z/OS Validated Boot

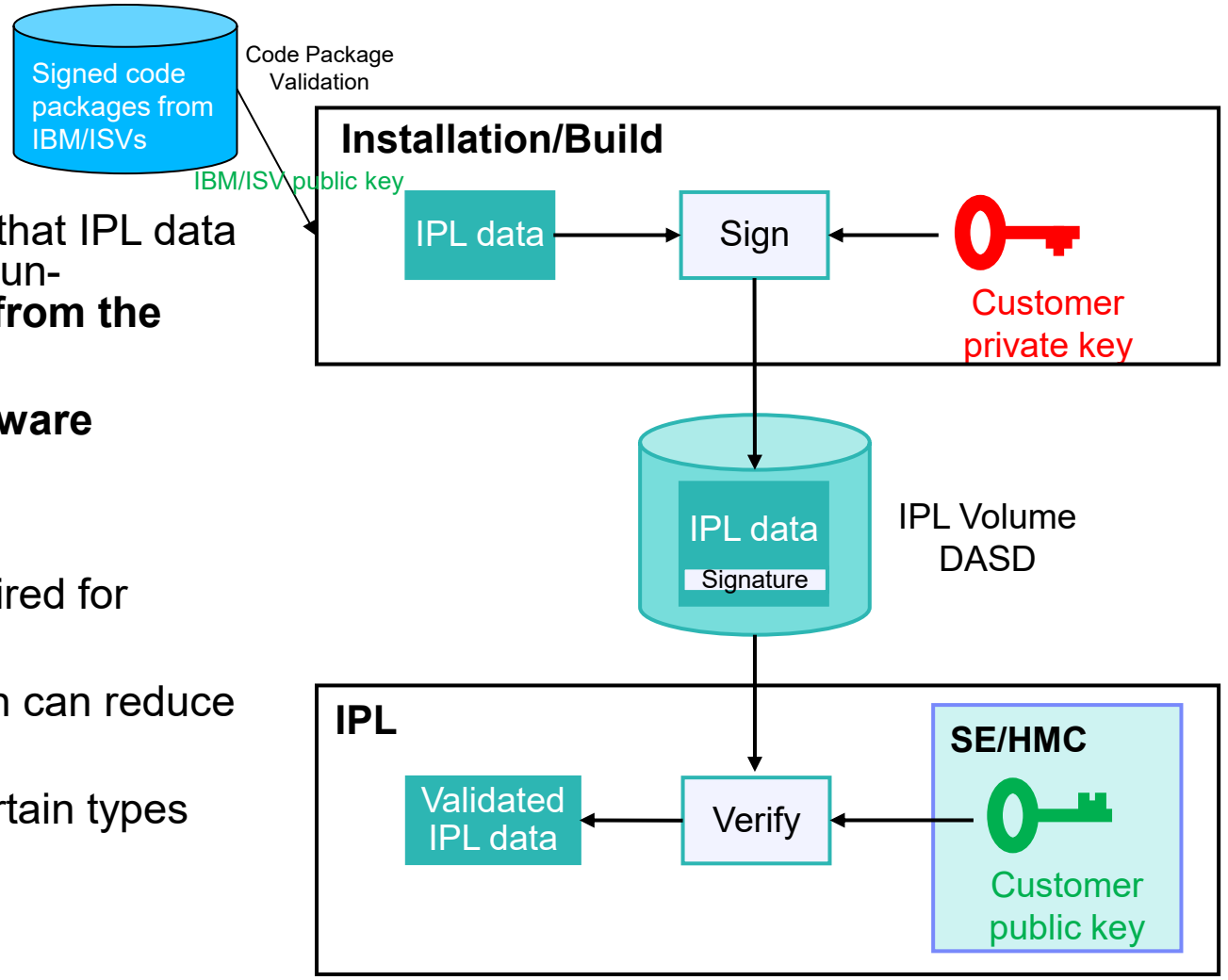
Dave Surman
IBM Poughkeepsie – z/OS Development
surman@us.ibm.com

SHARE New Orleans
Session 10480
August, 2024

Z/OS VALIDATED BOOT SOLUTION OVERVIEW

z/OS Validated Boot

- **What is z/OS Validated Boot?**
- Uses digital signatures to provide an IPL-time check that IPL data (ie. executables residing on an IPL volume) is intact, un-tampered-with, and originates from a trusted source **from the time it was built and signed**
- Enables detection of **unauthorized changes to software executables**
- **What value does it provide?**
- Ability to meet regulatory compliance standards required for certain secure software deployment scenarios
- Early detection of *accidental* IPL data changes, which can reduce impact and scope of outages
- Detection of *malicious* IPL data changes can stop certain types of attacks



- **Note:**
  - "Validated Boot" = "Secure Boot" = "Boot Integrity Validation"
  - "Boot" = "IPL" **(NOT firmware IML)**
  - Note that Validated Boot is NOT the same thing as "Secure Execution"

Related Solutions…

- **GIMZIP package signing and verification in z/OS SMP/E and z/OSMF Software Management**
  - Ensures the integrity of signed code packages delivered to customers from IBM and other software vendors

- **Validated Boot for z/OS**
  - Ensures the integrity of built z/OS system executables (load modules), once the above code packages have been validated and "unpacked" and the contents have been used to build signed, IPLable z/OS systems

- **Secure Boot for ECKD Devices**
  - Provides firmware support for secure List-Directed IPL (LD-IPL) from ECKD storage devices in addition to SCSI devices, for both z/OS and Linux use

See z16 GA1.5 and z/OS Product Announcement verbiage in the presentation backup section

Design Approach: Load Module Signing and Validation, with Firmware Support

- **Code Package Validation validates the incoming signed code package deliverables using software vendor's public key**
- **Client's "secure build" process creates and signs the *z/OS IPL Text and system load modules* with the client's private key, and stores them on disk**
- **Client's public keys for validation purposes are provided to and maintained in the platform firmware and made available to partitions for IPL-time validation use**
- **At IPL time, platform firmware (Z Bootloader) validates the IPL Text using the client's public keys; the IPL Text contains validation support which z/OS uses in subsequent load module validation steps during the IPL**
- **As z/OS loads subsequent authorized load modules during IPL, it validates their signatures using the client's public keys**
- **Validation failures during IPL may result in non-restartable wait state termination of the IPL, or not, depending on the requested IPL validation mode**

➢ ***Build up a chain of trust through digital signature validation, at every step of this cascading process, anchored in the firmware validation of the IPL Text and the secure firmware repository for the validation certificates/keys***

NIAP Certification and Regulatory Compliance

- **We are targeting Z and z/OS NIAP Certification with OS Protection Profile (OSPP) 4.3, which now requires both code package signing/validation and Boot Integrity Validation for IPLed "kernel" software**
  - Requirements for code package signing/validation and for Validated Boot are relatively new and are now incorporated into the NIAP profile; earlier certifications of z/OS and Z hardware/firmware *do not implement this requirement*
  - This higher level of certification may be required in some client environments, based on industry regulations and/or other security requirements
  - Boot Integrity Validation is an important part of securing the "supply chain" for system software from the software vendors, through the build process, to time of use

- ***We provided the basic validation capabilities needed to achieve NIAP certification first, then potentially we will provide additional value-add validation capabilities in future stages of z/OS Validated Boot support…***

Support Requirements – Overall Solution

- **Hardware/Firmware Support**
  - CPACF digital signature support with Elliptic Curve ECDSA-P521 support and SHA-512 hashing support
  - Virtual Flash Memory (VFM), also known as Storage Class Memory (SCM), for use in z/OS paging for LPA pages
  - z16 GA1.5 firmware (5/2023) provides:
    - **Support for List-Directed IPL (LD-IPL) from ECKD DASD (in addition to SCSI DASD)**
      - Via SE/HMC and DPM new load panel/load profile options – SCSI vs ECKD, CCW-IPL vs LD-IPL, Enforce vs Audit security mode
        - **For Linux on Z** – supports Linux IPL from ECKD DASD, not just SCSI DASD like today
        - **For z/OS** – supports z/OS Validated Boot from ECKD DASD
      - Current CCW-IPL (non-validated IPL) capabilities are preserved for migration, compatibility, and fallback, from same IPL Volume
    - **Certificate Store for Validated Boot**
      - Via SE/HMC and DPM Certificate import and Certificate management, including mapping imported certificates to specific LPARs for IPL-time validation use
      - PR/SM provides support for the Certificate Store on a per-LPAR basis
        - **For Linux on Z** - provides value for key management of Linux distributor validation keys without needing to deliver those Linux distributor keys in IBM firmware as we do today, simplifying distributor key rotation etc.
        - **For z/OS** – provides the trusted validation keys for z/OS Validated Boot

- **z/OS Software Support**
  - z/OS 2.5 post-GA support, delivered via z/OS CD APARs (5/2023); also 3.1 base
  - ServerPac installation workflow support

Support Details – Overall Solution

- **z/OS APARs**
  - ICKDSF – PH45198
  - Binder – OA63323
  - Signing Utility – OA63377
  - Supervisor – OA62783/OA63507
  - ASM/VSM – OA63420
  - RACF – OA61878
  - SAF – OA61901
  - SADMP – OA63404
  - IOS/Loadwait – OA63392
  - BCPii – OA63488

