z/OS 3.2 IBM Education Assistant

Solution Name: RACF Extend JWT Support

Solution Element(s): z/OS Security Server RACF

July 2025

Agenda

- Trademarks
- Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Upgrade & Coexistence Considerations
- Installation & Configuration
- Summary
- Appendix

Trademarks

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

Objectives

The Challenge: Prior to this support RACF Identity Tokens signatures can only be generated and validated with HMAC algorithms with keys from ICSF PKCS#11 tokens.

When using symmetric HMAC keys for signatures, both the JWT generator and validator must use the same key material. Distributing the key material securely is a challenge.

Asymmetric key based signatures with RSA keys presents a more streamlined and well understood way to distribute cryptographic keys using x509 digital certificates. The JWT consumer only needs the public key from the certificate to validate the JWT signature.

The Objective: Add support to RACF IDT processing to generate and validate IDT signatures with RSA from secure ICSF CCA key labels.

Add support to RACF IDT processing to generate and validate IDT signatures with HMAC keys from clear and secure ICSF CCA key labels.

IDT Overview – Intro to IDT Support

Identity Token:

- An Identity Token is used to encode details about a user which can be trusted by the consumer of the token.
- Our use adheres to the JSON Web Token (JWT) IETF specifications: RFC 7519

RACROUTE Support for Identity Tokens:

- Support is added to RACROUTE authentication processing to generate and validate Identity Tokens (IDT).
- **Generation -** An application can call RACROUTE or initACEE and request that an IDT be returned.
- **Validation -** An application can call RACROUTE to authenticate a user with an IDT specified instead a credential like a password.

IDT Configuration:

- The security administrator can create profiles in the IDTDATA class to configure how certain fields in an IDT are generated and validated.
- The profiles can be used to control options such as which key is used to sign the IDT and its validity period.

Existing Identity Token Use Cases (IDT1) (1 of 2)

- **Replaying Proof of Authentication:**
  - Some applications authenticate a user and “replay” that authentication multiple times.
- **Problem:**
  - Some applications cache the user provided credential and replay it back again later.
  - For users with one time use MFA tokens, this does not work.
- **Solution:**
  - The Identity Token support allows applications to authenticate a user and receive proof of that authentication which can be supplied back to RACROUTE in place of other credentials like a password.
  - Signed JWTs can be returned to an end user for later use by the application.

Existing Identity Token Use Cases (IDT1) (2 of 2)

- **Linking Multiple Authentication API Calls:**
  - In some cases, user authentication requires multiple steps:
  - Expired Password / Invalid New Password / MFA Expired PIN …
- **Problem:**
  - MFA credentials are one-time use.
  - When multiple authentication calls are required, an already consumed MFA token will fail.
- **Solution:**
  - The Identity Token can be used to link authentication status information between multiple authentication API calls without replaying the MFA credentials.

Existing Identity Token Use Cases (IDT2)

- **Generate IDT from existing ACEE security environment:**
  - In some cases, an application needs to flow work onto another LPAR for a protected user.
- **Problem:**
  - Protected users can not be authenticated with password/PassTicket/JWT.
- **Solution:**
  - Applications can generate an IDT for an existing ACEE security environment (for a protected or non-protected user ID) using the SAF initACEE callable service.
  - Protected users can be authenticated with RACROUTE REQUEST=VERIFY with a JWT generated by initACEE.

IDT3 Overview

- **Who (Audience)**
  - RACF installations who generate and validate RACF Identity Tokens (IDT).

- **What (Solution)**
  - Starting with OA65299 (RACF) and OA66783 (SAF) for z/OS 3.1:
    - Add support to generate and validate RACF IDTs with RSA signatures using secure ICSF CCA key labels.
    - Add support to generate and validate RACF IDTs with HMAC signatures using clear and secure ICSF CCA key labels.

- **Wow (Benefit / Value, Need Addressed)**
  - More secure and streamlined key distribution for RSA keys via digital certificates.
  - More installations use CCA secure keys (than EP11), which provides more security for HMAC keys vs using clear keys with PKCS#11 tokens.

