Digital Signatures for z/OS Software Packages

Kurt Quackenbush

IBM

z/OSMF Software Management and SMP/E

Poughkeepsie, NY, USA

kurtq@us.ibm.com

Agenda

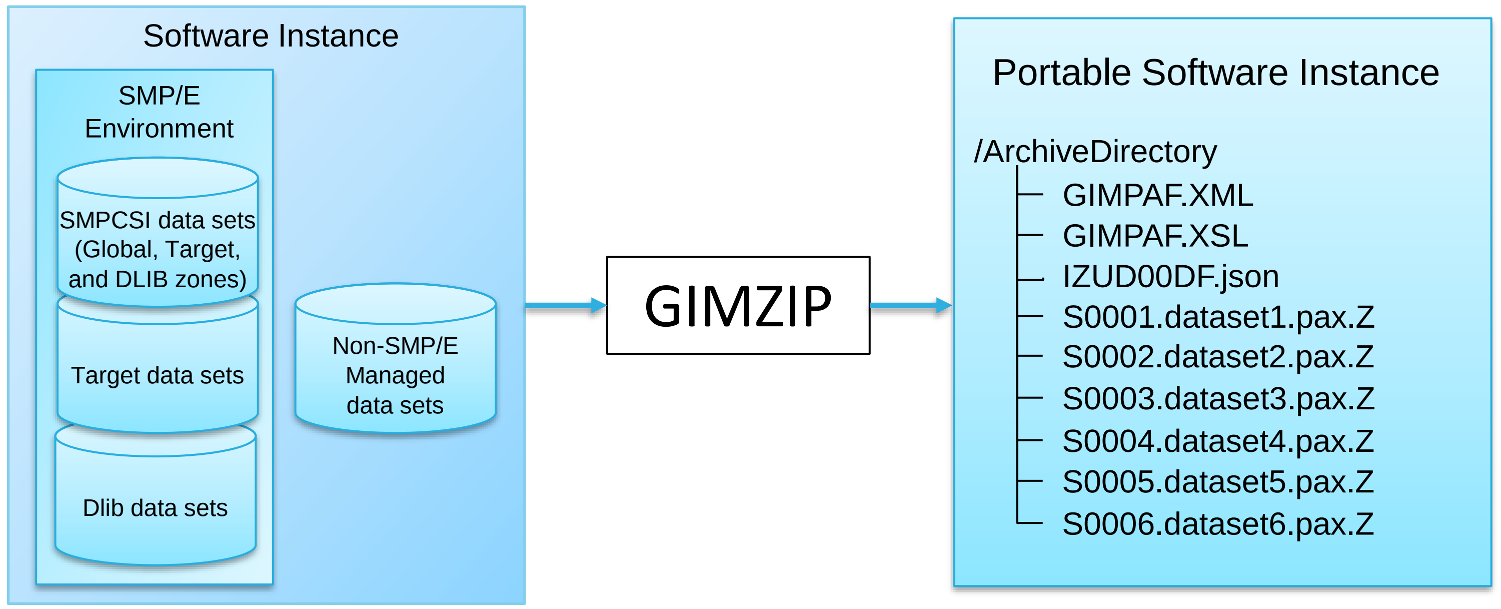
- Digital Signature Background
- GIMZIP Package Signing Overview
- Details for a Provider
- Details for a Consumer

z/OS Software Package Digital Signatures

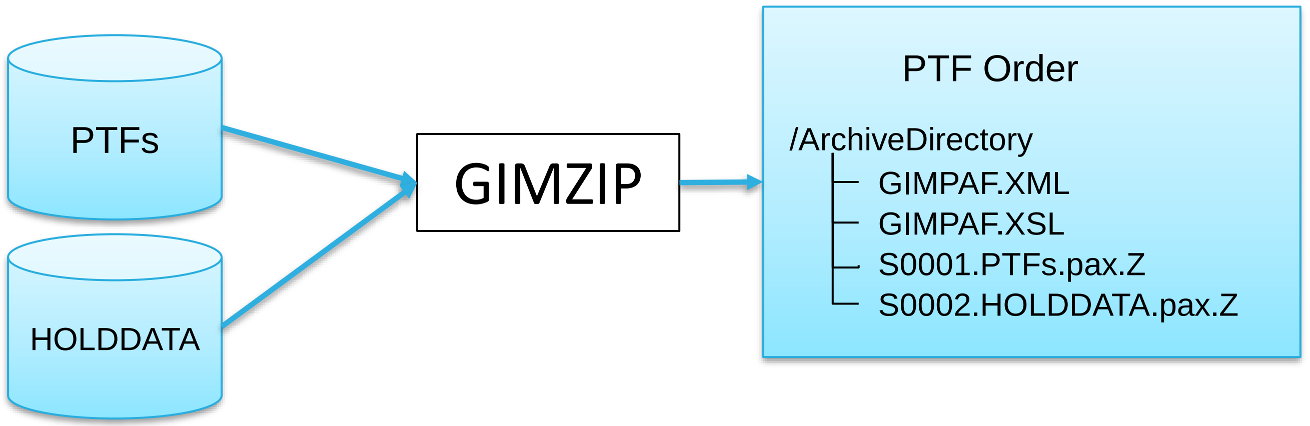
- The SMP/E **GIMZIP** service routine **creates portable packages** of ready to install SMP/E consumables, or already installed software.
- GIMZIP packages are **delivered from IBM and other providers** to customers over the internet.
- GIMZIP is currently in use by IBM for all z/OS software product and service deliveries:
  - z/OSMF Portable Software Instances (ServerPac)
  - CBPDO
  - Shopz PTF orders
  - SMP/E RECEIVE ORDER PTF and HOLDDATA
- GIMZIP packages are **consumed by SMP/E and z/OSMF on a customer's z/OS** system where the packaged software is installed.

z/OS Software Package Digital Signatures

A z/OSMF Portable Software Instance is a GIMZIP package.



An SMP/E RECEIVE ORDER package is a GIMZIP package.



z/OS Software Package Digital Signatures

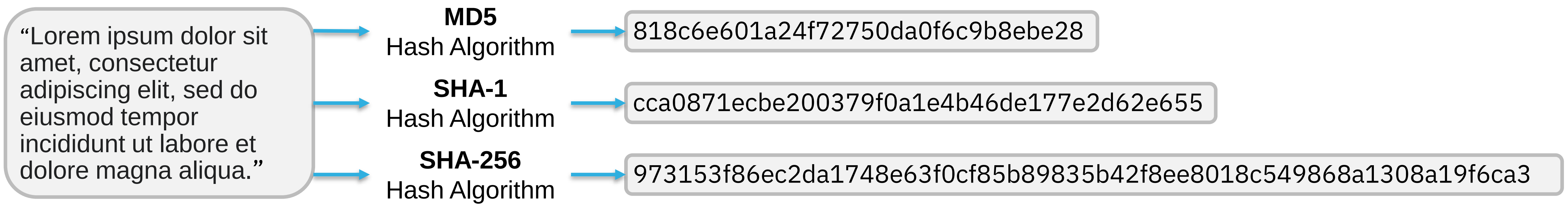
- GIMZIP is extended to **digitally sign packages**.
- z/OSMF is extended to exploit GIMZIP digital package signing for Portable Software Instances.
- SMP/E and z/OSMF are extended to **verify package signatures**.

- Verifying digitally signed software packages increases confidence in **authenticity** (who produced it?) and **integrity** (has it changed in transit?) of the software delivered in those packages.

Digital Signature Background

What is a Hash Algorithm?

- A mathematical function to convert input data of arbitrary length to a unique output bit string of a fixed length.
- Hash values are irreversible.

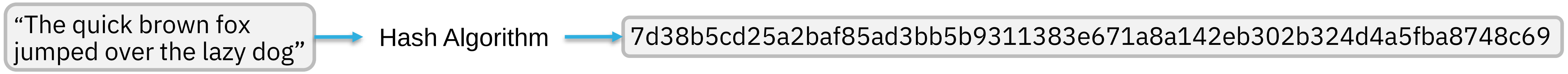


“Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.”

MD5 Hash Algorithm → 818c6e601a24f72750da0f6c9b8ebe28

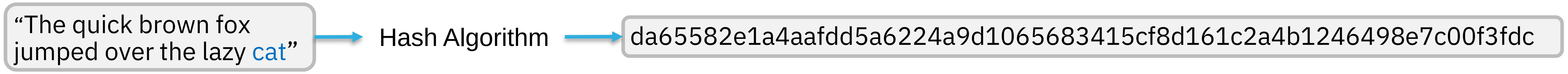
SHA-1 Hash Algorithm → cca0871ecbe200379f0a1e4b46de177e2d62e655

SHA-256 Hash Algorithm → 973153f86ec2da1748e63f0cf85b89835b42f8ee8018c549868a1308a19f6ca3



“The quick brown fox jumped over the lazy dog” → Hash Algorithm → 7d38b5cd25a2baf85ad3bb5b9311383e671a8a142eb302b324d4a5fba8748c69

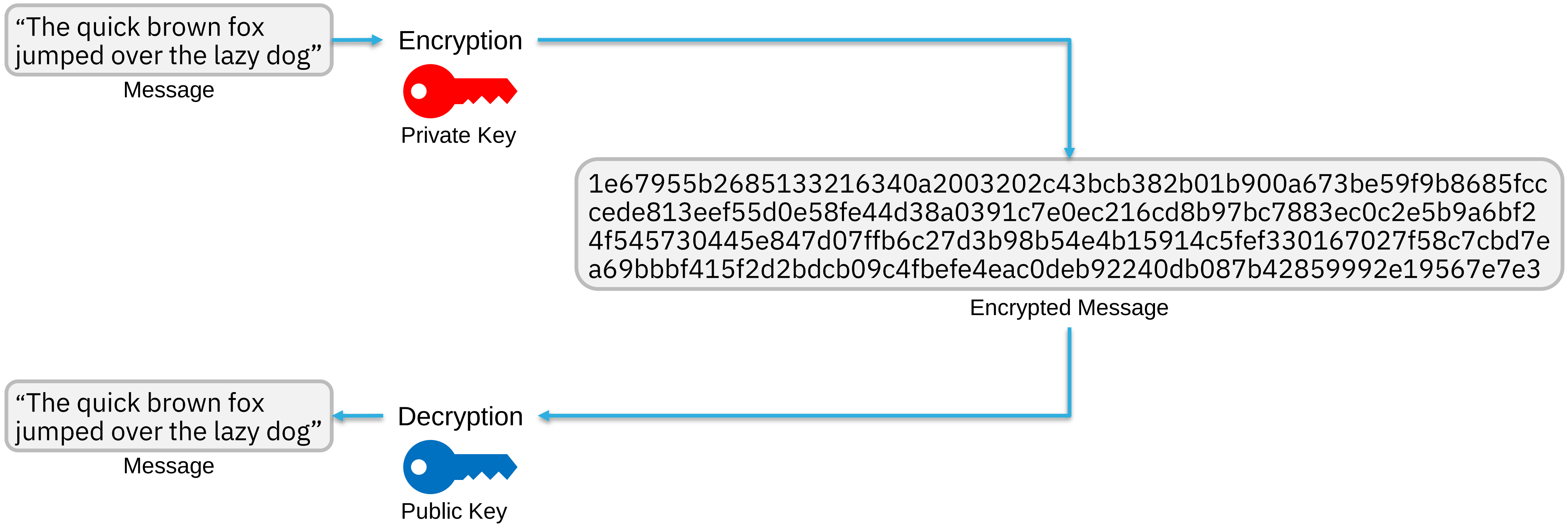
Any difference in input data, large or small, produces a different hash value.