- **ServerPac z/OS Installation Workflow Support**

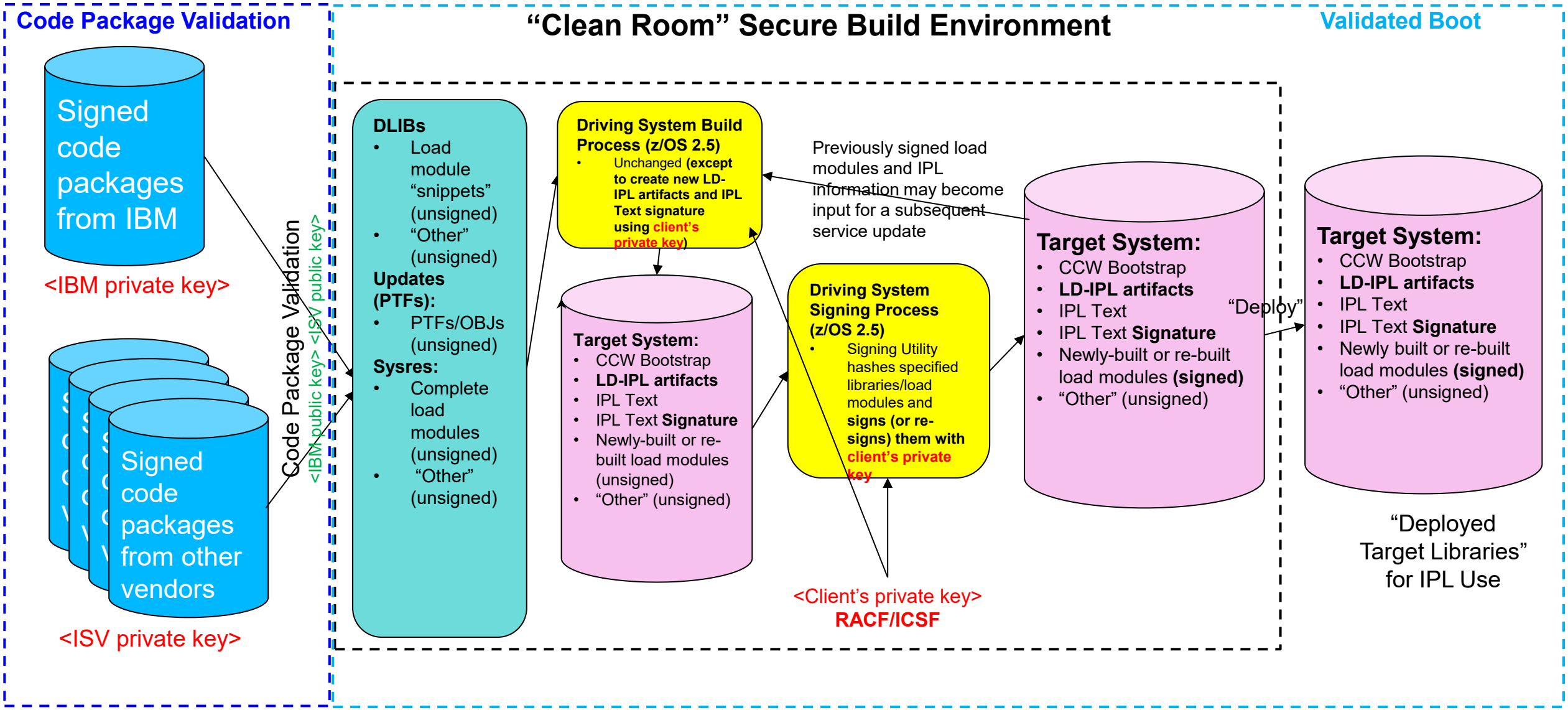
- **FIXCATs**
  - FIXCAT for Validated Boot for z/OS (all support):
    - **IBM.Function.ValidatedBoot**
  - FIXCAT for Exploitation support for z16:
    - **IBM.Device.Server.z16-3931.Exploitation**
  - FIXCAT for “Clean Room” Driving System (front-end support only):
    - **IBM.DrivingSystem-RequiredService**

Support for z/OS Validated Boot is Optional and IPL Volumes are Bimodal

- **The use of z/OS Validated Boot is <u>entirely optional</u>**
  - **Clients may continue to build IPL Volumes in the "classic" way that does not support Validated Boot, and continue to perform unvalidated CCW IPLs, with no change**
  - **Clients may elect to build IPL volumes and sign load module executables in the new way to support Validated Boot, and then perform individual IPLs:**
    - In Validated Boot Enforce Mode (validation failure terminates IPL) using LD-IPL
    - In Validated Boot Audit Mode (validation failures are logged but do not terminate IPL) using LD-IPL
    - Unvalidated, using CCW-IPL
  - **Desired IPL mode is specified by authorized individuals via SE/HMC load panel/load profile UI or APIs**
  - **Software-initiated IPLs (AUTOIPL) *cannot* change the security mode of the IPL**

- ***Use Audit Mode to discover signing and certificate setup issues, and correct them***
- ***Fallback capability to performing unvalidated IPLs exists at all times***

Front-End Driving System Processing



Support Requirements – Front-end Driving System (“Clean Room”)

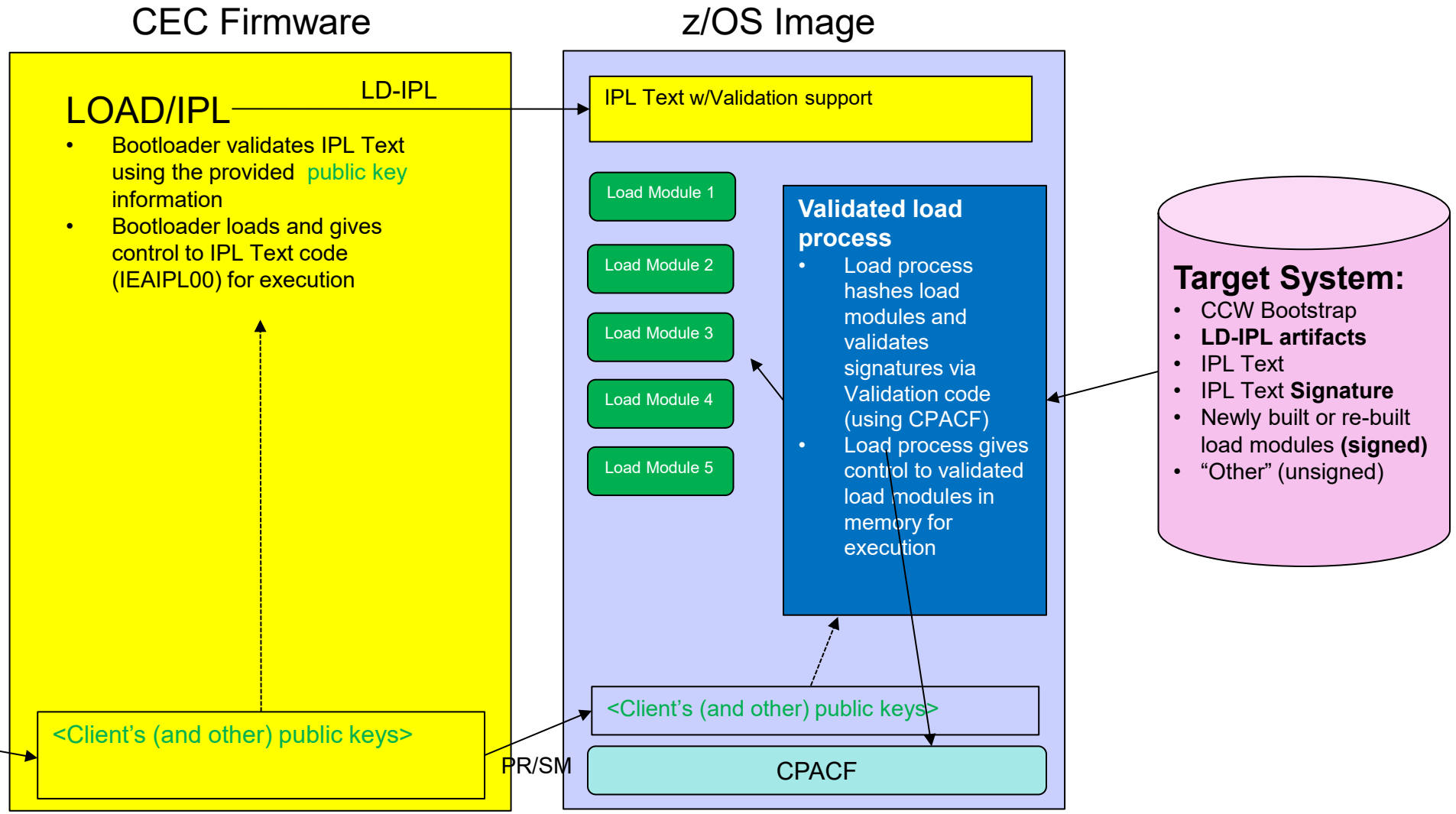
- **No specific Hardware/Firmware requirements**

- **z/OS Software Support**
  - z/OS 2.5 post-GA support, delivered via z/OS CD APARs (5/2023); also 3.1 base
  - ServerPac installation workflow support

- **“One-time” setup**
  - RACF setup (including creation and assignment of signing certificates) needed for doing IPL Text signing and load module signing
  - Build Target System’s IPL Volume with LD-IPL artifacts needed to support Validated Boot

- **Recurring processes**
  - Build/re-build z/OS Target System
  - Sign/re-sign z/OS Target System IPL Text, Nucleus and LPA load modules
  - Rotate/delete/replace signing certificates as needed, and re-sign z/OS Target System

Back-End Target System z/OS Validated Boot Process (via LD-IPL)



Support Requirements – Back-end Target System (“IPLing System”)

- **Hardware/Firmware Support**
  - z16 GA1.5 firmware
  - CPACF digital signature support with Elliptic Curve ECDSA-P521 support and SHA-512 hashing support
  - Virtual Flash Memory (VFM), aka Storage Class Memory (SCM), for use in z/OS paging for LPA pages
- **z/OS Software Support**
  - z/OS 2.5 post-GA support, delivered via z/OS CD APARs (5/2023); also 3.1 base
- **“One-time” setup**
  - Import public-key validation certificates to SE/HMC and assign to the set of LPARs where they will be needed for IPL-time validation
- **Recurring processes**
  - Rotate/delete/replace validation certificates via SE/HMC and reassign to LPARs as needed
  - Customize LOAD profiles and LOAD panel usage to request the desired IPL type for IPLs