IDT3 - Externals Overview

New IDT Configuration - IDTPARMS Keywords:

RDEFINE/RALTER:

- New IDTPARMS segment keywords: SIGLABELPRIMARY() & SIGKIDPRIMARY()
- New RSA values for existing IDTPARMS keyword SIGALG

RACF Templates:

- New fields in IDTPARMs segment

R_admin:

- New IDTPARMS keywords for extract/set functions

RACF Database Unload Utility:

- New IDTPARMS segment fields unloaded

New IDT Processing - RSA signatures and HMAC signatures with CKDS key labels:

RACROUTE REQUEST=VERIFY,ENVIR=CREATE:

- Updated IDT processing: RSA signatures & HMAC signatures with CKDS CCA key labels.
- Two new return/reason code combinations
- New IDTA Generation Return Code

SMF Auditing & SMF Unload Utility:

- New fields in SMF 80 Event Code 1 (JOBINIT) Relocate 443

initACEE:

- Updated IDT processing: RSA signatures & HMAC signatures with CKDS CCA key labels.
- New Return/Reason code combination

IDT3 - Configuration

IDTDATA Class profiles and IDTPARMS segment:

- Security administrators can control the use of tokens by defining profiles in the new IDTDATA general resource class, using a new IDTPARMS segment

IDTDATA class:

- Must be **ACTIVE** before Identity Tokens will be generated or validated
- Must be **RACLISTed** before any profiles in the class will be used

IDTDATA profile format: <IDT Type>.<application name>.<user ID>.<IDT issuer name>

- IDT Type: “JWT”
- Application name: The value specified in the APPL= parameter
- User ID: The user being authenticated
- IDT issuer name: “SAF”

Note: Generics are allowed. When a user is authenticated with a JWT, the best matching profile is used.

IDT3 – New IDTPARMS Keywords

IDTPARMS segment RALTER command keywords

[IDTPARMS(

[SIGTOKEN(*pkcs11-token-name*) | NOSIGTOKEN]

[SIGSEQNUM(*pkcs11-sequence-number*) | NOSIGSEQ]

[SIGCAT(pkcs11-category) | NOSIGCAT]

→ Location of the signing key (PKCS#11 TKDS HMAC key) **Note:** Does not support RSA

[SIGLABELPRIMARY(*primary-label*)] → Location of the signing key (CCA CKDS RSA or HMAC key)

[SIGKIDPRIMARY(*primary-kid*)] → Key Identifier (KID) of SIGLABELPRIMARY

[SIGALG(HS256 | HS384 | HS512 |

RS245 | RS384 | RS512) | NOSIGALG] → Signature algorithm to use

[ANYAPPL(YES | NO)] → Whether IDTs can be used by other applications

[IDTTIMEOUT(*timeout-minutes*)] → Validity interval of a token

[PROTALLOWED (YES | NO)] → Whether IDT can authenticate a protected ID

)

NOIDTPARMS]

IDT3 - New IDTPARMS Keywords - Templates

The IDTPARMS segment in the GENERAL section is updated to add new fields:

```
$/SEGMENT  021 IDTPARMS
IDTPARMS  001 00 00 00000000 00 IDTPARMS - Start of segment fields
IDTTOKN   002 00 00 00000000 00 IDTPARMS - PKCS#11 Token Name
IDTSEQN   003 00 00 00000000 00 IDTPARMS - PKCS#11 Sequence Number
IDTCAT    004 00 00 00000000 00 IDTPARMS - PKCS#11 Category
IDTSALG   005 00 00 00000000 00 IDTPARMS - Signature Algorithm
IDTTIMEO  006 00 00 00000004 00 IDTPARMS - IDT Timeout
IDTANYAP  007 00 00 00000001 80 IDTPARMS - IDT Any Application
IDTPROTA  008 00 00 00000001 80 IDTPARMS - IDT Protected allowed
IDTLABP   009 00 00 00000000 00 IDTPARMS - Primary Label
IDTKIDP   010 00 00 00000000 00 IDTPARMS - Primary KID
IDTLABB   011 00 00 00000000 00 IDTPARMS - Reserved
IDTKIDB   012 00 00 00000000 00 IDTPARMS - Reserved
```

