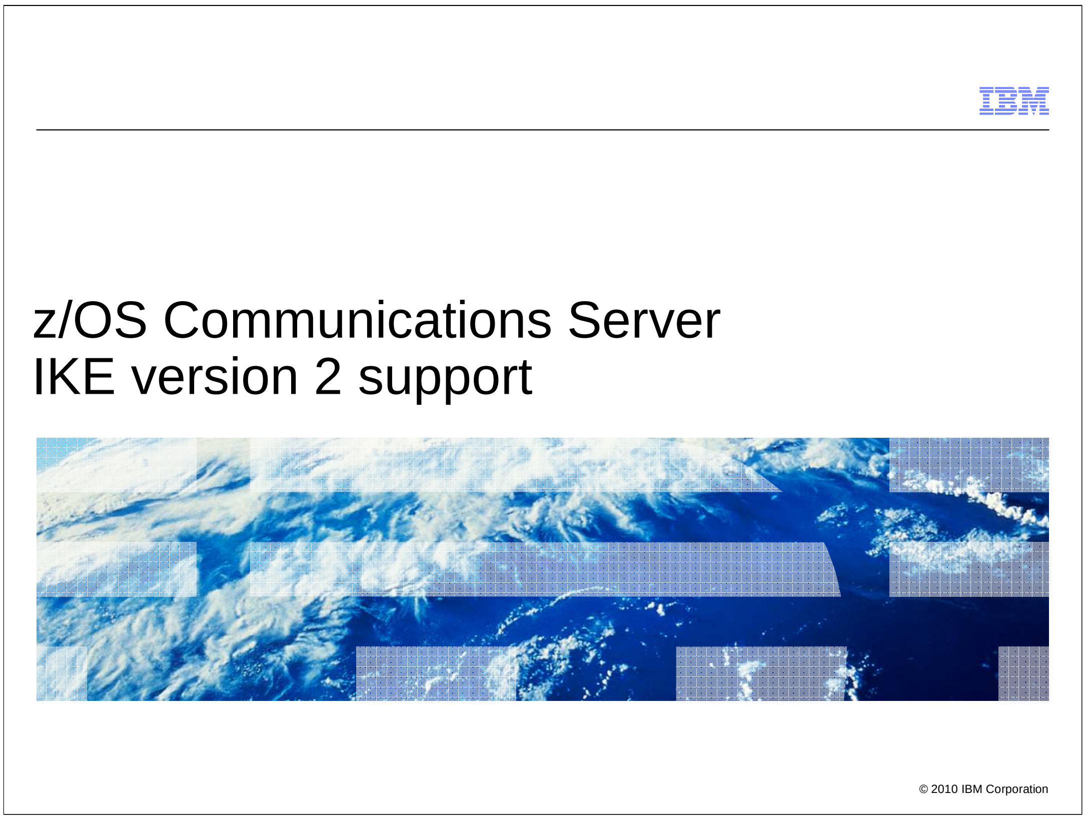
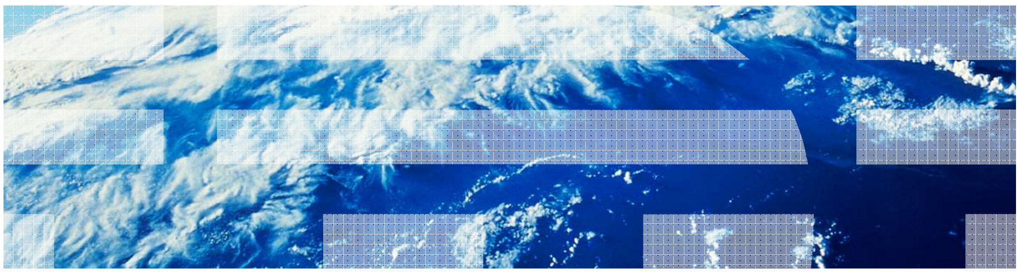
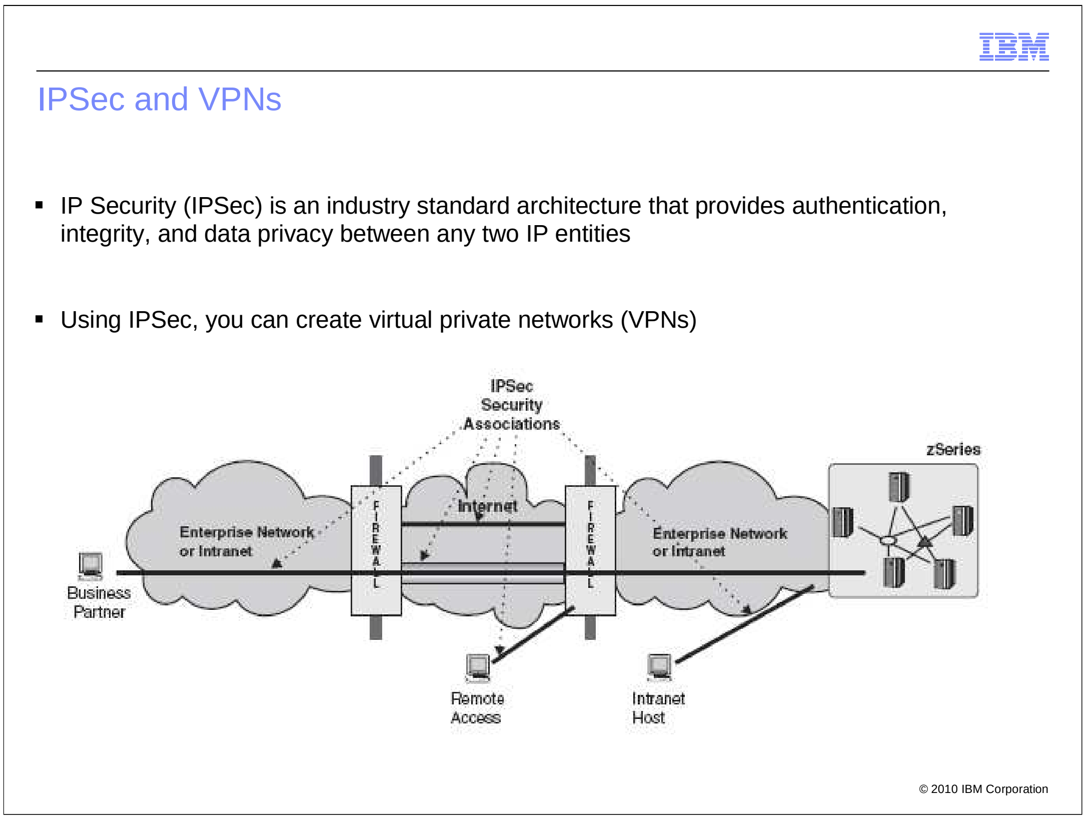