Scope of z/OS Validated Boot

- **The following executables are in scope as the validated z/OS "kernel" for validated boot purposes:**
  - IPL Text (IEAIPL00)
  - All load modules through LPA creation:
    - IRIMs and RIMs
    - SYS1.NUCLEUS load modules including z/OS Nucleus (IEANUCxx)
    - LPA load modules (LPALST concatenation) and MLPA/FLPA load modules

- **All subsequent load processing for authorized executables beyond LPA creation is *not* considered part of the validated boot "kernel" in the initial release – but further validation could be added in a future stage of support**
  - Dynamic LPA
  - LNKLST load modules (LNKLST concatenation)
  - General system load processing for authorized load modules

z/OS System Requirements for Validated Boot

- **IPL must be done with CLPA (cold start) on Validated Boot IPLs**
  - Avoids any tampering with LPA executable code on the PLPA page dataset while paged-out on disk
- **On a Validated Boot IPL, we do not support paging out LPA pages to disk – LPA pages are paged to SCM/VFM Flash memory ONLY**
  - Requires use of Virtual Flash Memory (VFM) for PLPA paging
- ***Fail the IPL (in Enforce Mode) or produce appropriate error messages/logs (in Audit Mode) when these requirements are not met on a Validated Boot***

IPL with Validated Boot

- **Each partition has been set up with a set of client public keys that have been imported for use in signature validation, provided via the SE/HMC and LPAR, for this specific partition**
- **SE/HMC load panels offer choices of IPL options**
  - ECKD vs SCSI IPL device, CCW-IPL vs LD-IPL, Enforce mode vs Audit mode
- **Validation of IPL Text (Z Bootloader firmware)**
  - Bootloader performs signature validation for the IPL Text, using the provided list of public keys
  - Validation failure detected? Fail the IPL (in Enforce mode) or log and continue (in Audit mode)
- **Validation of in-scope authorized load modules (z/OS loader)**
  - z/OS Loader hashes the loaded load modules/sections and performs signature validation using CPACF primitives, using the provided list of public keys
  - Validation failure detected? Fail the IPL with non-restartable z/OS wait state without giving control to load module (in Enforce mode) or log/message and continue (in Audit mode)

Z/OS VALIDATED BOOT COMPONENTS
HIGH LEVEL DESCRIPTION AND SETUP STEPS

RACF Signing Service

- **The RACF Signing Service is used by the Signing Utility to sign load modules, and by ICKDSF to sign the IPL Text on an IPL volume**
  - The **RACF Signing Service** and the security manager manages all access to **private** key signing certificates
- **To implement this, the existing R_PgmSignVer SAF callable service (IRRSPS00) signing function is updated to accept new algorithms and return new signature formats for z/OS Validated Boot…**
  - **New signature algorithms**
    - **SHA2-512** used for hashing in creating the signature, hashing the data to be signed
    - **Elliptic Curve ECDSA-P521 key** used for signing
  - **New "bimodal" signature formats**
    - **PKCS#7 in DER format** – signature used when signing IPL Text
    - **Simplified KDSA format** – signature used when signing Load Modules
  - Output also includes generated hash value, certificate fingerprint, key ID, and hash and signing algorithm information, for inclusion in signature metadata

ICKDSF and IPL Volume Initialization

- **ICKDSF supports initializing the IPL volume with all the required IPL records and artifacts for:**
  - CCW IPL records (existing)
  - IPL Text (existing, but updated with new code for Validated Boot)
  - ***LD-IPL artifacts (new)***
  - ***IPL Text signature (in PKCS#7 format) (new)***

- **ICKDSF generates an IPL Text signature whenever the new LD-IPL artifacts are requested to be built on the IPL Volume:**
  - Call **RACF Signing Service** to sign IPL Text at ICKDSF time, as part of client's secure build process

ServerPac

- **ServerPac's Post-Deploy workflow "Create IPL Text" and "Standalone IPL Text" steps can *optionally* use new ICKDSF support for building the required LD-IPL artifacts on the IPL Volume, at build/install time.**
  - ICKDSF generates an IPL Text signature whenever the new LD-IPL artifacts are requested to be built
  - When **not** requested, no LD-IPL artifacts nor signatures are produced, and the IPL volume will only be capable of being used for an unvalidated CCW IPL

- **ServerPac can also optionally invoke the Signing Utility to *sign* all the necessary z/OS load modules as part of the build/install process**
  - *And produce the JCL needed to re-sign these load modules subsequently!*

Signing Utility

- **z/OS provides a new batch utility program, the Signing Utility (IEWSIGN) for doing load module signing**
  - Input: a PDS load library data set for which load module executables are to be signed
    - Sign (or re-sign) all load modules in the data set; Sign all currently-unsigned load modules in the data set; Sign specific load module(s) in the data set – by name or wildcarded name
  - Input and output data sets may be the same (sign-in-place) or different
  - INCLUDE and EXCLUDE member lists tailor which members are to be processed
  - Invokes the **RACF Signing Service** to generate the required hashes and signature(s) for each selected load module
  - Incorporates time-of-signing timestamp and time-of-signing load module directory information as part of the hashed and signed executable, along with the load module contents per se
    - This enables us to detect tampering in the directory entry associated with a load module, as well as in the load module itself
  - Creates one or more Directory Entry records and one or more Signature Records for the different subsets of the load module records, as needed, and stores the directory information, signature(s), and metadata in the load module itself
  - Reports on all load modules that were processed/signed on the current invocation

- **Also supports an "unsign" capability to remove signatures from signed load modules when/if needed**
- **Also supports a "report" capability to report on signing information for members of a PDS load library, without signing or unsigning anything**

Binder – Load Module Format

- **When a load module is signed, new load module records used for Validated Boot are placed *after the current last record* in the load module (for compatibility)**
  - **Directory Entry records**. Embedded Directory Entry information from time-of-signing, including alias information
  - **Signature records**. Contain a KDSA-format signature:
    - Signature record used for normal fetch; If SCTR load, signature record used for nucleus fetch
    - Both signature records contain necessary signature metadata
- **Note that because the load module signatures incorporate directory entry information, most kinds of directory entry changes (such as renaming) will require the load module to be re-signed before it can be used for a subsequent Validated Boot IPL**
- **Any re-build/re-linkedit of a signed load module will make the load module look "unsigned" and restore the "unsigned" load module format, which will require the load module to be re-signed before it can be used for a subsequent Validated Boot IPL**

AutoIPL (Program-Initiated IPL/LOAD)

- **Program-Initiated IPLs may also request a Validated Boot IPL in either enforce or audit mode**
- **Program-Initiated IPLs cannot change the security state of an IPL – only an IPL initiated by an authorized individual via the SE/HMC or DPM UI can do that**
  - AutoIPL is enhanced to support Validated Boot
  - Subsequent AutoIPLs triggered pursuant to a z/OS Validated Boot IPL will all share the same IPL type (LD-IPL vs CCW-IPL) and security mode (either Enforce or Audit mode) as the original z/OS IPL
    - SADMP IPL
    - z/OS re-IPL
    - Or both

Validated Boot Log Data Format Utility

- **Following a z/OS Validated Boot, an in-memory data area mapped by IHAVBA documents various results experienced during z/OS IPL-time validation**
  - Datasets and load modules processed/validated
  - Certificates used/not used
  - Load module validation results
  - Adherence (or not) to system-level requirements
  - …
- **z/OS provides a utility program (IEAVBPRT) to format these data areas after completion of the IPL**
- **z/OS IPCS also provides a similar program (IEAVBIPC) to format these data areas in a dump**

Standalone Dump (SADMP)

- **In addition to z/OS IPL Text, ICKDSF and the Standalone Dump generation process can sign (and Z Bootloader can validate) the SADMP IPL Text**

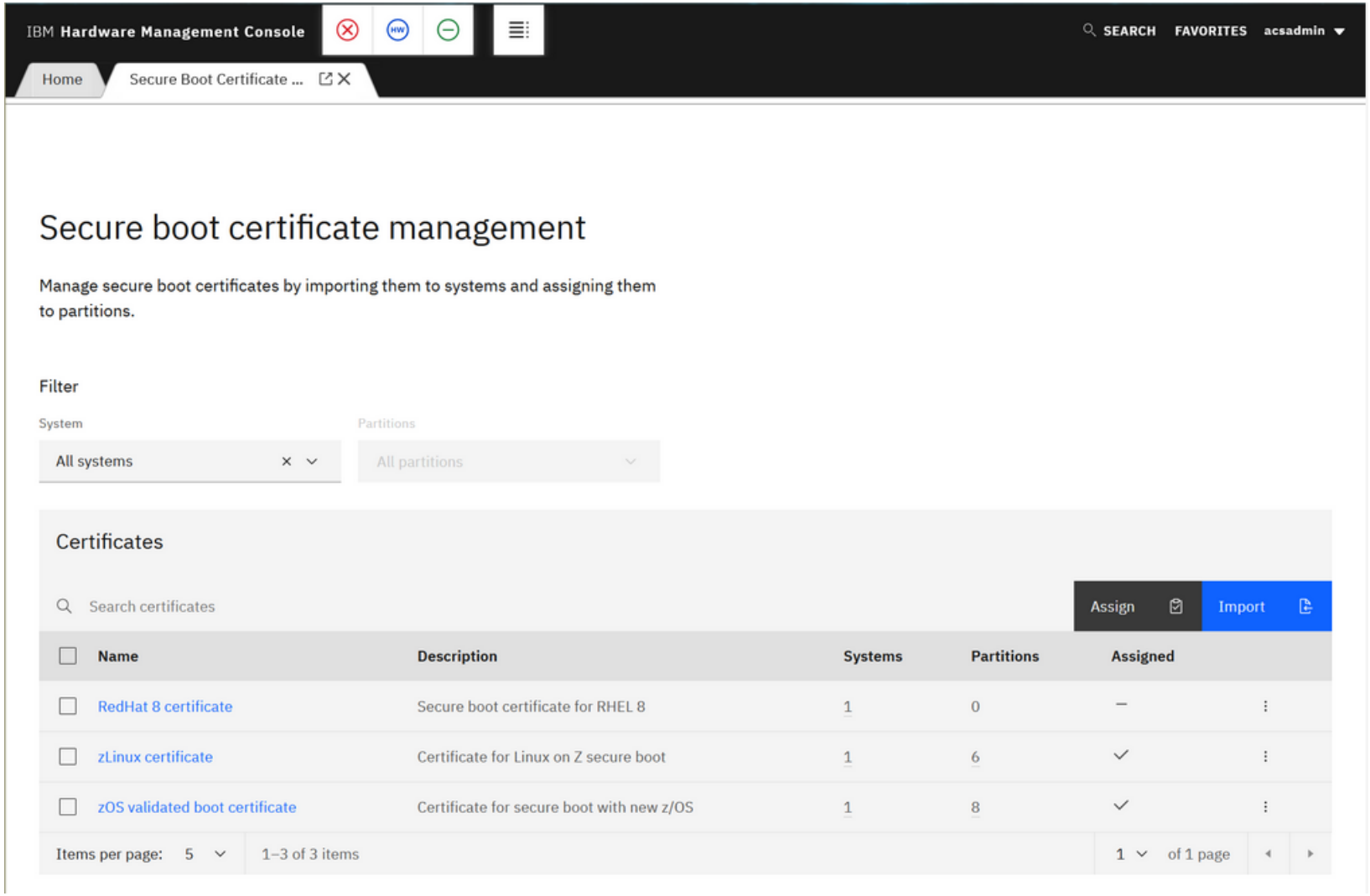
- **In support of Validated Boot, SADMP supports a dump-type IPL making use of the LD-IPL process, and participates in the LD-IPL required save/restore processing of memory areas that would otherwise be overlaid by Z Bootloader (so these memory areas can successfully be dumped)**