“The quick brown fox jumped over the lazy cat” → Hash Algorithm → da65582e1a4aafdd5a6224a9d1065683415cf8d161c2a4b1246498e7c00f3fdc

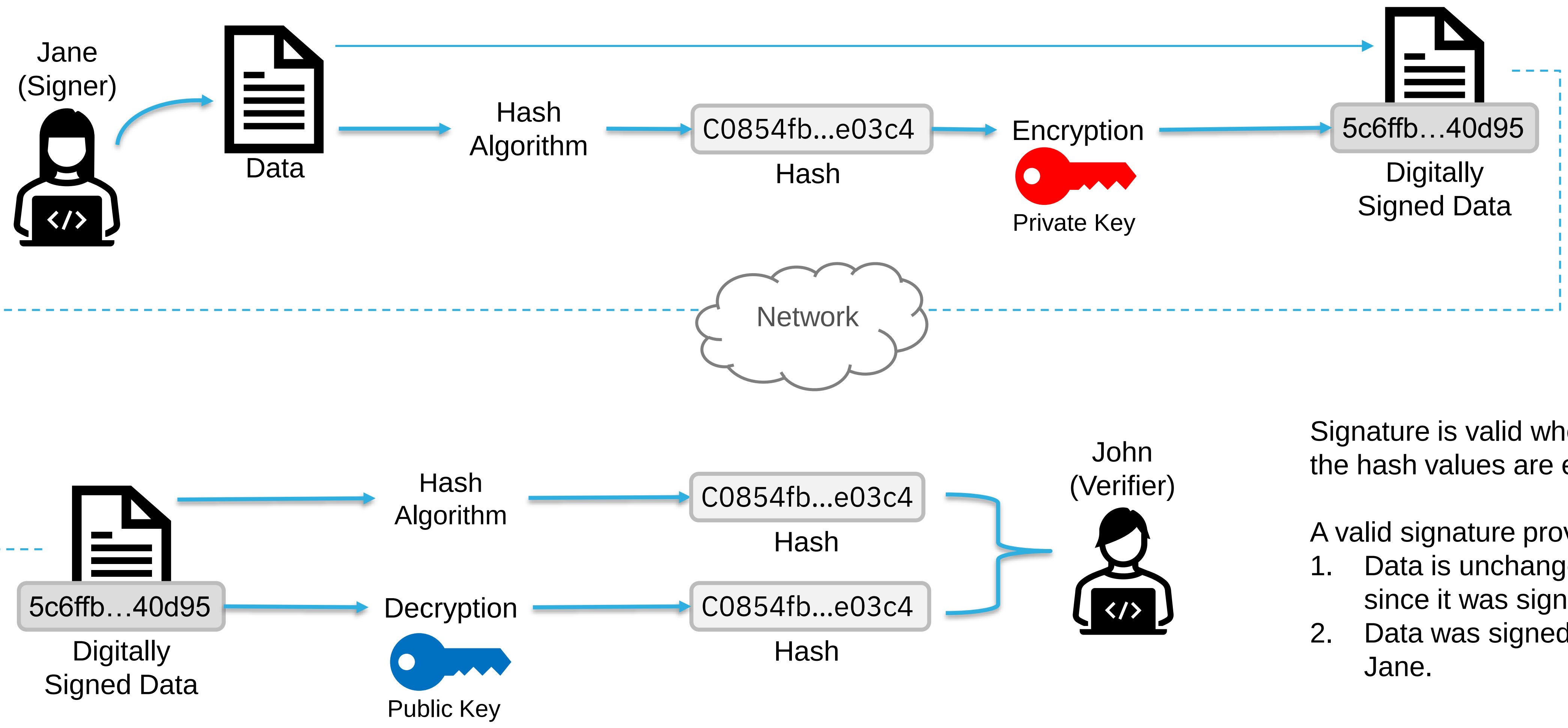
What is Public/Private Key Encryption?

- Encryption encodes data making it inaccessible to unauthorized users.
- Public/Private key pairs are very large mathematically related prime numbers.
- Data encrypted by one key can only be decrypted by the other.


“The quick brown fox jumped over the lazy dog”
Message
Encryption
Private Key
1e67955b2685133216340a2003202c43bcb382b01b900a673be59f9b8685fcc
cede813eef55d0e58fe44d38a0391c7e0ec216cd8b97bc7883ec0c2e5b9a6bf2
4f545730445e847d07ffb6c27d3b98b54e4b15914c5fef330167027f58c7cbd7e
a69bbbf415f2d2bdcb09c4fbefe4eac0deb92240db087b42859992e19567e7e3
Encrypted Message
“The quick brown fox jumped over the lazy dog”
Message
Decryption
Public Key

What is a Digital Signature?

A mathematical technique to verify the authenticity and integrity of digital data.



Signature is valid when the hash values are equal.

A valid signature proves:

1. Data is unchanged since it was signed
2. Data was signed by Jane.

What is a Digital Certificate?

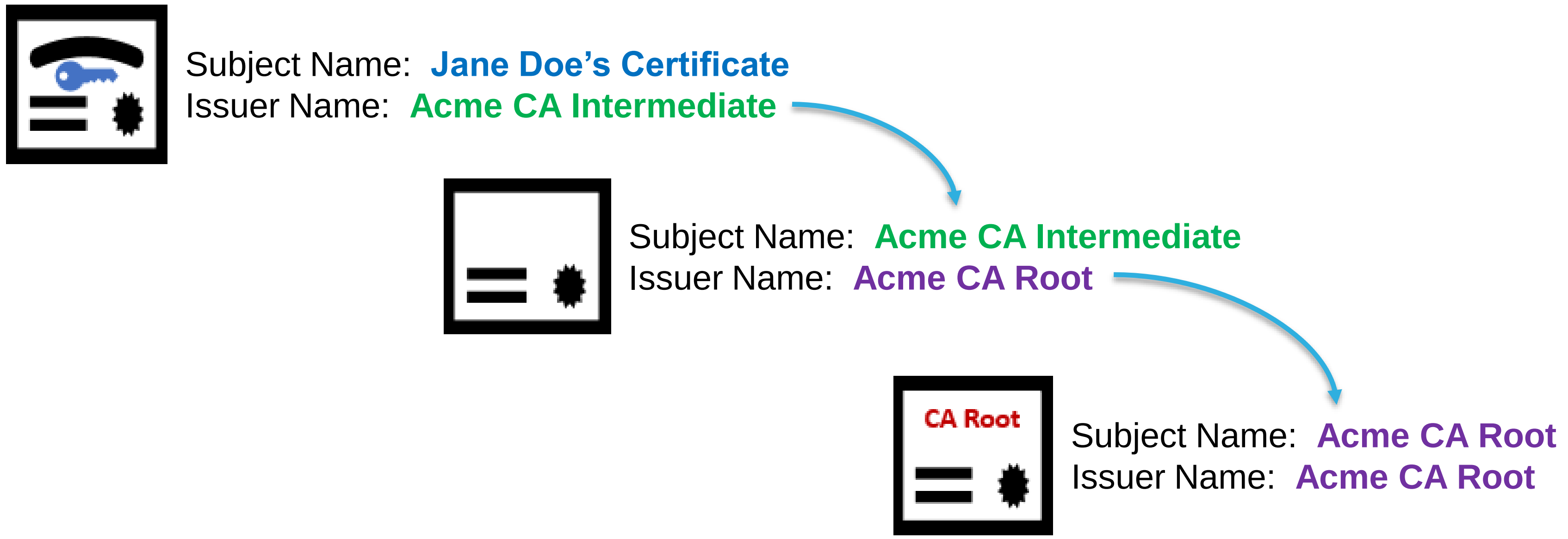
- File containing **identity** information and the **public key** for the certificate holder.
  - X.509 is the widely accepted standard for the file format.
- A certificate authority (CA) is a **trusted** entity that validates identity information and binds it to a public key in the form of a digital certificate.
- A digital certificate is “issued by” (or “signed by”) a certificate authority (CA).

Certificate

- **Subject Name**
- Issuer Name
- Not Before Date
- Not After Date
- **Public Key**

What is Certificate Path Validation?

- Procedure to ensure a certificate is **trusted** and valid for use.
- A certificate is trusted if it is issued by a trusted certificate authority (CA).


Subject Name: Jane Doe's Certificate
Issuer Name: Acme CA Intermediate
Subject Name: Acme CA Intermediate
Issuer Name: Acme CA Root
CA Root
Subject Name: Acme CA Root
Issuer Name: Acme CA Root

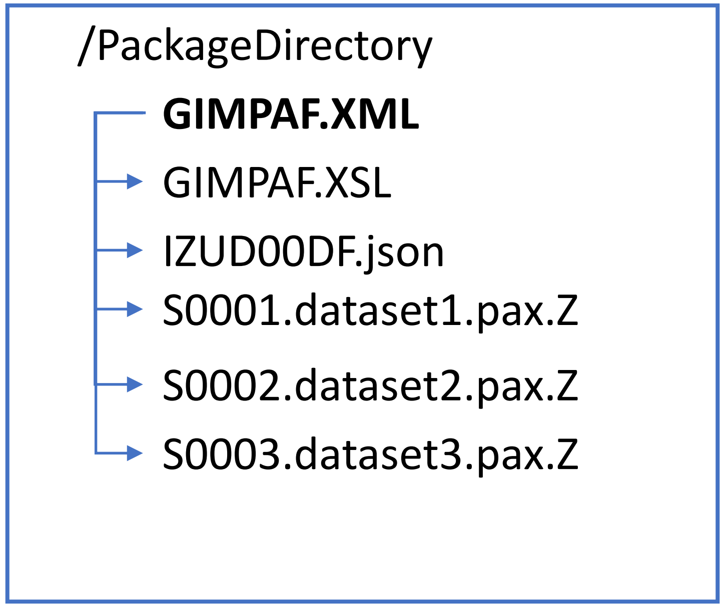
GIMZIP Package Signing Overview

GIMZIP Package Signing Overview

- GIMZIP package signing is implemented using **public/private key** technology
  - A **private key** is used to **calculate** digital signatures.
  - The corresponding **public key** is used to **verify** the signatures.
  - The public key is associated with an X.509 certificate, the “signing certificate”.
- The signing certificate is issued by a well known and trusted certificate authority
  - The certificate authority establishes the **authenticity** of the package signer (is the signer who they say they are?).
  - If the certificate authority is trusted, so then a signing certificate issued by that certificate authority can also be trusted.
- The signing certificate for the GIMZIP packages produced for IBM's z/OS product and service offerings is issued by an IBM z/OS certificate authority
  - **STG Code Signing Certificate Authority - G2**.
  - This CA certificate is built-in to RACF and other security managers.