z/OS Communications Server IKE version 2 support

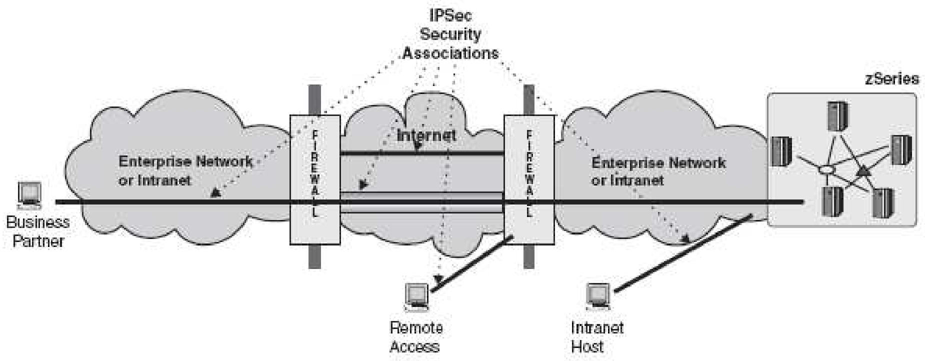


There are many new security functions in z/OS V1R12 Communications Server. This presentation describes how IKED now supports IKE version 2 in addition to IKE version 1.