- **SADMP also handles “store status” appropriately so that the dumped status info is normally associated with the z/OS image being dumped, but (when needed), it can capture the store status info associated with a prior IPL of SADMP for debug purposes**

z/VM Second-Level Guest Support

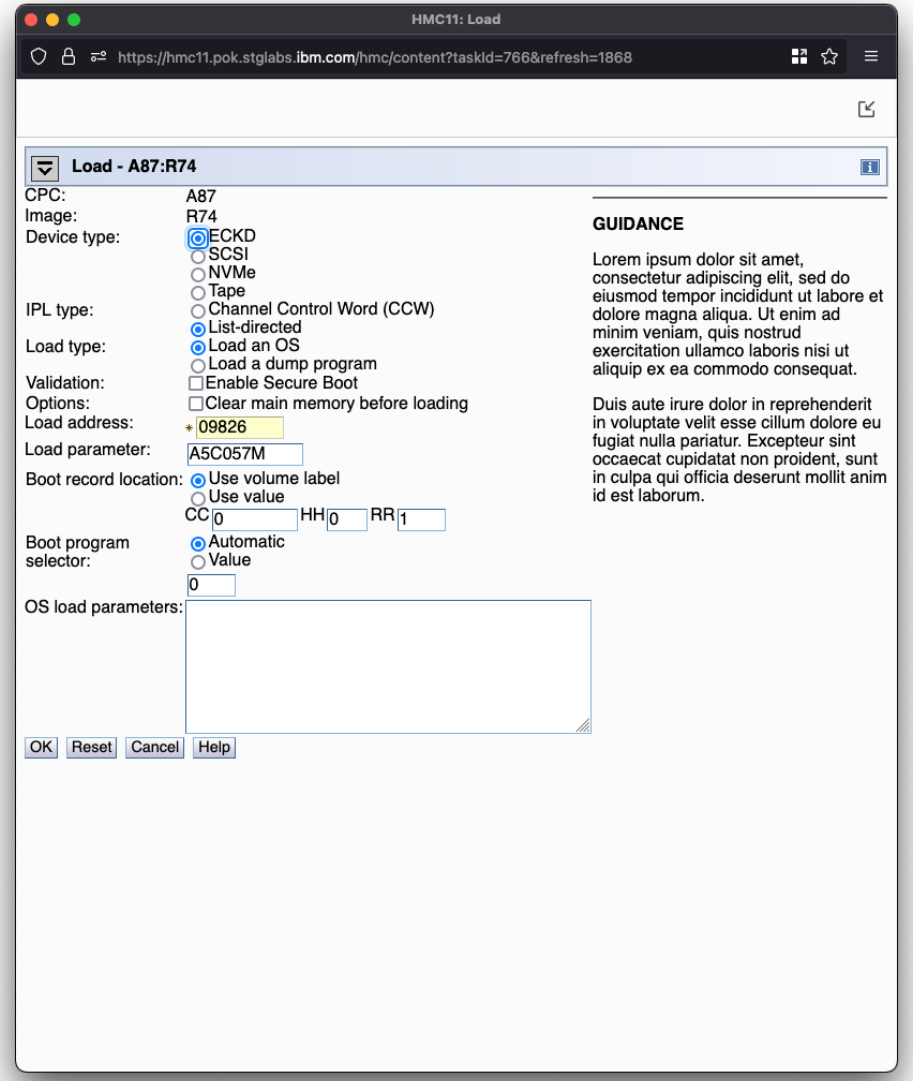
- **z/VM provides *very limited support* for performing a z/OS and/or SADMP Validated Boot in second-level guest z/OS environments**
  - No support is provided for Validated Boot of the z/VM operating system itself
  - No z/VM support is provided for guest-unique key management, just pass-thru of Certificate Store information from the underlying LPAR
  - z/VM does not support the use of VFM for guest paging, which is a z/OS system requirement for Validated Boot.
    - Therefore, only an Audit mode Validated Boot can be successfully performed for a second-level z/OS guest, and it will always report exceptions related to lack of VFM paging for PLPA.

SE/HMC Certificate Management – Certificate View



- **Import**
- **Delete/Remove**
- **Assign (to partitions)**
- **Unassign (from partitions)**
- **Copy**

SE/HMC Load Panel



- **Choice of load IPL Volume type**
  - List all valid candidates.
- **CCW-IPL for compatibility and for Non-Validated Boot**
  - Bimodal IPL Volume support
- **List-Directed IPL (LD-IPL) for Validated Boot**
  - Enforce Mode – “Enable Secure Boot” selected
  - Audit Mode – “Enable Secure Boot” not selected
- **OS vs Dump type IPL**
  - OS for z/OS
  - Dump for SADMP – for LD-IPL, saves memory contents around Bootloader usage

Additional Information on Validated Boot for z/OS

- **Content Solution:** https://www.ibm.com/support/z-content-solutions/validated-boot-for-zos/
- **z/OS Documentation:** https://www.ibm.com/docs/en/zos/2.5.0?topic=solutions-validated-boot-zos
- **z/OS C3 Content Solution PDF:** https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/izsv100_v2r5.pdf
- **White Paper:** https://ibm.biz/zosValidatedBoot
- **GIMZIP code package signing/validation:** https://www.ibm.com/docs/en/zos/2.5.0?topic=guide-preparing-verify-signatures-gimzip-packages

AUDIENCE Q & A

Experience more with IBM

SHARE
Kansas City 2024
August 4-8

Visit us at Booth #613

Engage with IBM experts to explore the world-renowned IBM z16™ **Plexi and LEGO® Brick Model** as well as more than 12+ solution demos

Register for the IBM Z® Client Reception

Monday 8/5, 7:30-9:30pm

No Other Pub, 1370 Grand Blvd

ibm.biz/IBMReception-SHAREKC

Join our 2 Lunch & Learn Sessions

Looking for ways to accelerate your application modernization journey and simplify your platform experience?

Monday 8/5 12:00-1:00pm
Neptune D - Loews KC Level 1

Spin to Win! Navigate Threats at the Wheel of Cybersecurity Game Show

Thursday 8/8 11:15am-12:15pm
Neptune D - Loews KC Level 1

IBM Z Day

October 1

One day free virtual conference open to all to share best practices across our global vibrant community on October 1st.

Register here:
ibm.biz/ibmzday-2024

IBM TechXchange

October 21-24

Fuel your AI at the ultimate IBM learning event. Join us in Las Vegas on October 21-24 for 130+ IBM Z hands on labs, roadmaps , tech deep dives, and the SHARE sponsored IBM z/OS Academy!

Register here:
ibm.biz/TechXchange2024

Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation

Z/OS VALIDATED BOOT BACKUP ISV CONSIDERATIONS

ISV Considerations for Validated Boot

Validated Boot Of or With ISV Code

- **Signing/validation of ISV "code packages" with ISV private keys as part of software distribution process**
  - Validated by client using ISV public keys when unpacked into the clean room
- **Participation in client's secure build and signing process, for load modules that contain some ISV code – which should be mostly transparent …**
  - Load module signing is handled at build time via client's use of the **Signing Utility**
  - Load module validation is handled transparently by z/OS fetch/load processing for those in-scope load modules that happen to contain ISV code
- **Standalone-IPLable ISV products may wish to support Validated Boot for themselves**
  - Analog to SADMP or other similar standalone IPLable vendor programs
  - For dump-type LD-IPLs, will need to participate in the required protocols for handling store status and restoring saved memory areas

ISV Considerations for Validated Boot

Toleration of Validated Boot Load Module Format Changes

- **Provide ISV toleration/understanding of changes to IPL-able volumes to support LD-IPL**
  - If there are programs with dependencies on on-disk formatting details for IPL volumes
- **Provide ISV toleration/understanding of changes to load module format/contents/signatures**
  - Toleration of the presence of one or more embedded Directory Entry and Signature Records in a signed load module
  - Maintaining these records intact, to preserve the “signed” state of a load module, as load modules are “processed” in whatever way – copied, moved, etc.