GIMZIP Package Content

Unsigned GIMZIP package content:

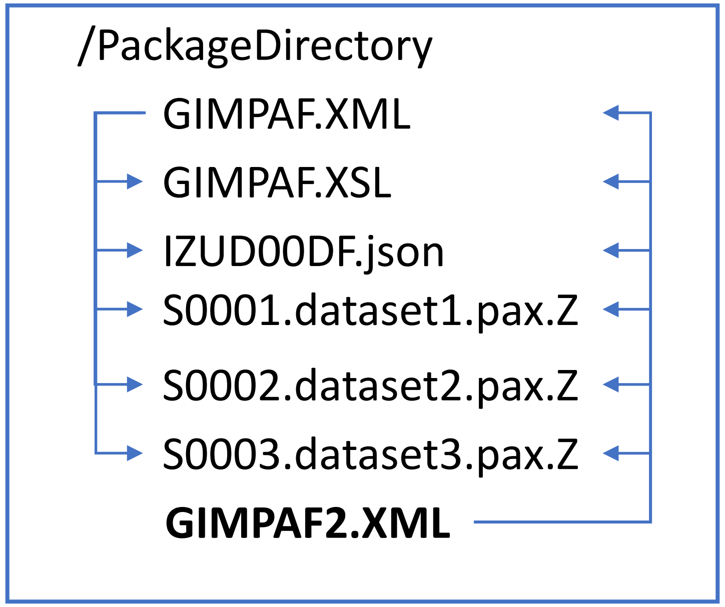


GIMPAF.XML file:

- Identifies all files in the package.
- Contains SHA-1 hash for each file.
- Contains SHA-1 hash for the package.

GIMZIP Package Content…

Signed GIMZIP package content:



GIMPAF.XML file (Unchanged):

- Identifies all files in the package. *
- Contains SHA-1 hash for each file.
- Contains SHA-1 hash for the package.

GIMPAF2.XML file:

- Identifies all files in the package.
- Contains SHA-256 hash for each file.
- Contains SHA256withRSA signature for the package.
- Contains certification path for the signing certificate, used for signature validation.

GIMZIP Package Content…

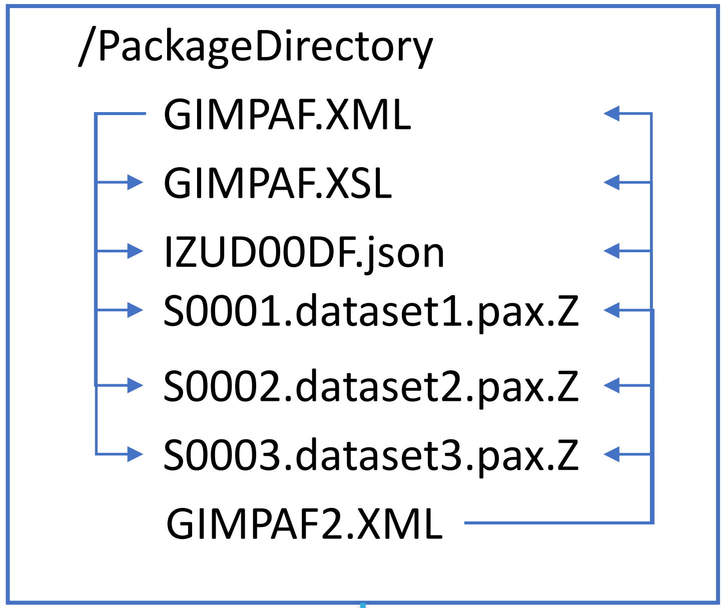
- SMP/E does **NOT** require signature verification for a signed GIMZIP package

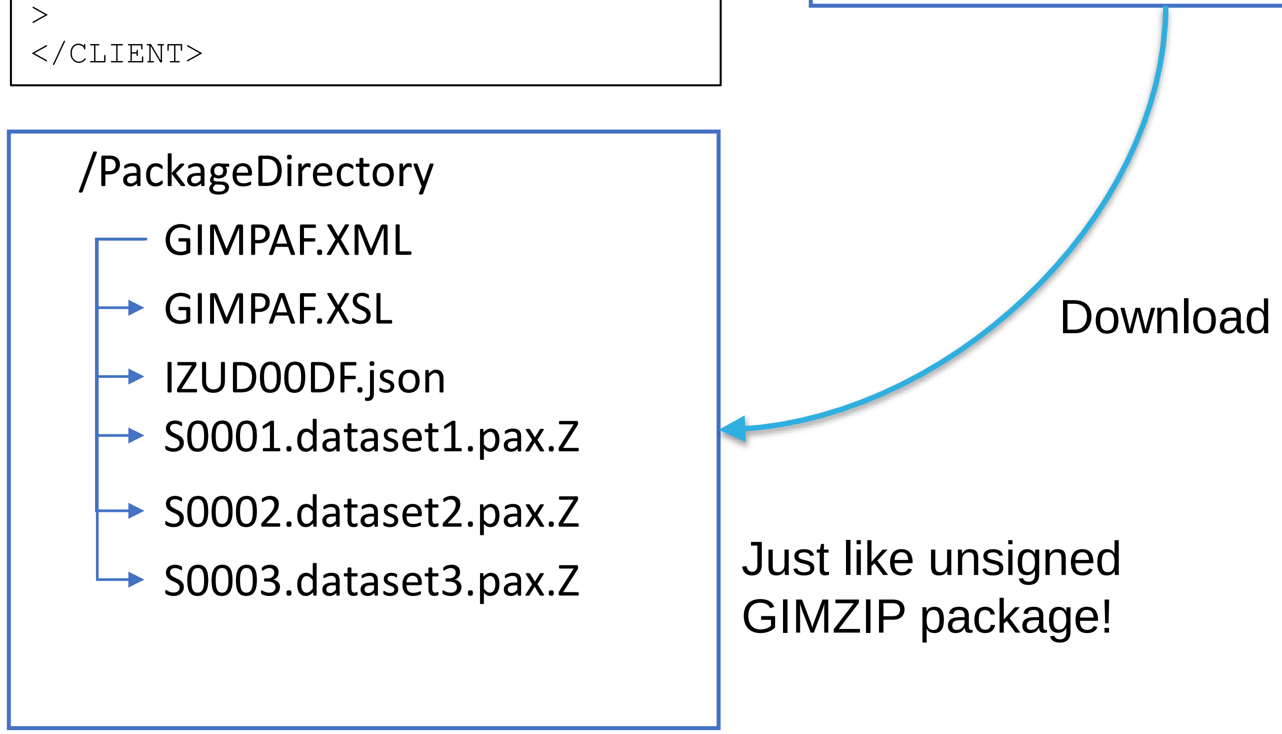
- Therefore, as a provider, you may sign GIMZIP packages whether or not consumers can or will verify the signatures.

Package Acquisition

No changes to RECEIVE input,
no signature verification.

```
<SERVER...
file="/orderdir/GIMPAF.XML"
hash="3A1B4C2D... " >
</SERVER>

<CLIENT...
>
</CLIENT>
```



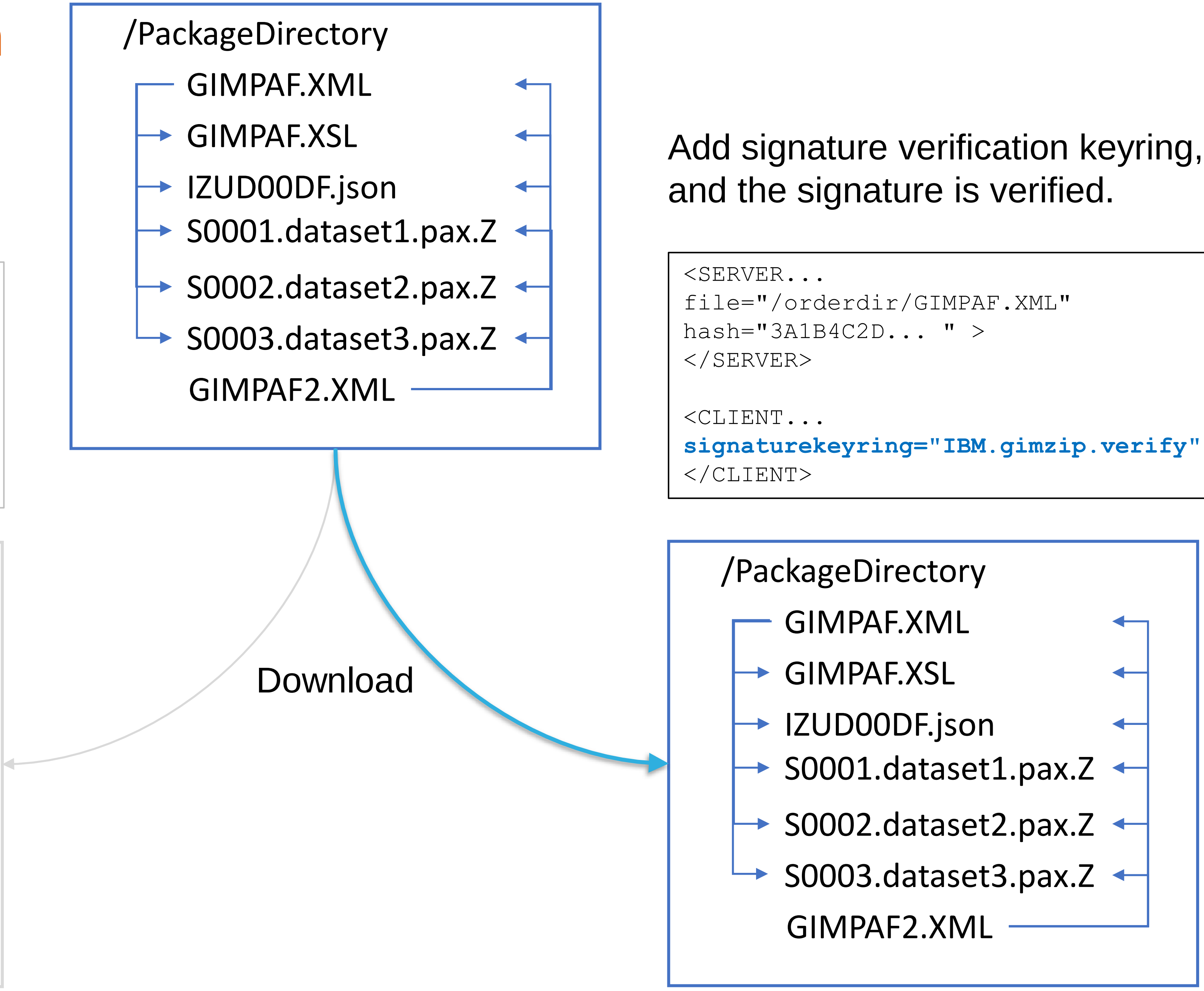


Just like unsigned
GIMZIP package!