New IDTPARMS Keywords – R_Admin

Field name	SAF field name	Flag byte value	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
SIGTOKEN	sigtoken	'Y'	IDTPARMS (SIGTOKEN(xx))	Yes	Yes	Yes
		'N'	IDTPARMS (NOSIGTOKEN)	No	Yes	
SIGSEQN	sigseqnum	'Y'	IDTPARMS (SIGSEQNUM(xx))	Yes	Yes	Yes
		'N'	IDTPARMS (NOSIGSEQNUM)	No	Yes	
SIGCAT	sigcat	'Y'	IDTPARMS (SIGCAT)	Yes	Yes	Yes
		'N'	IDTPARMS (NOSIGCAT)	No	Yes	
SIGALG	sigalg	'Y'	IDTPARMS (SIGALG(xx))	Yes	Yes	Yes
		'N'	IDTPARMS (NOSIGALG)	No	Yes	
IDTTIMEO	idttimeout	'Y'	IDTPARMS (IDTTIMEOUT(xx))	Yes	Yes	Yes
ANYAPPL	anyappl	'Y'	IDTPARMS (ANYAPPL(YES))	Yes	Yes	Yes
		'N'	IDTPARMS (ANYAPPL(NO))	Yes	Yes	
IDTPROTA	idtprota	'Y'	IDTPARMS (PROTALLOWED(YES))	Yes	Yes	Yes
		'N'	IDTPARMS (PROTALLOWED(NO))	Yes	Yes	
SIGLABP	siglabp	'Y'	IDTPARMS (SIGLABELPRIMARY(YES))	Yes	Yes	Yes
		'N'	IDTPARMS (SIGLABELPRIMARY(NO))	Yes	Yes	
SIGKIDP	sigkidp	'Y'	IDTPARMS (SIGKIDPRIMARY(YES))	Yes	Yes	Yes
		'N'	IDTPARMS (SIGKIDPRIMARY(NO))	Yes	Yes	

R_Admin Updates:

- Add new IDTPARMS segment keywords:
  - SIGLABELPRIMARY()
  - SIGKIDPRIMARY()

New IDTPARMS Keywords - DBUNLOAD

Field Name	Type	Start	End	Comments
GRIDTP_RECORD_TYPE	Int	1	4	Record type of the Identity Token data record (05K0)
GRIDTP_NAME	Char	6	251	General resource name as taken from the profile name.
GRIDTP_CLASS_NAME	Char	253	260	Name of the class to which the general resource profile belongs, namely IDTDATA.
GRIDTP_SIG_TOKEN_NAME	Char	262	293	The ICSF PKCS#11 token name.
GRIDTP_SIG_SEQ_NUM	Char	295	302	The ICSF PKCS#11 sequence number.
GRIDTP_SIG_CAT	Char	304	307	The ICSF PKCS#11 category.
GRIDTP_SIG_ALG	Char	309	340	The signature algorithm.
GRIDTP_TIMEOUT	Int	342	351	IDT timeout setting.
GRIDTP_ANYAPPL	Char	353	355	Is the IDT allowed for any application? Valid values include “Yes” and “No”.
GRIDTP_PROTECTED_ALLOWED	Char	358	361	Is the IDT allowed to authenticate a protected user? Valid values include “Yes” and “No”.
GRIDTP_SIG_LABEL_PRI	Char	363	426	ICSF label name of the primary key.
GRIDTP_SIG_KID_PRI	Char	428	459	Key identifier of the primary key.

DBUNLOAD Updates:
The general resource IDTPARMS definition record (05K0) is updated to add new fields.

IDT3 – Authentication Updates – RSA Signatures

RACROUTE and initACEE IDT RSA Signatures:

- Using the new SIGLABELPRIMARY keyword, installations can specify an RSA CCA key label in the PKDS and one of the RSA SIGALG values.
- RACF does not generate the RSA key, it must be created by the installation in the ICSF CCA PKDS.
- A CCA cryptographic coprocessor is required to use RSA keys.
- The SIGLABELPRIMARY() keyword overrides any value in the SIGTOKEN() keyword.