ISV Considerations for Validated Boot

Be a Part Of, or Provide Analogs to Parts of, the Validated Boot Solution itself

- **Provide ISV analog to the RACF Signing Service, used to generate load module signatures**
  - Non-RACF security products
- **Provide ISV analog to the Signing Utility, used to sign load modules en masse**
  - Perhaps providing a higher-level UI, more advanced sign/unsign or reporting capabilities, etc.
- **Provide ISV analog to ICKDSF, creating IPL-able volumes in LD-IPL format with the artifacts needed to support a Validated Boot**
- **Provide ISV analog to IEAVBPRT, for processing/formatting of new z/OS Validated Boot Audit information/logs**
  - Over and above what z/OS itself does with this information

Z/OS VALIDATED BOOT
BACKUP INFORMATION
ADDITIONAL DETAILS AND SETUP STEPS
SOD AND ANNOUNCEMENTS

RACF Signing Service – Requirements for Certificates Used

- **Signing certificate and its complete certificate chain (if any) up to the root must be in RACF DB and connected to a RACF keyring**
  - The signing certificate must have the associated private key in RACF DB and connected as the DEFAULT certificate, or alternatively, the signing key could be kept in the PKDS or TKDS.

- **CA certificate(s) requirements (for non-self-signed signing certificate)**
  - Certificate extensions
    - If basicConstraints extension exists, cA indicator must be set
    - If KeyUsage extension exists, keyCertSign bit must be set
  - Key type
    - Must be RSA or ECC
  - Signing algorithm on the certificate
    - Any algorithm permissible, except those based on MD2, MD5 and SHA1

- **Signing certificate requirements**
  - Certificate extensions
    - KeyUsage extension must exist with the digitalSignature set. Validated Boot does not care about other KeyUsage attributes, and does not care about ExtendedKeyUsage attributes
    - Subject Key ID extension must exist
  - Key type
    - Must be NISTECC with key size 521 (secp521r1 - { 1 3 132 0 35 })
  - Signing algorithm on the certificate
    - Any algorithm permissible

- **RACF calls ICSF functions to perform signing**

RACF Signing Service – Profile Setup for Signing

- **The RACF Signing Service manages all access to private key signing certificates**
- **Multiple users may be required to sign code. It is possible that some people will sign code for one product, and other people are responsible to sign other code.**
  - The keyring containing the signing key can be scoped on a per user or per group basis.
- **A profile in the FACILITY controls the signing authority and includes APPLDATA which indicates the hash algorithm and keyring**
  - `IRR.PROGRAM.V2.SIGNING.group.user`: qualified by both a RACF connect group name and a RACF user ID.
  - `IRR.PROGRAM.V2.SIGNING.user:` qualified by a RACF user ID
  - `IRR.PROGRAM.V2.SIGNING.group:` qualified by a RACF connect group name
  - `IRR.PROGRAM.V2.SIGNING:` one key ring and message digestion algorithm for everyone
- **These resources are checked in the order shown, and the first one found is used. *Generic profiles cannot be used.***

- **APPLDATA in the profile designates the hash algorithm and keyring which contains the key for signing**
  - Format:

    hash-algorithm keyring_owner/keyring-name

    Where:

    hash-algorithm – The message digestion algorithm to be used for signing. If omitted, default to SHA512.
    keyring_owner/keyring_name – The fully qualified SAF key ring
    E.g. MYID/MYKEYRING
  - Example:

    RDEFINE FACILITY IRR.PROGRAM.V2.SIGNING. product_build_id UACC(NONE) APPLDATA('SHA512 product_build_id/BUILDRING')

    PERMIT IRR.PROGRAM.V2.SIGNING. product_build_id CLASS(FACILITY) ID(product_build_id) ACC(READ)

RACF Signing Service – Example to Setup Certificates (RACF generated) and Keyring

- **A RACF SPECIAL admin performs the following configuration to allow non-SPECIAL user XYZBLD to sign code. Green steps are optional. Red keywords are required.**

- **Create a CERTAUTH certificate that is used to issue code signing certificates to user XYZBLD**
  - `RACDCERT CERTAUTH GENCERT SUBJECTSDN(OU('Code Signing CA') O('XYZ Corp') C('US')) SIZE(4096) RSA(PKDS) WITHLABEL('Code Signing CA')NOTAFTER(DATE(2032-12-31))`
- **Create a code signing certificate with 521 bits NISTECC key stored in PKDS for user XYZBLD signed by the CA created in step 1.**
  - `RACDCERT ID(XYZBLD) GENCERT SUBJECTSDN(CN('XYZ Code Signing') O('XYZ Corp') C('US')) SIZE(521) NISTECC(PKDS) WITHLABEL('XYZ Code Signing') SIGNWITH(CERTAUTH LABEL('Code Signing CA')) KEYUSAGE(HANDSHAKE DOCSIGN) NOTAFTER(DATE(2023-12-31))`
- **Create a key ring to hold the certificates created above**
  - `RACDCERT ID(XYZBLD) ADDRING(CODESIGNKEYRING)`
- **Connect the certificates to the key ring**
  - `RACDCERT ID(XYZBLD) CONNECT(RING(CODESIGNKEYRING) CERTAUTH LABEL('Code Signing CA'))`
  - `RACDCERT ID(XYZBLD) CONNECT(RING(CODESIGNKEYRING) LABEL('XYZ Code Signing') DEFAULT)`
- **Create the RDATALIB class profile to allow user XYZBLD to use the private key in that keyring**
  - `RDEF RDATALIB XYZBLD.CODESIGNKEYRING.LST`
  - `PE XYZBLD.CODESIGNKEYRING.LST CLASS(RDATALIB) ID(XYZBLD) ACC(READ)`
- **Create the FACILITY class profile to allow user XYZBLD to sign using that keyring**
  - `RDEF FACILITY IRR.PROGRAM.V2.SIGNING.XYZBLD APPLDATA('SHA512 XYZBLD/CODESIGNKEYRING')`
  - `PE IRR.PROGRAM.V2.SIGNING.XYZBLD CLASS(FACILITY) ID(XYZBLD) ACC(READ)`

ICKDSF – Create/Format IPL Volume and IPL Text

LDIPL Parameter: write List Directed IPL records and a signed user-supplied IPL program record on the volume

Parameters/ Abbreviations	Description
LDIPL(STANDARD|DUMP)	
LDIPL	Write List Directed IPL records and a signed user-supplied IPL program record on the volume.
STANDARD	Indicates that the user-supplied IPL program is standard IPL text.
DUMP	Indicates that the user-supplied IPL program is for stand-alone dump.
Default:	None.
Restrictions:	Valid only in the MVS version of ICKDSF. Valid only when the IPLDD parameter is specified. The SYSIN option for passing in the user-supplied IPL program in the IPLDD parameter is not supported.

LDIRTS Parameter: specify which user-supplied IPL program record number to sign when doing List Directed IPL

Parameters/ Abbreviations	Description
LDIRTS(record number)	Specifies the record number for the user-supplied IPL program that is to be signed and placed on the volume when doing List Directed IPL. For record number specify a value between 1 to 10 to identify which record passed in with the IPLDD parameter is to be signed.
Default:	4
Restrictions:	Valid only in the MVS version of ICKDSF. Valid only when the LDIPL parameter and the IPLDD parameter with the ABSFORMAT sub parameter are specified.

IPLDD parameter: write a user-supplied IPL program on the volume

Parameters/ Abbreviations	Description
IPLDD({dname|sysin}[, OBJFORMAT|,ABSFORMAT])	
IPLDD IPL	Allows you to supply an IPL program to be written on the volume or minidisk.
OBJFORMAT OBJECT OBJ	Specifies that IPL data is being supplied in object deck format; that is, cards will have one of the following strings of EBCDIC characters in columns 2 through 4: TXT, RLD, ESD, END Note that only cards with TXT will be processed. All others will be ignored.
ABSFORMAT ABSOLUTE ABS	Specifies that IPL data is being supplied as variable-length records that contain executable instructions.
Default:	OBJFORMAT The system provides special IPL bootstrap records if you specify the IPLDD parameter without specifying the BOOTSTRAP parameter.
Restrictions:	When an IPL program is included in the SYSIN stream, it must immediately follow the command and end with an ENDIPLTEXT card. The ENDIPLTEXT card is optional when the IPL program is in a data set other than the one specified by SYSIN, or when the end-of-file indicator (/*) immediately follows the data for the IPL program. The IPLDD parameter is not valid for 3995-151 and 3995-153. IPLDD is not valid in the VSE environment.

IEWSIGN Signing Utility – JCL example

```
//SIGN EXEC PGM=IEWSIGN,
//  PARM='ACTION=SIGN,STATE=UNSIGNED,VERBOSE=YES'
//STEPLIB DD DSN=SYS1.SIEAMIGE,DISP=SHR
//*
//INFILE  DD DSN=USER100.TEST.PDS,DISP=SHR
//OUTFILE DD DSN=USER100.TEST.SIGNED.PDS,
//           DISP=(NEW,CATLG),DSNTYPE=PDS,
//           SPACE=(CYL,(5,5,5)),UNIT=3390
//*
//SYSPRINT DD SYSOUT=*
//*
//INCLUDE DD *
  LM*
//EXCLUDE DD *
  LM2*
```

- **Parameter explanations:**
  - ACTION=SIGN: sign load module
  - STATE=UNSIGNED: sign unsigned load modules in INFILE
  - VERBOSE=YES: output detailed messages
- **SYSPRINT:** messages from the signing utility
- **INCLUDE**
  - LM* : include all members/aliases matching “LM*”
- **EXCLUDE**
  - LM2* : exclude all members/aliases matching “LM2*”

IEWSIGN Signing Utility – SYSPRINT example

```
 INFILE summary:       members   aliases    signed  unsigned    non-LM
                             3         2         0         5         0

Member/alias selected from INFILE with STATE=UNSIGNED:
Member      Alias(es)
LM1         AL11     AL12
LM222
PGM1

Member/alias selected from INFILE after INCLUDing:
Member      Alias(es)
LM1         AL11     AL12
LM222

Member/alias selected from INFILE after EXCLUDing:
Member      Alias(es):
LM1         AL1      AL2

Signing results:
LM1      RC=0
Signing summary:    processed      RC=0      RC=4      RC=8
                            1         1         0         0

 OUTFILE summary:      members   aliases    signed  unsigned    non-LM
                             1         2         3         0         0
Comparing directory entries between INFILE and OUTFILE ...
Members compared:    1, unmatched:    0.
Aliases compared:    2, unmatched:    0.
Task completed with RC=0.
```

- **Member/alias selected from INFILE with STATE=UNSIGNED:**
  - List all valid candidates.
- **Member/alias selected from INFILE after INCLUDing:**
  - PGM1 has been removed.
- **Member/alias selected from INFILE after EXCLUDing:**
  - LM222 has been removed.
- **Signing summary**
  - Summarize the number of RC=0/4/8
- **Comparing directory entries**
  - Verify directory entries.