Package Acquisition

Add signature verification keyring, and the signature is verified.

```
<SERVER...
file="/orderdir/GIMPAF.XML"
hash="3A1B4C2D... " >
</SERVER>

<CLIENT...
>
</CLIENT>
```



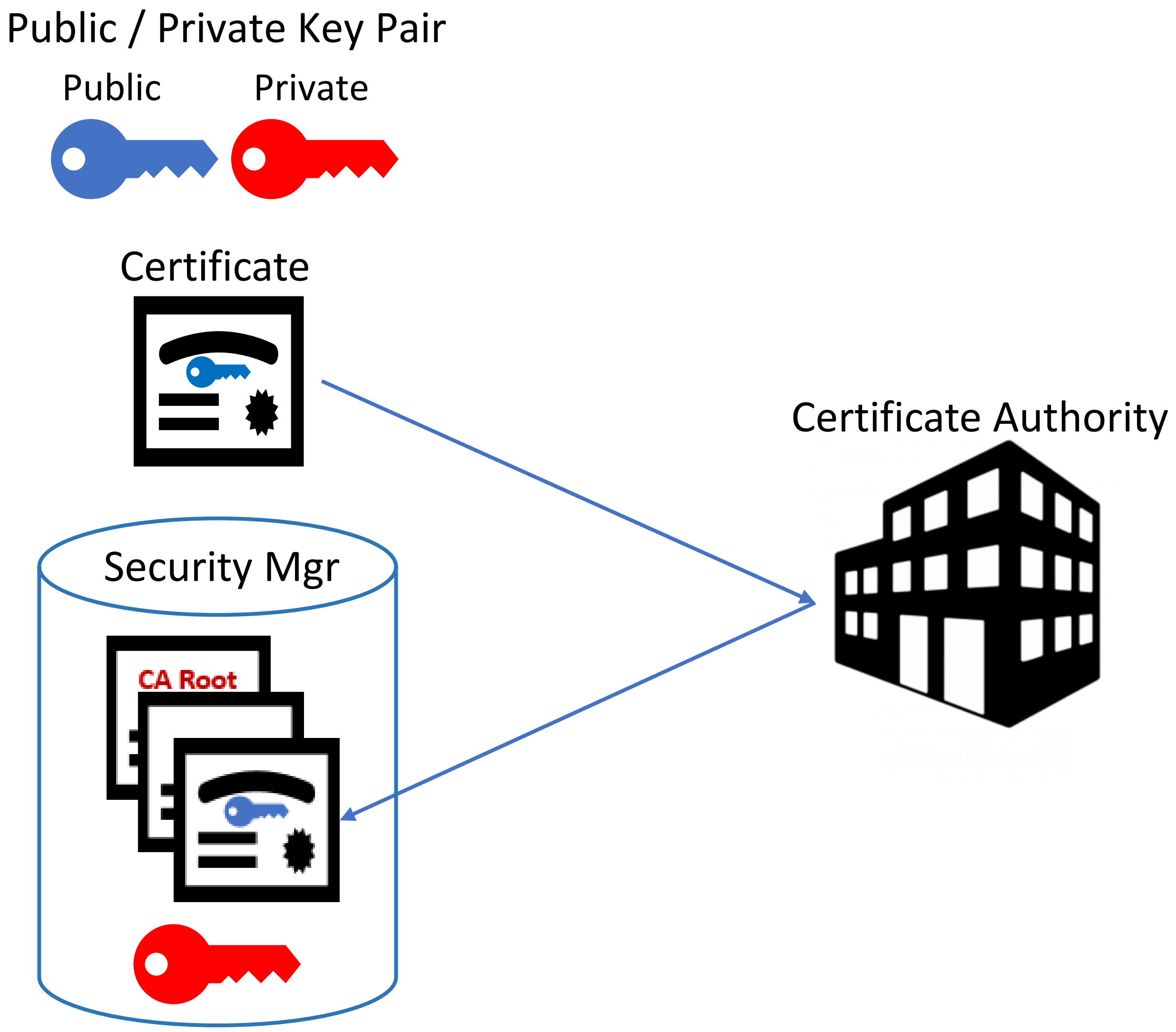
```
<SERVER...
file="/orderdir/GIMPAF.XML"
hash="3A1B4C2D... " >
</SERVER>

<CLIENT...
signaturekeyring="IBM.gimzip.verify" >
</CLIENT>
```

/PackageDirectory
GIMPAF.XML
GIMPAF.XSL
IZUD00DF.json
S0001.dataset1.pax.Z
S0002.dataset2.pax.Z
S0003.dataset3.pax.Z

Provider One-Time Setup

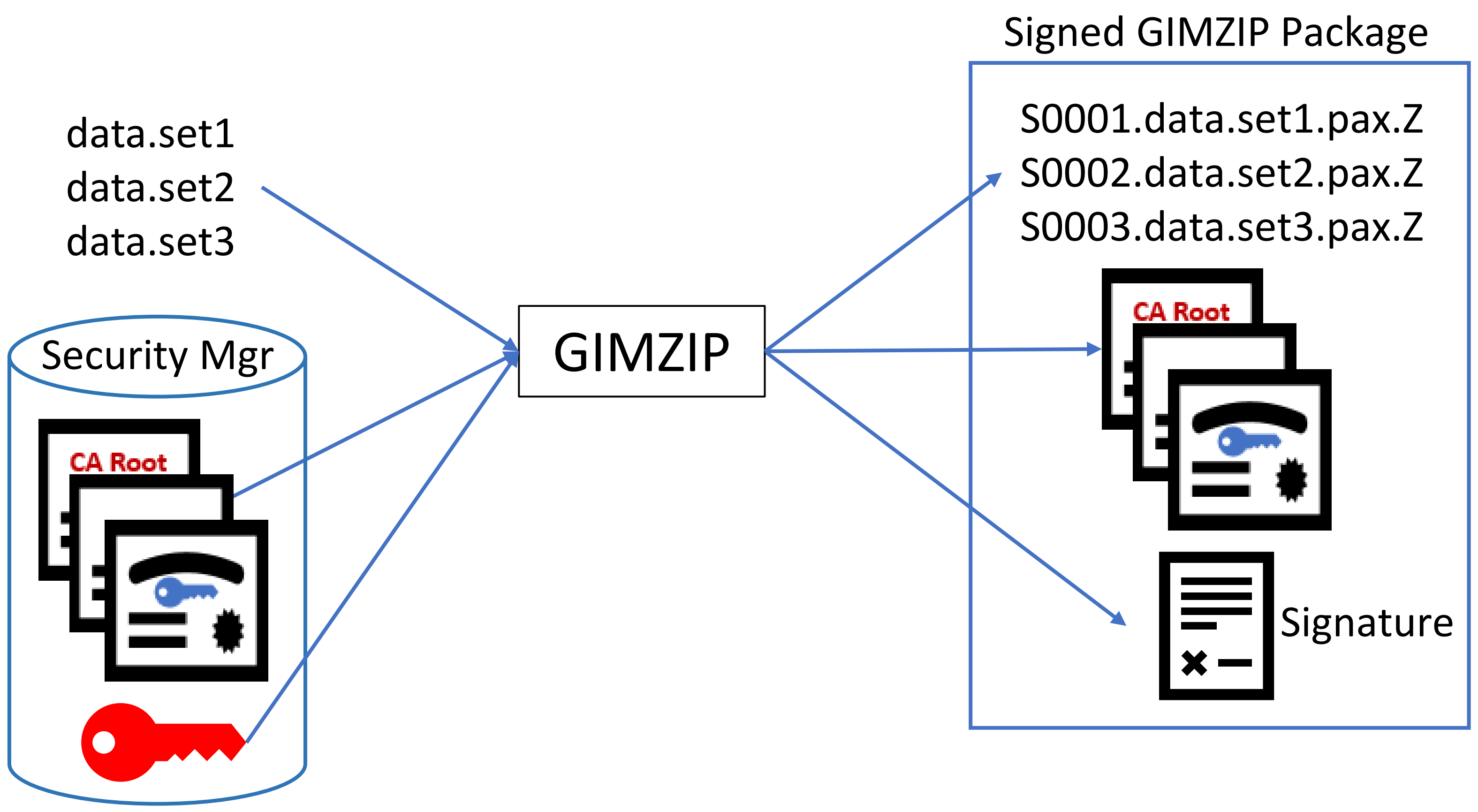
1. **Generate** a public/private key pair and certificate.
2. **Request** the certificate be signed by a Certificate Authority (Certificate Signing Request).
3. **Store** the signed certificate, its certification path, and private key in a SAF security manager* on z/OS.

* RACF or other SAF security manager



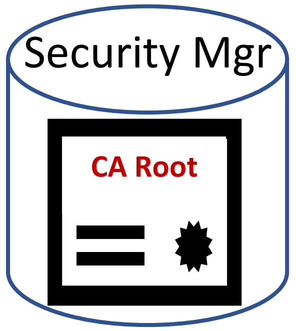
GIMZIP Signing Process

1. **Discover** and **Validate** the certification path.
2. **Create** archive files for each data set.
3. **Write** the certification path to the package.
4. **Sign** the package using the private key.



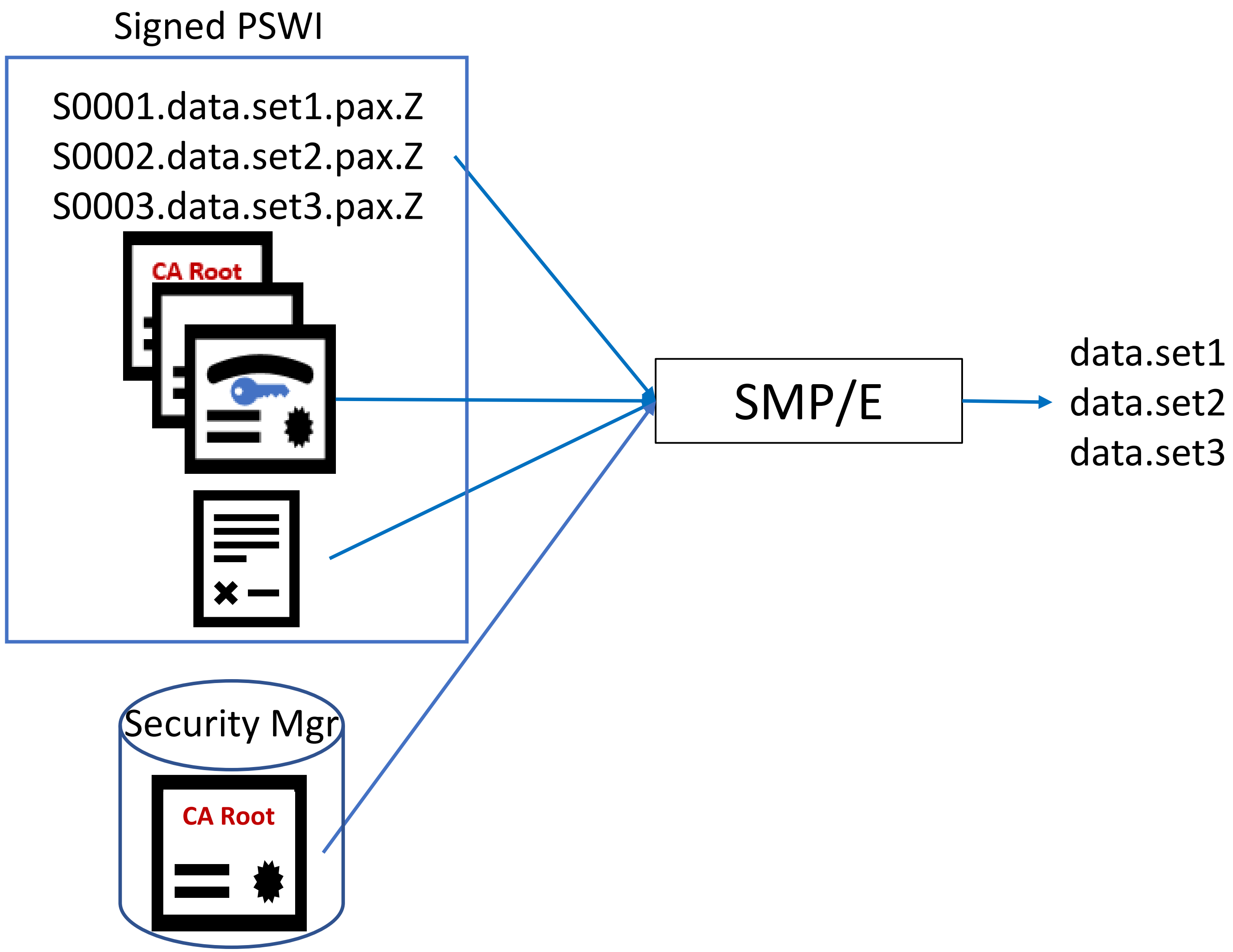
Consumer One-Time Setup

1. **Connect** the CA root certificate to a keyring in your SAF security manager.



