state), RRSF automatically deletes the APPC node definition, and “steals” the APPC files. TCP will drain the files and then automatically deallocate and delete them.

This slide shows sample TARGET LIST output while in the process of draining the APPC files, which are designated as “CONVERSION FILE”. Both OUTMSG files are treated as a single logical file, and RRSF will insure that requests flow in order. Same for INMSG.

Usage and invocation – Protocol conversions (5 of 5)

- After the conversion completes, there will be no trace left of the APPC node:

```
<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.
```


All you have to do after the conversion is complete is delete the APPC commands out of the RACF parameter library. In fact, you don't even have to. If you don't, the next time the subsystem is started, new APPC files will be created and APPC will attempt to speak, but will eventually be taken over by TCP as though a conversion were taking place. It'll be a bit noisier, and you'll need enough DASD space to do all that file allocation, but should otherwise be harmless.

Usage and invocation – New and changed messages

- Too numerous to list
- 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

The set of messages issued by TCP nodes might be a tad more numerous than those issued by APPC, but we feel they provide a much more straightforward picture of what's actually occurring in the RACF subsystem. APPC message processing was not changed.

Usage and invocation – Diagnostics (1 of 5)

- 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

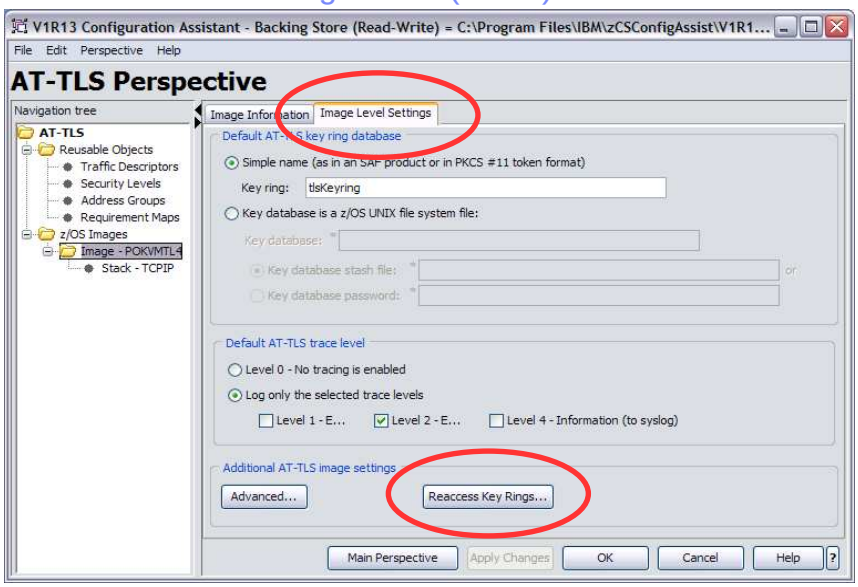
Due to the nature of TCP/IP sockets, you will occasionally see things like receive (BPX1RCV) failures as connections are broken, and these may not seem intuitive. However, before investing a lot of time in looking up messages and return codes, just do a quick TARGET LIST on the node (or look for subsequent console messages), because RRSF does its best to automatically re-establish broken connections, and there may not even be a problem by the time you look.

Usage and invocation – Diagnostics (2 of 5)

- 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)

In my experience, TLS handshake errors give the least intuitive/helpful error messages. All of this processing occurs under RRAF's radar (hence Application **Transparent** TLS), and hen there is an authentication failure, TCP/IP tends to close connections without giving the application any sort of detailed reason why. However, these errors are almost always caused by key ring problems, which can be quickly sanity-checked before diving into diagnosis books or opening PMRs.

Usage and invocation – Diagnostics (3 of 5)



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

In my opinion, one of the least intuitive policy externals is the mechanism by which you force your key ring to be re-accessed after making a change to it. In fact, I got burned several times by not even knowing that I had to do such a thing; scratching my head as to why I still got the same (non-intuitive) AT-TLS error after thinking that I had fixed my problem. So hopefully you won't waste time on the same sort of problem!

Usage and invocation – Diagnostics (4 of 5)

- 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
- Do not run this way proactively all the time, because it will generate a **lot** of GTF records!

We do not document the format of the GTF records produced by SET TRACE(RRSF). They may help IBM Level 2 understand the processing flow that occurs as you are reproducing a problem (not that there will be any, of course).

Usage and invocation – Diagnostics (5 of 5)

- 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
...

...
```


The note about the local node is important because you might **think** that after having established a listener, but before initiating a remote connection, that the listener (local node) address would have been resolved. But it wouldn't have been.

Interactions and dependencies

- Software Dependencies
  - None
  - The z/OS Communication Server will ship sample AT-TLS policy in their Configuration Assistant
- Hardware Dependencies
  - None
- Exploiters
  - None

Migration and coexistence considerations

- APPC continues to be supported
- APPC and TCP/IP can coexist in a mixed protocol environment: any given RRSF node can speak APPC to one set of nodes/systems, and TCP/IP to the others.
- A Multi-System Node (MSN) can consist of mixed protocol peer systems, and even continue to share a RACF parameter library
  - Might be a bit “noisy” using shared parmlib while pre-R12 systems are included, but will still work
- No compatibility/toleration APARs are necessary on R12 and lower systems.

There may be additional MSN considerations in the area of protocol conversions, but these have not been fully resolved in time for T3 submission. If you are seeing this note while I am presenting, please ask me about them! Otherwise, have no fear, they will be covered in the System Programmer's Guide.

Installation

- No considerations
- But you do want to be planning on how/when you will convert your network protocols, and how you will deploy the requisite digital certificates

You could actually deploy your key rings and certificates in your RACF database(s) on a pre-R13 system if you want to get a jump on it. You just won't be able to verify that it actually works.

Session summary

- Overview of the RACF Remote Sharing Facility (RRSF)
- Simplified illustration of creating a TCP node
- Roadmap of setup steps
- Making the RACF subsystem address space a UNIX process
- Trust policy (digital certificates)
- Application Transparent Transport Layer Security (AT-TLS)
- Updates to the TARGET command
- Protocol conversions
- Diagnostics

Appendix – References (1 of 2)

- 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 – References (2 of 2)

- 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

Trademarks, disclaimer, and copyright information