IEWSIGN Signing Utility – Other JCL Examples

- **Sign all load modules:**

```
//SIGN EXEC PGM=IEWSIGN,PARM='ACTION=SIGN’
//STEPLIB  DD DISP=SYS1.SIEAMIGE
//SYSPRINT DD SYSOUT=*
//INFILE   DD DSN=SYS1.LPALIB,DISP=SHR,UNIT=3390,VOL=SER=BPX111
//OUTFILE  DD DSN=SYS1.LPALIB,DISP=SHR,UNIT=3390,VOL=SER=BPX222
```

- **Sign currently-unsigned load modules, in-place, under control of INCLUDE:**

```
//SIGN EXEC PGM=IEWSIGN,PARM='ACTION=SIGN,State=Unsigned’
//STEPLIB  DD DISP=SYS1.SIEAMIGE
//SYSPRINT DD SYSOUT=*
//INFILE    DD DSN=SYS1.LINKLIB,DISP=SHR
//OUTFILE   DD DSN=SYS1.LINKLIB,DISP=SHR
//INCLUDE   DD *
# Sign selected things
AMBLIST
# blanks are allowed

IEHMVE*
```

- **Unsign signed load modules, in-place:**

```
//UNSIGN EXEC PGM=IEWSIGN,PARM='ACTION=UNSIGN'
//STEPLIB  DD DISP=SYS1.SIEAMIGE
//SYSPRINT DD SYSOUT=*
//INFILE    DD DSN=SYS1.LPALIB,DISP=SHR
//OUTFILE   DD DSN=SYS1.LPALIB,DISP=SHR
```

IEAVBPRT – Reporting on Validated Boot IPL Findings

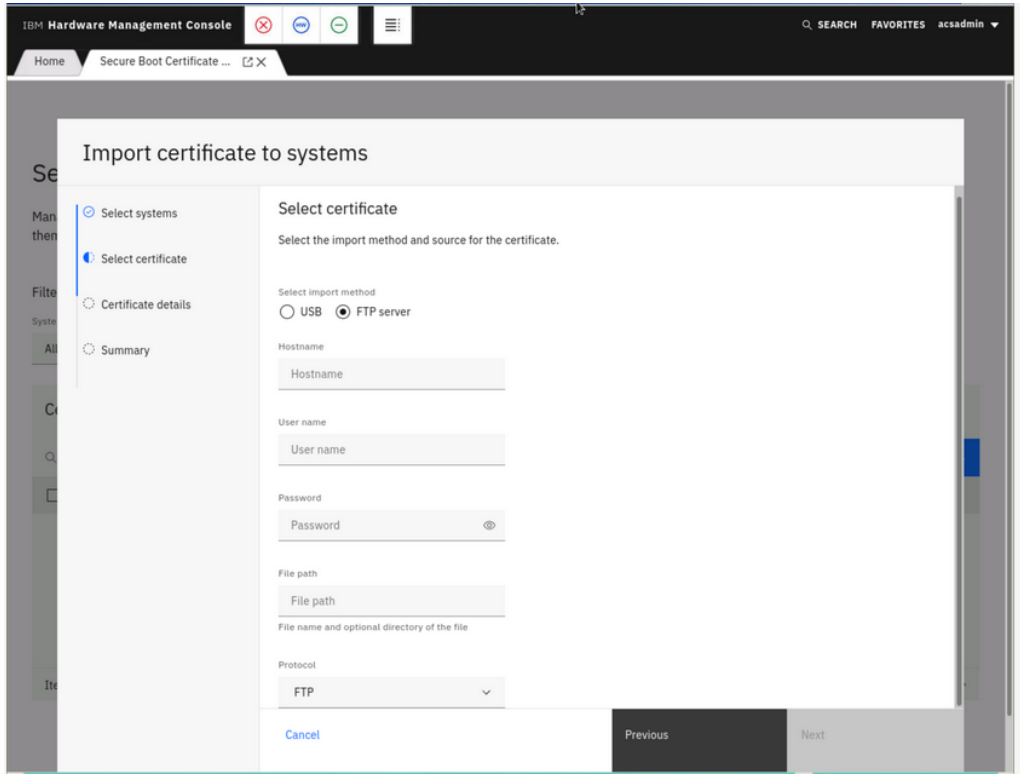
- **Invocation JCL:**

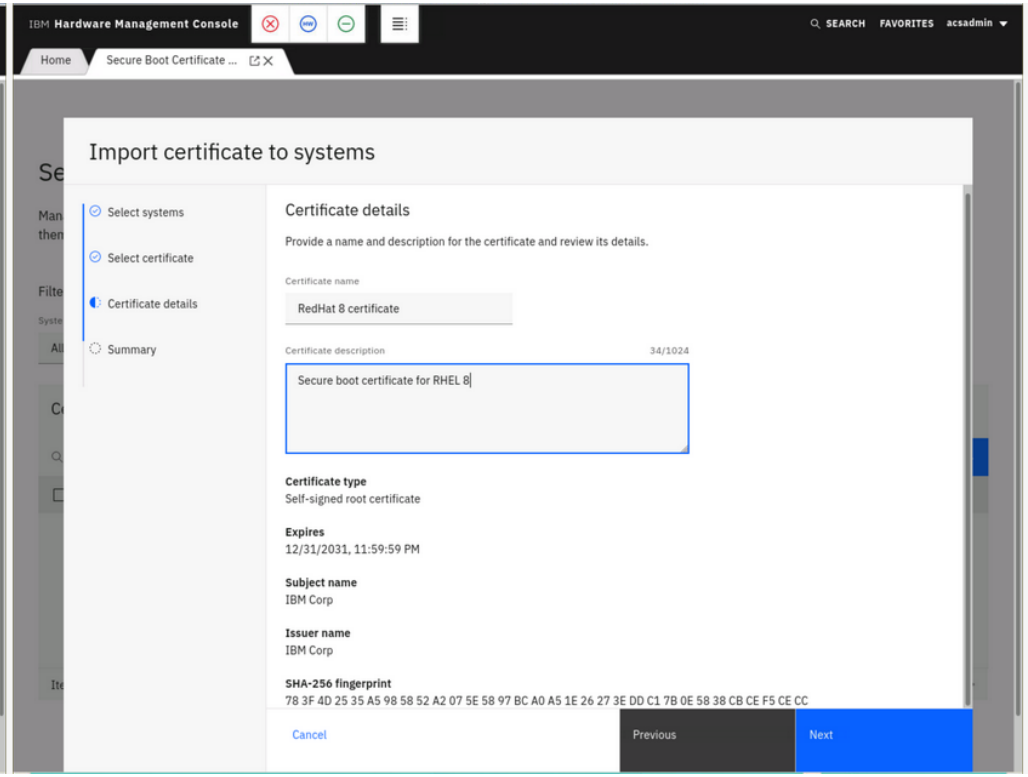
```
//VBPRT1 EXEC PGM=IEAVBPRT,TIME=1440,
//  PARM='DETAIL'
//SYSPRINT DD SYSOUT=*
```

- **Sample output:**

```
IEAVB001I Validated Boot Information
IEAVB011I PLPA page data set was specified. It would not be used if enforce mode

IEAVB003I Audit Information
  Total verification failures: 3104
  Number of DSNEs: 16
  Number of DSNE ModEs: 2938

  DSN(VOL): CEE.SCEELPA(SDR25)
    Total DSN verification failures: 10
    Number of DSNE ModEs: 10

    Modname: CEECOPP
      Reason: Module was not signed
      Fetch Type: LPA
      Number of failures: 1
      When first failed: 2022/09/19 11:05:39
      Key ID: not known
      Cert Name: not known
      When signed: not applicable
  <etc.>

IEAVB005I Valid Certificates

  Name: CERT 1 IS GOOD
    Key ID: 21CC95D0_8A12F9FE_5AA01598_430EF6A0_8D58DFDE
    Successful uses: 2011
    Valid as of: 2022/09/15 12:21:22
    Expiration:  2022/09/26 12:26:36

IEAVB007I Discarded Certificates

  Name: CERT 3 NOT STARTED
    Reason: Not valid yet
    Key ID: 21CC95D0_8A12F9FE_5AA01598_430EF6A0_8D58DFDE
    Valid as of: 2042/09/17 23:53:47
    Expiration:  2042/09/17 23:53:47

  Name: CERT 4 EXPIRED
    Reason: Expired
    Key ID: 21CC95D0_8A12F9FE_5AA01598_430EF6A0_8D58DFDE
    Valid as of: 2015/12/15 13:24:57
    Expiration:  2015/12/15 13:24:57
```