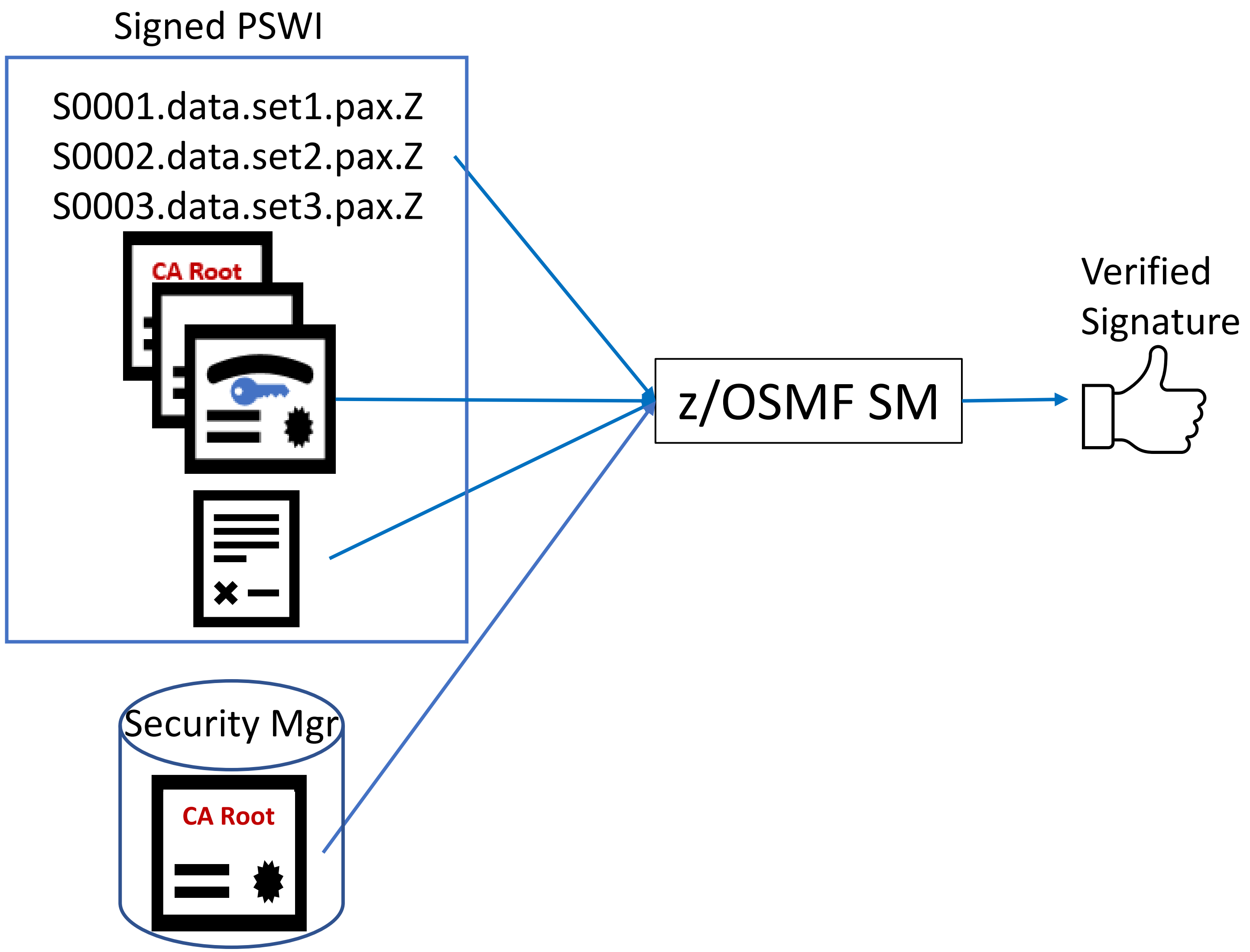
SMP/E Signature Verify Process

1. **Validate** certification path using the CA root certificate in the keyring.
2. **Verify** package signature using the public key.
3. **Create** data sets from archive files.



z/OSMF Software Management Signature Verify Process

1. **Validate** certification path using the CA root certificate in the keyring.
2. **Verify** package signature using the public key.
3. **Persist** verified signer information.



Details for a Provider

Calling GIMZIP

- Signing is optional
- New attributes in the input <GIMZIP> tag to specify:

1. Signing **certificate label**
2. SAF **keyring name** where the signing certificate and all certificates in its certification path are found.

```
<GIMZIP
  signingcertificate="Kurts Signing Cert"
  keyring="gimzip.signing.keyring"
  >
<FILEDEF name="/tmp/T1344212/IZUD00DF.json"
         archid="IZUD00DF.json" type="README"/>
<FILEDEF name="IBMUSR6.CICS.CBK.ACBKDWLD"
         archid="VSMPS3.IBMUSR6.CICS.CBK.ACBKDWLD"/>
...
</GIMZIP>
```

GIMPAF2.XML File

If signing is indicated, the GIMPAF2.XML file is created:

- One **<ARCHDEF>** for each file in the package.
- One **<X509Data>** for each cert in the certification path, from signing cert up to root CA.
- One **package signature**, for all <PKGDEF> data.

```
<PKGDEF ...>

<ARCHDEF name="filename" ...>
  <hash algorithm="SHA256">hash-value</hash>
</ARCHDEF>

<SignatureInfo>
  <SignatureAlgorithm>SHA256withRSA</SignatureAlgorithm>
  <SignerSubjectName>subject-name</SignerSubjectName>
  <CertPath>
    <X509Data>
      <X509SubjectName>subject-name</X509SubjectName>
      <X509Certificate>certificate-data</X509Certificate>
    </X509Data>
  </CertPath>
</SignatureInfo>

</PKGDEF>

<?PKGSIG signature="package-signature"?>
```

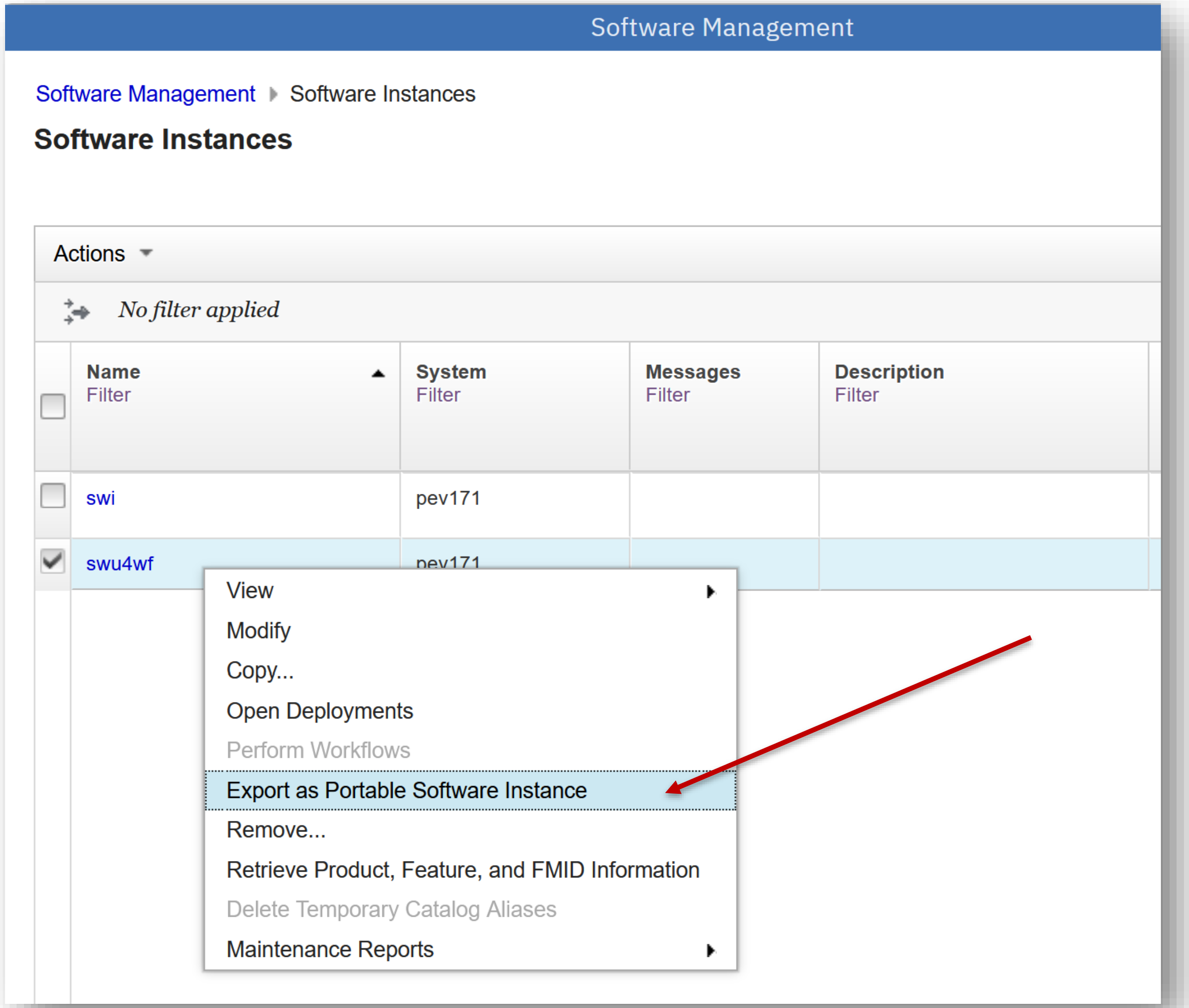
GIMPAF2.XML Example

```xml
<?xml version="1.0" ?>
<PKGDEF files="3" ... >
<ARCHDEF
name="S0001.SMPPTFIN.DATA.pax.Z"
originalsize="1703520"
size="96768">
<hash algorithm="SHA256">
E25E1F235D137EBFE4BD6B33B08C722AF973D2C8EB91D8D382737B4E77687480
</hash>
</ARCHDEF>

...

<SignatureInfo>
<SignatureAlgorithm>SHA256withRSA</SignatureAlgorithm>
<SignerSubjectName
CN=Kurts Package Signing Cert, O=IBM Systems Z, C=US
</SignerSubjectName>
<CertPath>
<X509Data>
<X509SubjectName>
CN=Kurts Signing Cert, O=IBM Systems Z, C=US
</X509SubjectName>
<X509Certificate>
-----BEGIN CERTIFICATE-----
MIIDxTCCAq2gAwIBAgIBATANBgkqhkiG9w0BAQsFADBEMQswCQYDVQQGEw
...
```

```xml
...
sjR9GJvZWm0x6zMRVeZhb5h4sT8aRPkwxncjjw==
-----END CERTIFICATE-----
</X509Certificate>
</X509Data>
</CertPath>
</SignatureInfo>

</PKGDEF>
<?PKGSIG signature="BFA69472F2C2BDAA950C6FB624DEF8F007C5082041B49A2742BF3172573
E609C24AEBB7A241A02FAEB18E96EAD0E4FECDB0238586D123682C0B315EC53FAAD9805224308B3
2775ACEBC1F4F784DF3FF7C528528FEB2588E8A0E649729CC7C9534626AF063D25218CD4F8FF9EE
208FA85796BBED516333904A641DD84187747FF76548B022BA9B9C23E086A68484A9949D4AD9716
613EC2F20CC9E81AECC24149B13D981D83C296D68D82F75E78B52777F30ACE043A0A4BDD17812D3
13A3AE162CFABE8602B2E20F390C3ADFCAC1889488D67F18CB5E4A6DA16ED0F8EC65674D2849B3A
F6A1FF8BDBA2880FF3EBA4B22332B257B040F07FFDD1198C7B56DE7E60"?>
```