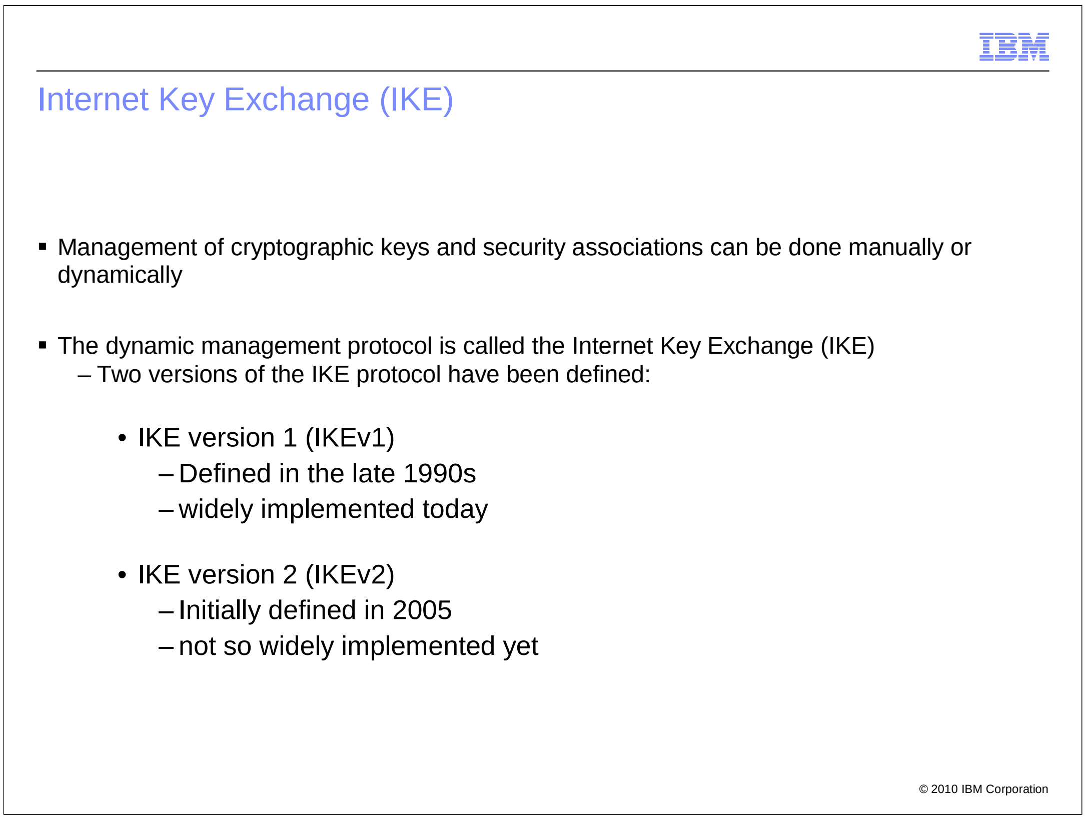
IPSec and VPNs

- IP Security (IPSec) is an industry standard architecture that provides authentication, integrity, and data privacy between any two IP entities
- Using IPSec, you can create virtual private networks (VPNs)


IPSec Security Associations
zSeries
Internet
FIREWALL
FIREWALL
Enterprise Network or Intranet
Enterprise Network or Intranet
Business Partner
Remote Access
Intranet Host

IP Security is an industry standard architecture that provides authentication, integrity, and data privacy between any two IP entities. Authentication ensures that the peer is who you think he is. Data integrity ensures that the data you receive is exactly what the peer sent, and was not modified. Data privacy ensures that middle nodes which transport the data cannot read the data.