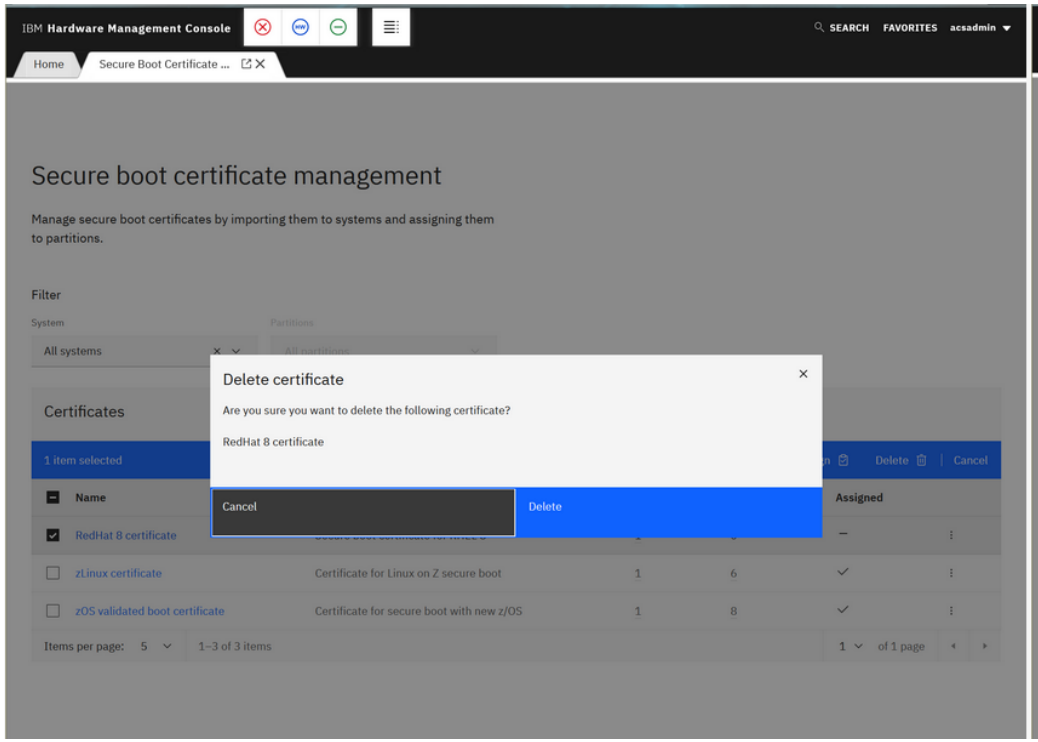
SE/HMC Certificate Management - Import

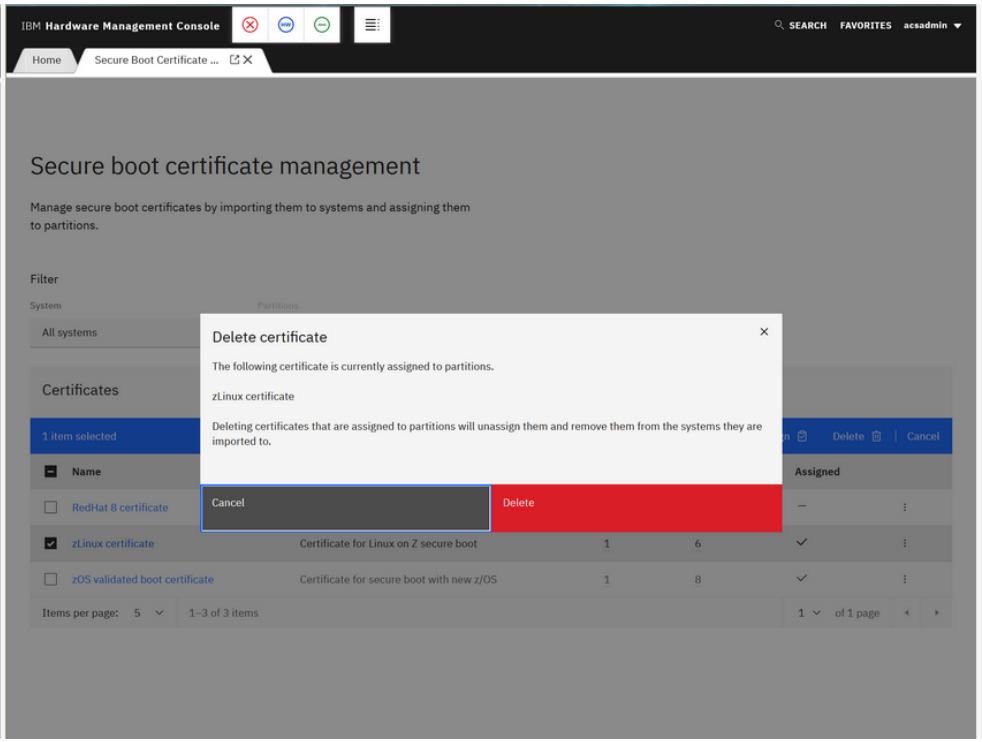




- **Import**
  - Select systems
  - Select certificate to import (from USB or FTP)
  - Show certificate details
- **Delete/Remove**
- **Assign (to partitions)**
- **Unassign**
- **Copy**

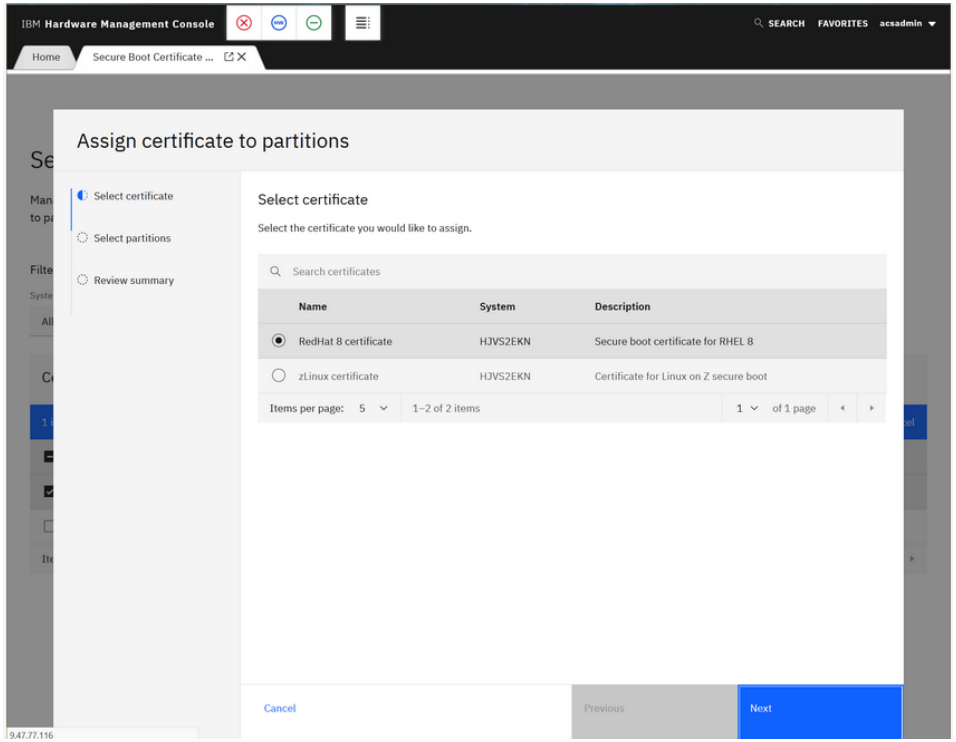
SE/HMC Certificate Management – Delete/Remove

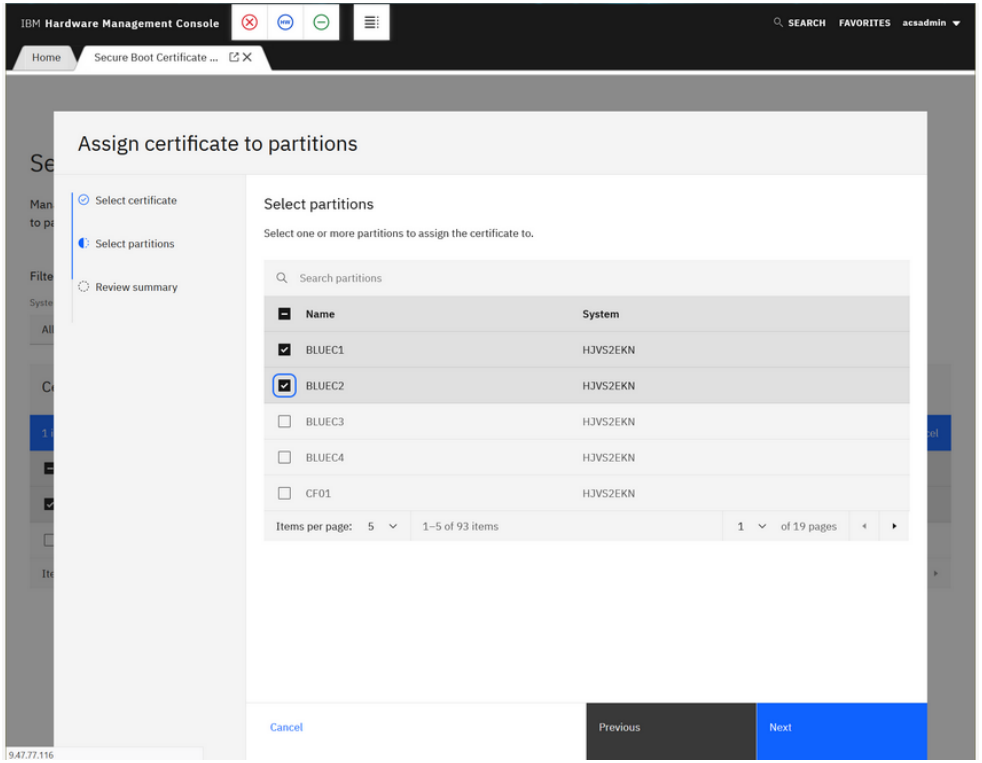




- Import
- **Delete/Remove**
  - Select certificate
  - Confirm impact of deletion/removal
  - Both unassigned certificates and assigned certificates
- Assign (to partitions)
- Unassign
- Copy