z/OSMF Software Management, Export Action

Use the Software Instance **Export as Portable Software Instance** action to create a portable software instance.


Software Management

Software Management ▸ Software Instances

Software Instances

Actions ▾

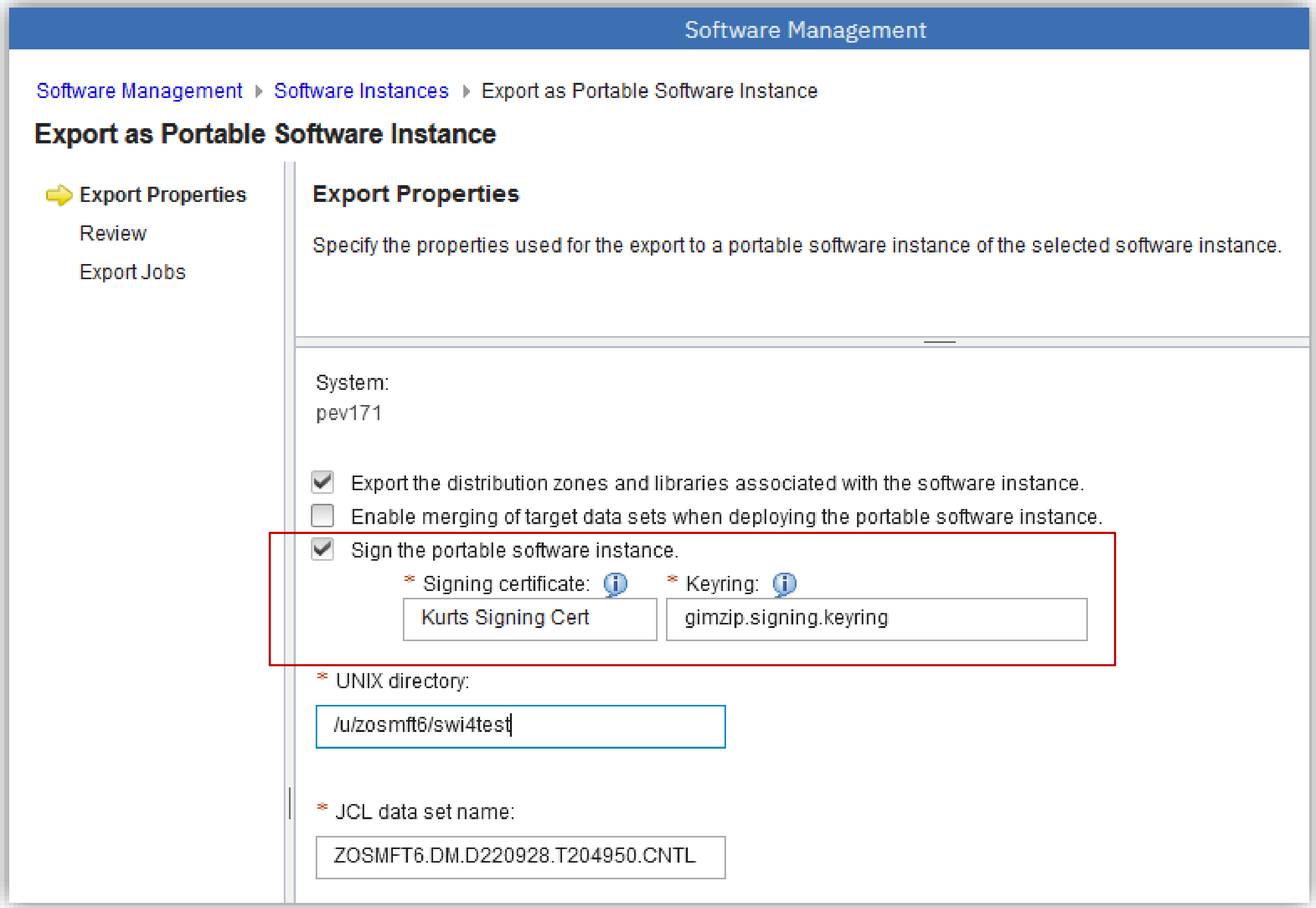
No filter applied

	Name Filter	System Filter	Messages Filter	Description Filter
☐	swi	pev171		
✓	swu4wf	pev171		

View
Modify
Copy...
Open Deployments
Perform Workflows
Export as Portable Software Instance
Remove...
Retrieve Product, Feature, and FMID Information
Delete Temporary Catalog Aliases
Maintenance Reports

z/OSMF Software Management, Export Action…

- New option for the **Export** action to sign the portable software instance
  - Provide the signing **certificate label** and the SAF **keyring**
- If the option is selected the generated Export JCL specifies the certificate and keyring for GIMZIP
- The Export REST API is also updated to accept input signing certificate and SAF keyring


Software Management
Software Management ▸ Software Instances ▸ Export as Portable Software Instance

Export as Portable Software Instance

Export Properties
Review
Export Jobs

Export Properties

Specify the properties used for the export to a portable software instance of the selected software instance.

System:
pev171

[✓] Export the distribution zones and libraries associated with the software instance.
[ ] Enable merging of target data sets when deploying the portable software instance.
[✓] Sign the portable software instance.

* Signing certificate: Kurts Signing Cert
* Keyring: gimzip.signing.keyring

* UNIX directory: /u/zosmft6/swi4test

* JCL data set name: ZOSMFT6.DM.D220928.T204950.CNTL

Certificate and Key Requirements

The signing certificate and public/private key pair must meet the following requirements:

1. Public/private key pair must be generated using the **RSA algorithm** and can be from 1024 to 4096 bits long.
2. Signing certificate must have the **Digital Signature key usage** certificate extension.
3. Signing certificate must not be expired.
4. Signing certificate should be issued by a well known and trusted certificate authority (CA), whose root certificate is easily obtained by your consumers.
5. Signing certificate, the issuing root CA certificate, and intermediate CA certificates if any, must be stored in the z/OS security manager database and connected to a keyring.

Certificate and Key Requirements…

This RACF command illustrates the key and certificate requirements.

```
RACDCERT GENCERT ID(cert-owner) +
  SUBJECTSDN(CN('My Package Signing Cert') +
             O('My Company') +
             C('US')) +
  RSA +
  SIZE(2048) +
  KEYUSAGE(HANDSHAKE) +
  NOTAFTER( DATE(2033-04-01) ) +
  SIGNWITH(CERTAUTH LABEL('My Root CA')) +
  WITHLABEL('My Package Signing Cert')
```

Certificate and Key Requirements…

Create a keyring and connect the signing certificate, issuing CA root certificate, and intermediate CA certificates if any.

```
RACDCERT ID(keyring-owner) ADDRING(keyringname)

RACDCERT ID(keyring-owner) +
  CONNECT(ID(cert-owner) +
  LABEL('My Package Signing Cert') +
  RING(keyringname) )

RACDCERT ID(keyring-owner) +
  CONNECT(CERTAUTH +
  LABEL('My Root CA’) +
  USAGE(CERTAUTH) +
  RING(keyringname) )
```

Certificate and Key Authorization

Identity for GIMZIP must be authorized to the keyring.

- If GIMZIP userid owns the certificate, then must have READ authority.

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) +
  ID(gimzip-userid) ACCESS(READ)
```

- If GIMZIP userid does NOT own the certificate, then must have UPDATE authority.

```
RDEFINE RDATALIB keyring-owner.keyring-name.LST UACC(NONE)
PERMIT keyring-owner.keyring-name.LST +
  CLASS(RDATALIB) ID(gimzip-userid) ACCESS(UPDATE)
SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
SETROPTS RACLIST(RDATALIB) REFRESH
```

- Detailed instructions for a provider:
  https://www.ibm.com/docs/en/zos/2.5.0?topic=routine-preparing-sign-gimzip-packages

Details for a Consumer

SMP/E RECEIVE

RECEIVE ORDER, RECEIVE FROMNET, GIMGTPKG

Package signature verification is optional.

1. A provider can sign packages, but **supply unchanged <SERVER> XML** to consumers (file = GIMPAF.XML and SHA-1 hash)
2. Consumers can continue to download packages with existing levels of SMP/E

- Signatures will not be verified

```
...
//SMPSRVR  DD *
<SERVER
  host="download.server.com"
  user="S679p074"
  pw="k09944D4604223r">
  <PACKAGE
    file="/2022102123341/PROD/GIMPAF.XML"
    hash="3A14791D9F3DAA8D3DB25499538EEFBCAB5467F8"
    id="21October2022">
  </PACKAGE>
</SERVER>
/*
//SMPCLNT DD *
<CLIENT
  javahome="/usr/lpp/java/J8.0_64"
  downloadmethod="https"
  downloadkeyring="*AUTH*/*"
  >
</CLIENT>
/*
```

SMP/E RECEIVE…

RECEIVE ORDER, RECEIVE FROMNET, GIMGTPKG

Package signature verification is optional.

- If signature verification is desired, specify new attribute in **<CLIENT> XML** to identify SAF keyring name for the root certificate
- If the GIMPAF2.XML file resides on the server, it is downloaded and the signature verified
- If the GIMPAF2.XML file does not reside on the server, processing will continue for the unsigned package