Example:

```
RDEF IDTDATA JWT.APPL01.*.SAF IDTPARMS(SIGLABELPRIMARY(ICSF.RSAKEY01)
SIGKIDPRIMARY(MYRSAKEY01) SIGALG(RS512))


RLIST IDTDATA JWT.APPL01.*.SAF IDTPARMS
...
IDTPARMS INFORMATION
--------------------
Signature Algorithm = RS512
Signature Label Primary = ICSF.RSAKEY01
Signature KID Primary = MYRSAKEY1
IDT TIMEOUT = 00000005
ANYAPPL = NO
Protected Allowed = NO
```

IDT3 – RSA Key Provisioning Example

RACDCERT to provision RSA Key Pair:

- Generate a key pair in the PKDS with RACDCERT:
```
RACDCERT GENCERT WITHLABEL('TEST.RSA2048') RSA(PKDS(*))
SIZE(2048) SUB(CN('TEST.RSA2048'))
```
  - **Note:** RACDCERT ADD with PKDS keyword can be used to add an existing certificate and keys to the PKDS.
- Define a covering IDTDATA profile:
```
RDEF IDTDATA JWT.TSOIM13.*.SAF
IDTPARMS(SIGLABELPRIM('TEST.RSA2048') SIGALG(RS256))
```
- Generated JWT RACROUTE REQUEST=VERIFY for RACFU01:
  Header: `{"alg": "RS256"}`
  Body:
```
{"jti":"00E04D2A6355877296000000B500001","txn":"00E04D2A6355877
2960000000B500001","iss":"saf","sub":"RACFU01","aud":["TSOIM13",
"*ANYAPPL*"],"iat":1736963652,"exp":1736963952,"amr":["saf-
pwd"]}
```
  Signature: `Binary RSA signature`

IDT3 – Authentication Updates – CCA HMAC Signatures

RACROUTE and initACEE IDT CCA HMAC Signatures:

- Using the new SIGLABELPRIMARY keyword, installations can specify an HMAC CCA key label in the CKDS and one of the HMAC SIGALG values.
- RACF does not generate the HMAC key, it must be created by the installation in the ICSF CCA CKDS.
- Both clear keys and secure keys are supported. Secure key requires a CCA cryptographic coprocessor.
- The SIGLABELPRIMARY() keyword overrides any value in the SIGTOKEN() keyword.

Example:

```
RDEF IDTDATA JWT.APPL02.*.SAF IDTPARMS(SIGLABELPRIMARY(ICSF.HMACKEY01)
SIGKIDPRIMARY(MYHMACKEY01) SIGALG(HS512))


RLIST IDTDATA JWT.APPL02.*.SAF IDTPARMS
...
IDTPARMS INFORMATION
--------------------
Signature Algorithm = HS512
Signature Label Primary = ICSF.HMACKEY01
Signature KID Primary = MYHMACKEY1
IDT TIMEOUT = 00000005
ANYAPPL = NO
Protected Allowed = NO
```

IDT3 – RACROUTE Updates – KID - Key Identifier

Using the new SIGKIDPRIMARY keyword, installations can configure a KID value to be logically associated with the key identified by SIGLABELPRIMARY.

- The consumer of the IDT can use the KID claim to help identify the appropriate key for signature validation.

KID Claim use in IDT Generation:

- When an IDT is signed with the SIGLABELPRIMARY, the value identified by SIGKIDPRIMARY is set in the KID claim in the IDT.

KID Claim use in IDT Validation:

- When an IDT contains a KID claim and the IDTDATA class profile contains a SIGKIDPRIMARY:
  - These values must match before the key identified by SIGLABELPRIMARY is used to validate the signature.
  - When they do not match, signature validation is not attempted, and a failure return code is returned

Note: The SIGKIDPRIMARY value is not used when an IDT is signed by or validated with a key from the SIGTOKEN keyword.