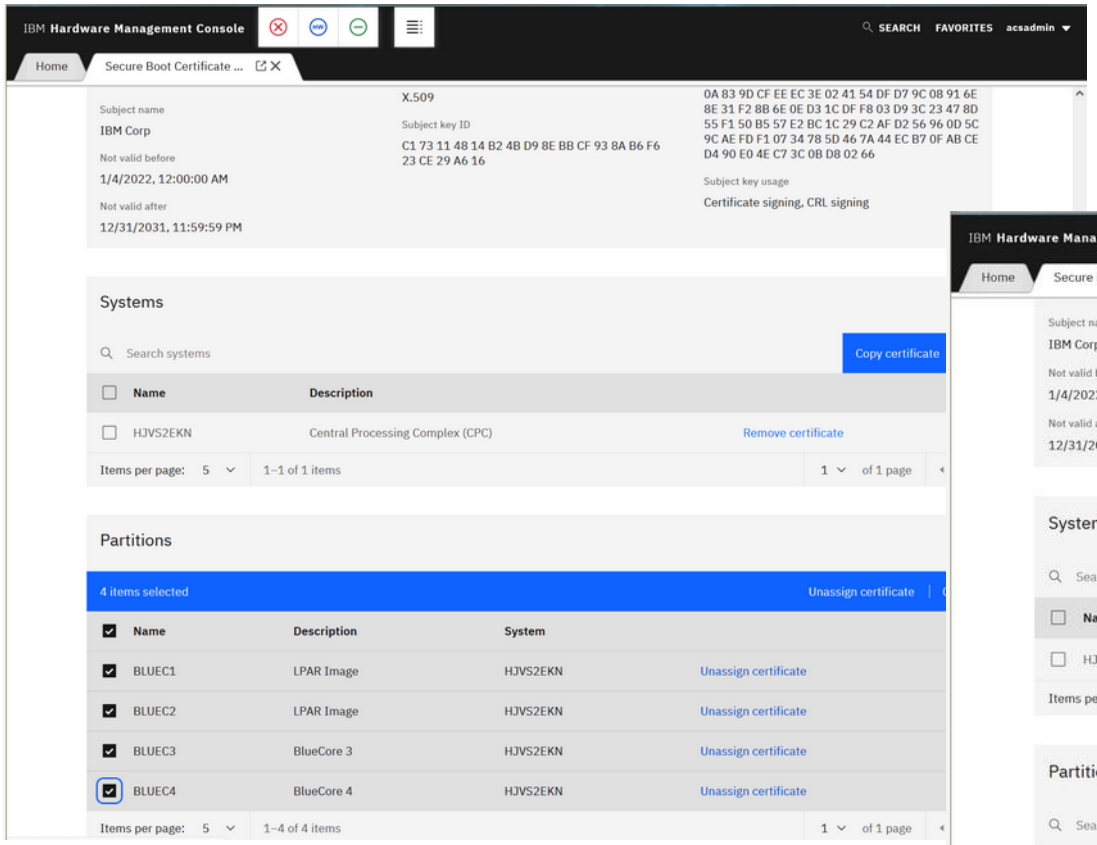
SE/HMC Certificate Management - Assign

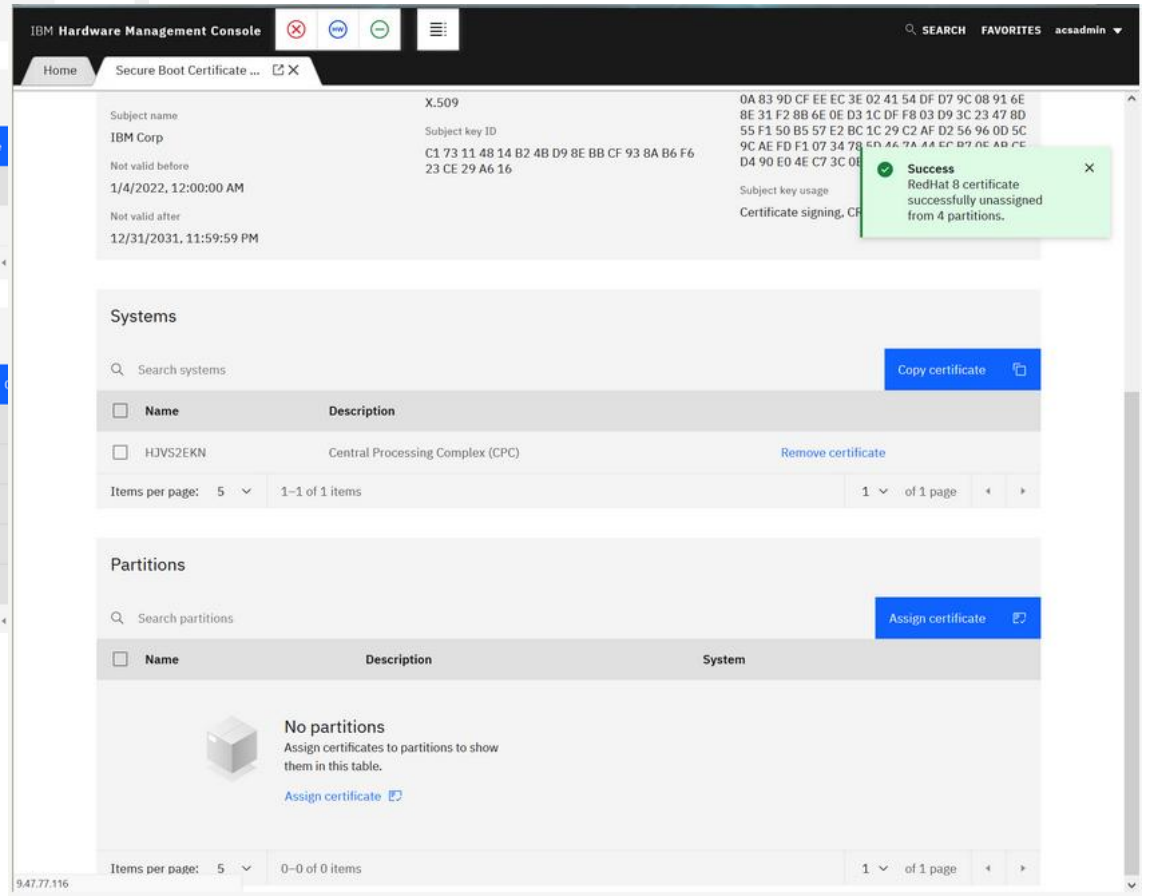




- Import
- Delete/Remove
- **Assign (to partitions)**
  - Select certificates to assign
  - Select partitions to assign them to
- Unassign
- Copy

SE/HMC Certificate Management - Unassign





- Import
- Delete/Remove
- Assign (to partitions)
- **Unassign**
  - Select certificates to unassign
  - Select partitions to unassign them from
- Copy

Product Announcement for z/OS 2.5 Continuous Delivery 1Q2023

- **GIMZIP package signing and verification in z/OS SMP/E and z/OSMF Software Management**

  To provide higher standards for security and integrity of z/OS software packages delivered to clients, z/OS SMP/E and z/OSMF Software Management now provide the capability to digitally sign and verify the signature of GIMZIP packages of software that may be delivered both electronically and physically, on all supported z/OS releases. This capability, designed for nonrepudiation and authenticity, ensures that a software package has not been modified since it was created and the package was signed by the expected provider. The choice to do this additional verification is left to the user upon receipt of the software package if the software provider has chosen to exploit this additional capability. IBM intends to sign the following software packages to allow clients more secure coverage for z/OS software deliverables:

  - z/OSMF ServerPac, also known as z/OSMF portable software instances
  - CBPDO
  - Electronic Shopz PTF orders
  - SMP/E RECEIVE ORDER PTFs and HOLDDATA

Product Announcement for z16 GA1.5 – April 2023

- **Validated Boot for z/OS:** With IBM z16 and accompanying z/OS V2.5 operating system support, IBM is providing basic support for performing a Validated Boot (IPL) of z/OS systems, using IPL volumes defined and built for ECKD DASD devices. The solution uses digital signatures to provide an IPL-time check that the z/OS system, including z/OS nucleus and LPA load module executables, is intact, untampered with, and originates from a trusted source from the time at which it was built and signed. This enables the detection of subsequent unauthorized changes to those software executables, whether those changes be accidental or malicious in nature. This new functionality is designed to meet standards such as the National Information Assurance Program (NIAP) Protection Profiles 4.2.1 (4.3).

Product Announcement for z16 GA1.5 – April 2023

- **Secure Boot for ECKD devices:** Linux on IBM Z has supported secure boot from FCP-attached devices since IBM z15. With support for a Validated Boot for z/OS, support in IBM z16 extends existing Linux secure boot capabilities to ECKD devices and allows client-provided validation certificates provided through the SE/HMC to be used for validation purposes during Linux secure boot. This new function is embedded in the baseline offering functionality. In addition, z/VM(R) 7.3 has been enhanced with support to securely boot a Linux guest. Details on setup requirements can be found in the IBM z16 A02 and IBM z16 Rack Mount Technical Guide.

Product Announcement for z/OS 2.5 Continuous Delivery 2Q2023

- **Validated Boot for z/OS**

  With the IBM z16 and the accompanying z/OS 2.5 operating system support, IBM is providing basic support for performing a Validated Boot (IPL) of z/OS systems, using IPL volumes defined and built on ECKD DASD devices. The solution uses digital signatures to provide an IPL time check that the z/OS system, including z/OS nucleus and LPA load module executables, is intact, untampered with, and originates from a trusted source from the time it was built and signed. This enables the detection of subsequent unauthorized changes to those software executables, whether those changes be accidental or malicious in nature.

  When the target system is built and digitally signed as part of the client's secure build process, the target system can be IPLed using List-Directed IPL (LD-IPL) with digital signature validation in either Enforce or Audit mode, or IPLed without digital signature validation using CCW-IPL. In Enforce mode, an IPL will terminate if there are validation failures for any of the load modules protected by Validated Boot or if the necessary configuration requirements are not met; in Audit mode, the IPL will continue, but audit records will be produced to describe the validation problems encountered.

  Note: The use of Validated Boot for z/OS IPLs is entirely optional and at the discretion of the client. A z/OS system can continue to be built without supporting or being signed for use with Validated Boot. Also, a z/OS system that has been built and signed for use with Validated Boot can be IPLed either with or without signature validation performed for it.

  z/OS support for Validated Boot on z/OS V2.5 requires the installation of PTFs associated with FIXCAT IBM.Function.ValidatedBoot.

  To learn more about Validated Boot for z/OS, see the Validated Boot for z/OS content solution