```
//SMPSRVR  DD *
<SERVER
  host="download.server.com"
  user="S679p074"
  pw="k09944D4604223r">
  <PACKAGE
    file="/2022102123341/PROD/GIMPAF.XML"
    hash="3A14791D9F3DAA8D3DB25499538EEFBCAB5467F8"
    id="21October2022">
  </PACKAGE>
</SERVER>
/*
//SMPCLNT DD *
<CLIENT
  javahome="/usr/lpp/java/J8.0_64"
  downloadmethod="https"
  downloadkeyring="*AUTH*/*"
  signaturekeyring="IBM.package.sig.verification"
  >
</CLIENT>
/*
```

Calling GIMUNZIP

GIMUNZIP

Package signature verification is optional.

- If signature verification is desired, specify new **EXEC parameter** and attribute in **<CLIENT> XML** to identify SAF keyring name for the root certificate
- SMP/E and GIMUNZIP write a signature information message

```
//UNZIP   EXEC PGM=GIMUNZIP,PARM='VERIFYSIG=YES'
...
//SMPCLNT DD *
<CLIENT
  javahome="/usr/lpp/java/J8.0_64"
  signaturekeyring="IBM.package.sig.verification"
  >
</CLIENT>
/*
```

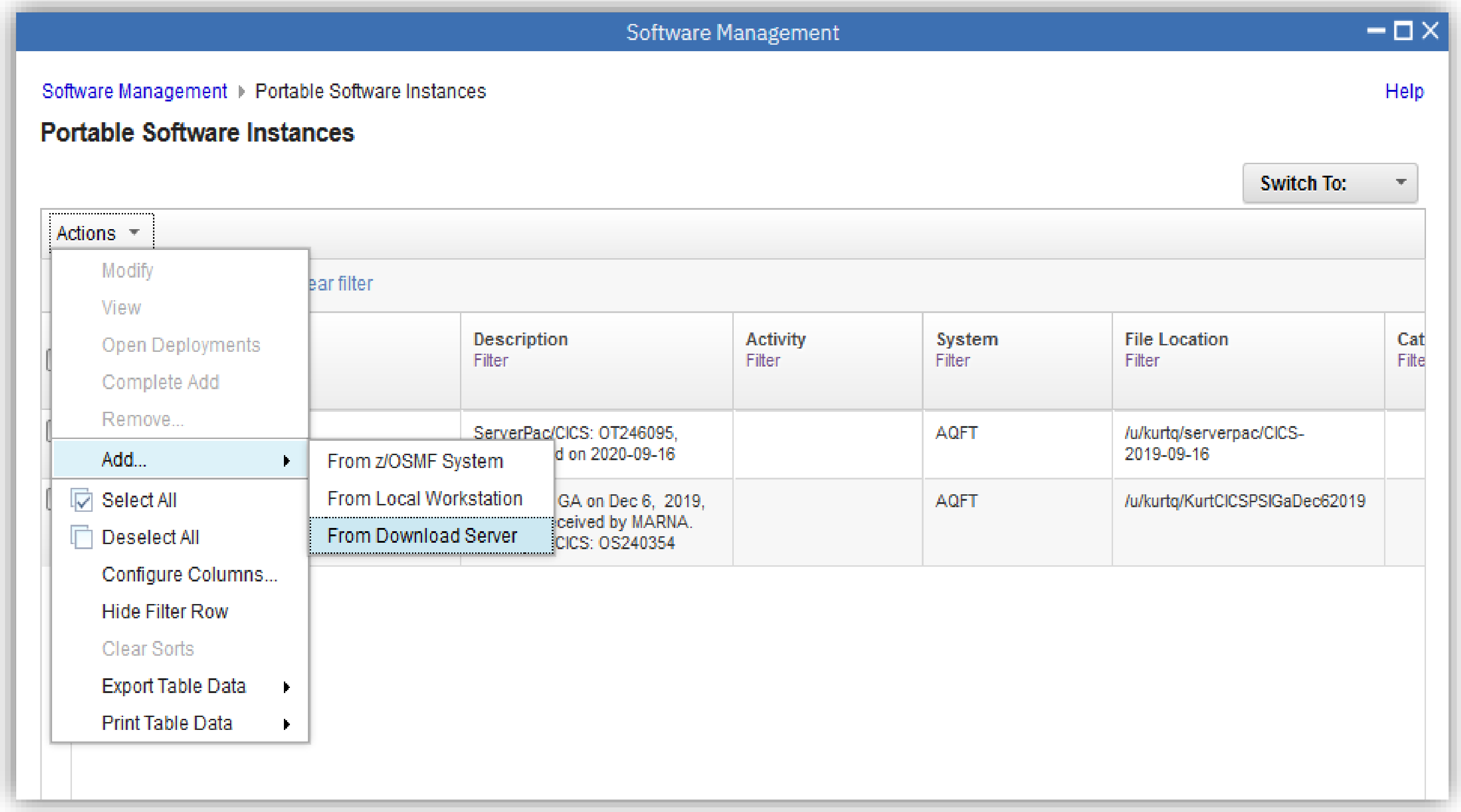
```
GIM69270I    SIGNATURE VALIDATION FOR FILE "/u/ibmusr6/smpnts/test/GIMPAF2.XML"
             WAS SUCCESSFUL. THE GIMZIP PACKAGE WAS SIGNED BY A CERTIFICATE WITH
             SUBJECT NAME "CN=Kurts Package Signing Cert, O=IBM System Z, C=US",
             SERIAL NUMBER "1" AND SHA256 FINGERPRINT
             "4aa0fc6708314ca95fc2699bad116158298808c089f43e1ed4600eb4170916f4".
             THE SIGNING CERTIFICATE WAS ISSUED BY "CN=Kurts Root CA, O=IBM
             System Z, C=US".
```

z/OSMF Software Management, Add Action

Portable Software Instance Add Action

Three Portable Software Instance **Add** actions:

1. From z/OS System
2. From Local Workstation
3. From Download Server

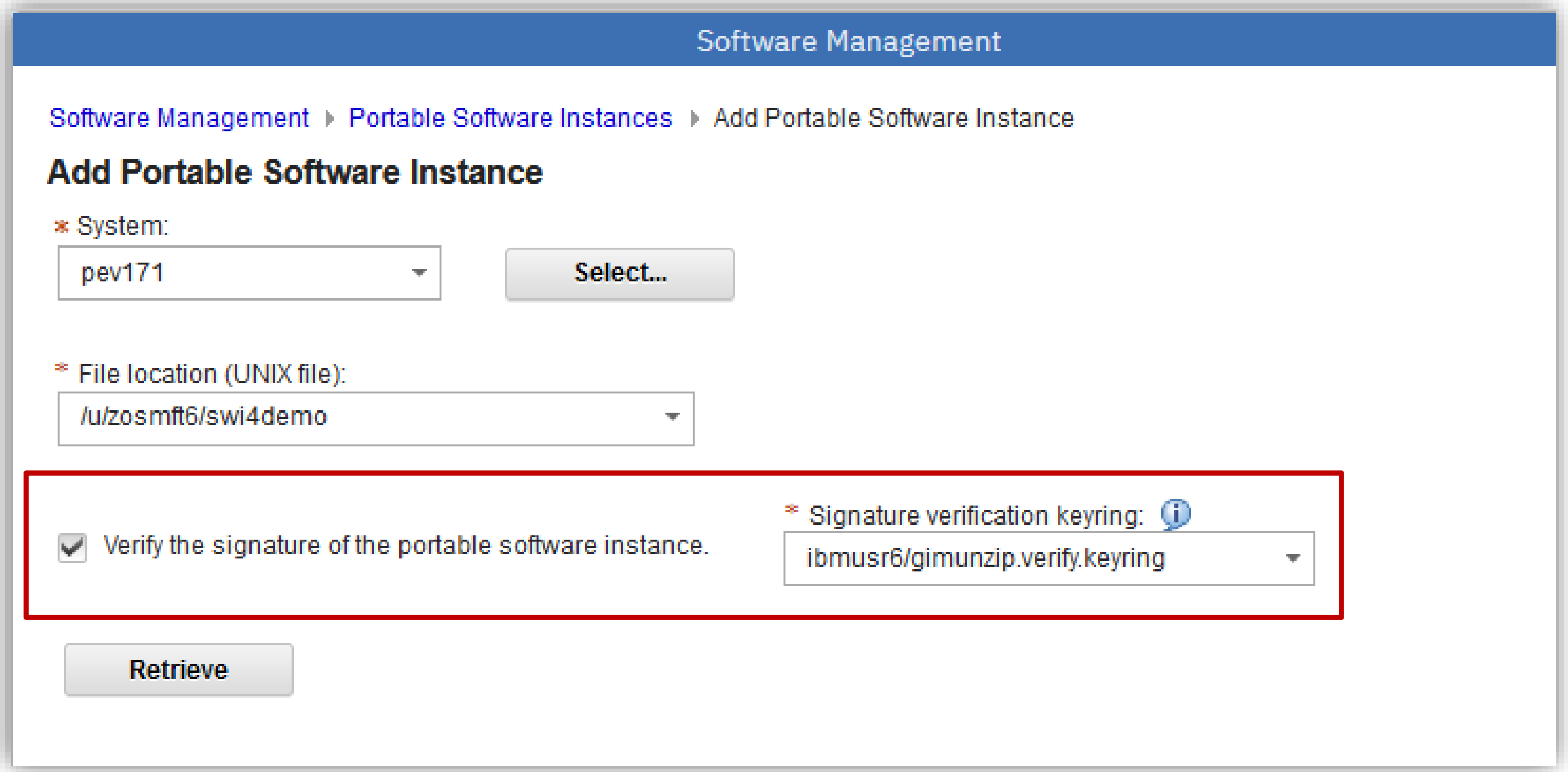


Description	Activity	System	File Location	Cat
Filter	Filter	Filter	Filter	Filte
ServerPac/CICS: OT246095, d on 2020-09-16		AQFT	/u/kurtq/serverpac/CICS-2019-09-16	
GA on Dec 6, 2019, ceived by MARNA. CICS: OS240354		AQFT	/u/kurtq/KurtCICSPSIGaDec62019	

z/OSMF Software Management, Add Action…

Portable Software Instance Add Action

- All 3 Add actions offer a new option to **verify the signature** for a portable software instance
- Specify the signature verification SAF keyring
- If the option is chosen the signature is verified for the portable software instance


Software Management
Software Management ▸ Portable Software Instances ▸ Add Portable Software Instance
Select...
Retrieve

Add Portable Software Instance

* System: pev171

* File location (UNIX file): /u/zosmft6/swi4demo

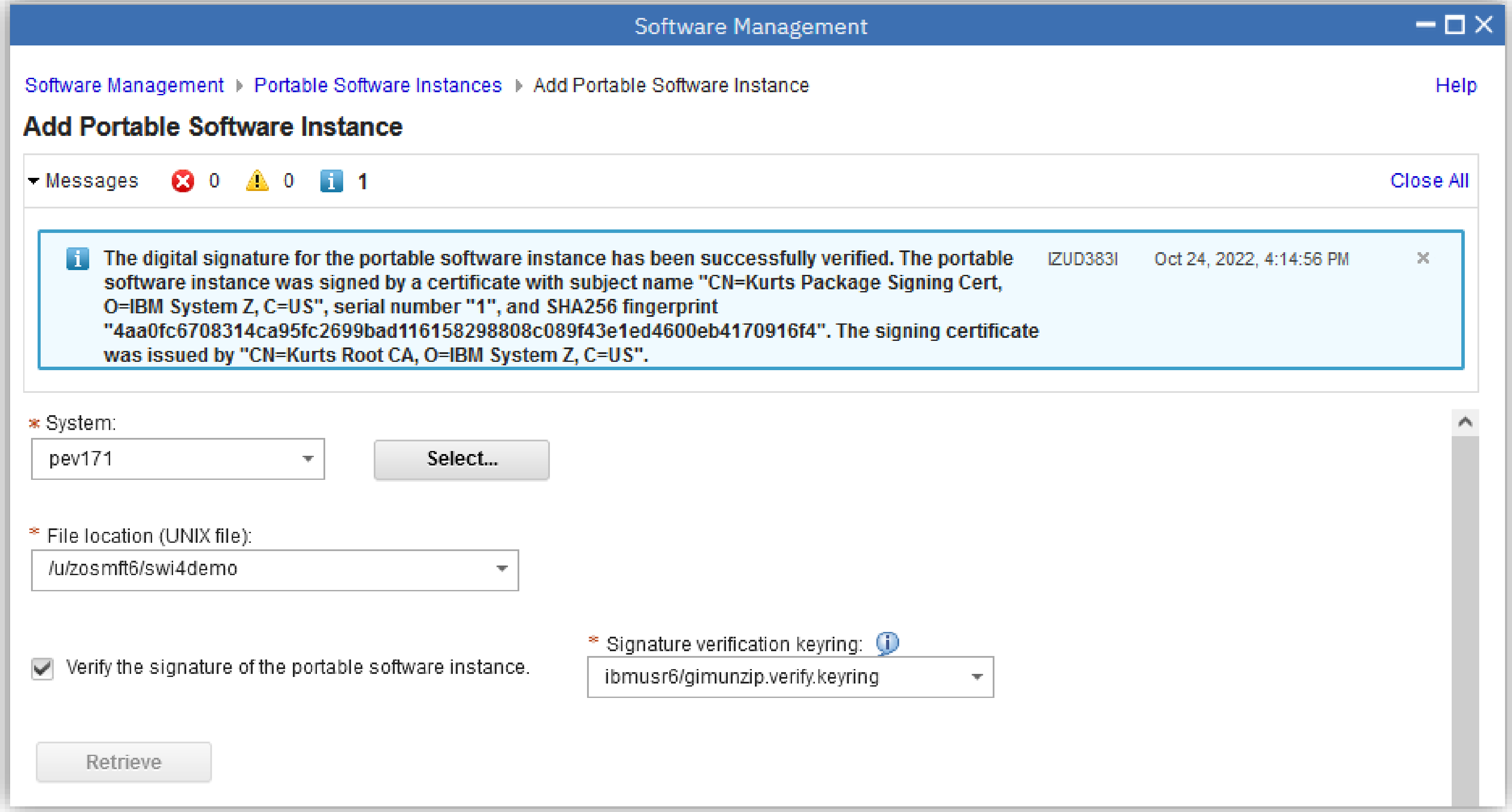
✓ Verify the signature of the portable software instance.

* Signature verification keyring: ibmusr6/gimunzip.verify.keyring

z/OSMF Software Management, Add Action…

Portable Software Instance Add Action

If the PSWI is signed, and if signature is verified, then the signer information is displayed.


Software Management
Software Management ▸ Portable Software Instances ▸ Add Portable Software Instance
Help
Add Portable Software Instance
Messages 0 0 1
Close All

The digital signature for the portable software instance has been successfully verified. The portable software instance was signed by a certificate with subject name "CN=Kurts Package Signing Cert, O=IBM System Z, C=US", serial number "1", and SHA256 fingerprint "4aa0fc6708314ca95fc2699bad116158298808c089f43e1ed4600eb4170916f4". The signing certificate was issued by "CN=Kurts Root CA, O=IBM System Z, C=US". IZUD383I Oct 24, 2022, 4:14:56 PM

* System: pev171

Select...

* File location (UNIX file): /u/zosmft6/swi4demo

✓ Verify the signature of the portable software instance.

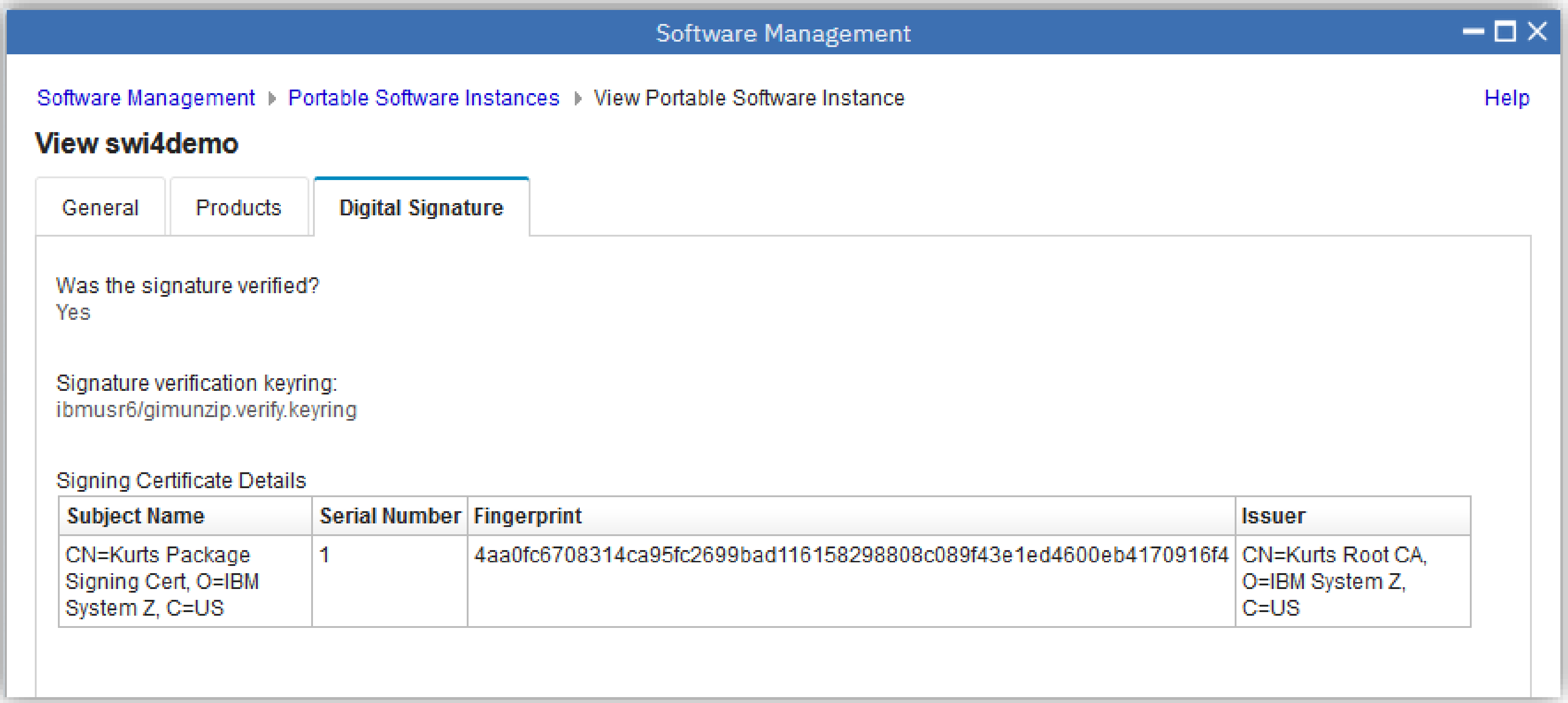
* Signature verification keyring: ibmusr6/gimunzip.verify.keyring

Retrieve

z/OSMF Software Management, Add Action…

Portable Software Instance View Action

If the PSWI is signed, and if the signature is verified, then the signer information is persisted and displayed on a new tab on the Portable Software Instance View page


Software Management
Software Management ▸ Portable Software Instances ▸ View Portable Software Instance
Help
General
Products
Digital Signature

View swi4demo

Was the signature verified?
Yes

Signature verification keyring:
ibmusr6/gimunzip.verify.keyring

Signing Certificate Details

Subject Name	Serial Number	Fingerprint	Issuer
CN=Kurts Package Signing Cert, O=IBM System Z, C=US	1	4aa0fc6708314ca95fc2699bad116158298808c089f43e1ed4600eb4170916f4	CN=Kurts Root CA, O=IBM System Z, C=US

CA Root Certificate and Authorization

- Detailed instructions:
  https://www.ibm.com/docs/en/zos/2.5.0?topic=guide-preparing-verify-signatures-gimzip-packages

- IBM certificate authority root is "**STG Code Signing CA – G2**"
  - Automatically supplied with RACF and other security managers
  - If not currently in your RACF db, RACF initialization will add it during next IPL

- Create a keyring containing the IBM CA root:

```
RACDCERT ID(userid) ADDRING(IBM.package.sig.verification)
RACDCERT ID(userid) CONNECT(CERTAUTH +
  LABEL('STG Code Signing CA - G2') +
  RING(IBM.package.sig.verification) +
  USAGE(CERTAUTH) )
```

Certificate and Key Authorization

User identity under which SMP/E RECEIVE, GIMGTPKG, and GIMUNZIP runs, and the logged-in z/OSMF userid, must be authorized to the specified keyring.

Must have READ authority to **either** of the following:

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) +
  ID(smpe-userid) ACCESS(READ)
```

```
RDEFINE RDATALIB keyring-owner.keyring-name.LST UACC(NONE)
PERMIT keyring-owner.keyring-name.LST +
  CLASS(RDATALIB) ID(smpe-userid) ACCESS(READ)
SETROPTS RACLIST(RDATALIB) CLASSACT(RDATALIB)
SETROPTS RACLIST(RDATALIB) REFRESH
```

smpe-userid is the user identity running the SMP/E job or the logged-in z/OSMF userid.

SMP/E and z/OSMF Software Management Availability

- Package signing and Signature verification is integrated into z/OS 3.1.
- PTFs for the following APARs are required for z/OS 2.5 and 2.4:
  - SMP/E – IO28360
  - z/OSMF – PH49385

IBM Exploitation of Package Signing

- As of **May 16, 2023**, IBM is exploiting GIMZIP package signing for all z/OS software **product** deliverables:
  - z/OSMF Portable Software Instances (ServerPac)
  - CBPDO

- IBM plans to exploit GIMZIP package signing for z/OS software **service** deliverables later in 2023:
  - Shopz PTF orders
  - SMP/E RECEIVE ORDER PTF and HOLDDATA orders

Summary

- Digital Signature Background
- GIMZIP Package Signing Overview
- Details for a Provider
  - How to specify a signing certificate
  - Certificate requirements and authorization
- Details for a Consumer
  - Create a keyring and connect the CA root
  - Keyring authorization