IDT3 – New JWT Generation Return Codes

JWT Generation Failures:

- RACROUTE authentication does not fail with a bad RC when a JWT can not be generated only the IDTA generation RC is set.
- initACEE fails when a JWT can not be generated.

IDTA Generation Return Code:

- When IDT generation fails in RACROUTE REQUEST=VERIFY or initACEE the IDTA control block contains a RC:
  - **New IDTA_IDT_Gen_RC value: X'0007' – IDTA_IDT_GEN_RC_ALG_ERR** - Signature Algorithm not valid.
    - SIGTOKEN is set, SIGLABELPRIMARY is not set and SIGALG is set to one of the RSA signature algorithms.

initACEE JWT Generation Return Codes:

- **SAF Return Code: 8 & RACF Return Code 16:**
  - **New RACF Reason code:** 7 – Signature Algorithm not valid.
    - SIGTOKEN is set, SIGLABELPRIMARY is not set and SIGALG is set to one of the RSA signature algorithms.

IDT3 – New JWT Validation Return Codes

RACROUTE JWT Validation Return Codes:

- **Existing SAF Return Code:** 8 & **RACF Return Code (hex):** 6C
  - Indicates that Identity Token (IDT) processing failed while attempting to validate an IDT. RACROUTE sets the IDTA_IDT_Len to zero. The calling application should reauthenticate the user.
  - **New RACF Reason code (hex):** 1C – KID is not valid.
    - This can occur if there is an error detected parsing the JSON KID claim.
  - **New RACF Reason code (hex):** 1D – IDT is signed but KID claim does not match the KID configured in IDTDATA class profile.
    - User is being authenticated with a signed IDT which contains a KID claim and the SIGKIDPRIMARY has a value, which does not match.

IDT3 – RACROUTE Updates – SMF 80 Records

RACROUTE REQUEST=VERIFY,ENVIR=CREATE generates SMF Type 80 Event Code 1 (JOBINIT) records.

Relocate 443 contains details about the authentication event including IDTs.

Relocate 443 is updated to add more details:

- Flag byte 4 in existing relocate 443 is updated to use reserved Bit 4.
- Relocate 443 is extended to add bytes 49-537.
- New fields are added to existing relocate 443:
  1. Signature algorithm from IDT
  2. Key Identifier (KID) from provided IDT
  3. Indication on which kind of key was used to authenticate the IDT (PKCS#11 token or CCA label)

Interactions & Dependencies

- Software Dependencies
  - ICSF is required for signed JWTs
- Hardware Dependencies
  - CCA coprocessor is required for RSA JWT signatures.
  - **Note:** PKCS#11 and CCA HMAC signatures support both clear and secure keys.
- Exploiters
  - BCPii and HMC – Plan to generate RSA based JWTs in BCPii to use to authenticate to the HMC.

Upgrade & Coexistence Considerations (1 of 2)

- To exploit this solution, all systems in the Plex:
  - Must be at the new z/OS level (or)
  - Must be on z/OS 3.1 with PTFs for OA65299 and OA66783 applied
- List any toleration/coexistence APARs/PTFs: None
- List anything that doesn't work the same anymore:
  - New function: When configured, RACF can generate and validate JWTs with RSA signatures.

Upgrade & Coexistence Considerations (2 of 2)

Note that when RACF configuration is shared via shared database or is propagated in an RRSF network, behavior on each system depends on the release and service level.

- If a command is propagated to a system without the support introduced by APAR OA65299 and specifies the SIGLABELPRIMARY, SIGKIDPRIMARY, or SIGALG(RS256|RS384|RS512), the command will fail on the remote system.
- If an IDT is generated with an RSA signature and propagated to a system without the support introduced by APAR OA65299, evaluation of the IDT will fail.
- When a system without the support introduced by APAR OA65299 sets the SIGLABELPIMARY or SIGKIDPRIMARY, a system sharing the RACF database without the required support for the SIGLABELPRIMARY and SIGKIDPRIMARY keywords will not attempt to use them for IDT generation or validation.
- When a system without the support introduced by APAR OA65299 sets an RSA signature algorithm, a system sharing the RACF database without the required support for RSA signatures will default to HS256.

Installation & Configuration

- Are any APARs or PTFs needed for enablement? Not on 3.2.
  - Function is also available on 3.1 via PTFs for OA65299 and OA66783
- What jobs need to be run? None
- What hardware configuration is required? None
- What PARMLIB statements or members are needed? None
- Are any other system programmer procedures required? No
- Are there any planning considerations? No
- Are any special web deliverables needed? No
- Does installation change any system defaults? No

Summary

- RACF IDT support is updated to be able to generate and validate JWTs with RSA signatures.
  - This provides a more secure and streamlined key distribution method using digital certificates.

- RACF IDT support is updated to be able to generate and validate JWTs with HMAC signatures using ICSF CCA (in addition to the existing support for ICSF PKCS#11 token based HMAC signatures).
  - More installations use CCA secure keys (than EP11), which provides more security for HMAC keys vs using clear keys with PKCS#11 tokens.

Appendix (1)

IBM Publications

- Security Server RACF Callable Services
- Security Server RACF Command Language Reference
- Security Server RACOUTE Macros Reference
- Security Server RACF Security Administrator's Guide
- Security Server RACF Macros and Interfaces

RACF APAR Details:

https://www.ibm.com/support/pages/apar/OA65299

SAF APAR Details:

https://www.ibm.com/support/pages/apar/OA66783

Appendix (2)

Terminology:

- **Identity Token (IDT)** – Token which represents an authenticated user.
- **JSON Web Token (JWT)** – Specific implementation of a type of identity token.

Appendix - JWT RFCs and Reference Doc

- rfc7515 - JSON Web Signature (JWS)
  - https://datatracker.ietf.org/doc/html/rfc7515
- rfc7516 - JSON Web Encryption (JWE)
  - https://datatracker.ietf.org/doc/html/rfc7516
- rfc7518 - JSON Web Algorithms (JWA)
  - https://datatracker.ietf.org/doc/html/rfc7518
- rfc7519 - JSON Web Token (JWT)
  - https://datatracker.ietf.org/doc/html/rfc7519
- rfc7523 - JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants
  - https://datatracker.ietf.org/doc/html/rfc7523
- JSON Object Signing and Encryption (JOSE)
  - https://www.iana.org/assignments/jose/jose.xhtml

IDT – JWT Example (1)

JWT:

- **Header – (JOSE):**
  - {“alg”:“HS256”}
- **Body Claims – (JWS Payload):**
  - {“jti”:“cb05…”,“iss”:“saf”,“sub”:“USER01”,“aud”:“CICSLP8”,“exp”:1486744112.473}
- **Signature - (JWS):**
  - x’389A21CD32108C3483DA’

JWT base64 Encoded:

- Base64url(JOSE Header).Base64url(JWS Payload).Base64url(JWS Signature)
- 2Ce23xe490fG989N.m239vm3N29U2n2pN9mA02l32lMn.8Nw5JlK2nQnc241Okl

IDT – JWT Example (2)

- **JWT:**
  - **Header (JOSE):**
    - **{“alg” : “RS256” or “none”** – Signature Algorithm: **RS256** = RSA with **SHA-256**, none = unsecured
    - **“kid” : ”mykey01” }** – Logically identifies the key that has signed this JWT
  - **Body Claims – (JWS Payload):**
    - **{“jti” : “cb05…”,** – JWT Unique identifier
    - **“iss” : “saf”,** – Issuer name – Entity that created the JWT
    - **“sub” : “USER01”,** – Subject (the authenticated user)
    - **“aud” : “CICSLP8”,** – Audience – Target consumer of the JWT
    - **“exp” : 1486744112,** – Expiration time - (Seconds since 1970 - Expired tokens should be rejected)
    - **“iat”: 1486740112,** – Issued at – The time at which the JWT was issued.
    - **“amr”:[”mfa-comp”,”saf-pwd”]}** – Authentication Method References - How the subject was authenticated
  - **Signature (JWS)** – Encoded in Binary
    - 389A21CD32108C3483DA…