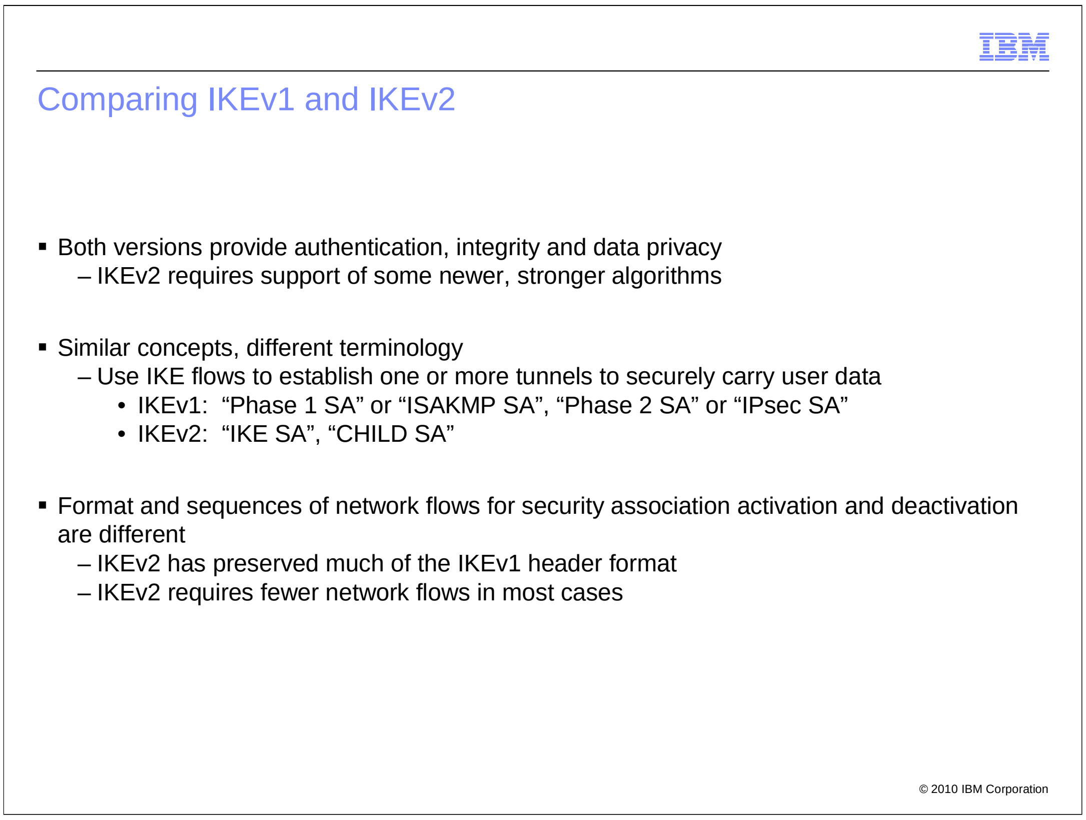
Using IPSec, you establish a secure tunnel, called a security association, with peer nodes. The terms “tunnel” and “security association” are used interchangeably in this presentation. Peer authentication is performed during security association activation, and the security association provides data integrity and security for the traffic it carries. Thus, you extend your private network across a public transport, such as the Internet.



Internet Key Exchange (IKE)

- Management of cryptographic keys and security associations can be done manually or dynamically
- The dynamic management protocol is called the Internet Key Exchange (IKE)
  - Two versions of the IKE protocol have been defined:
    - IKE version 1 (IKEv1)
      - Defined in the late 1990s
      - widely implemented today
    - IKE version 2 (IKEv2)
      - Initially defined in 2005
      - not so widely implemented yet

IPSec security associations and their cryptographic keys can be manually defined, or dynamically created. The protocol used for dynamically creating, refreshing and deleting IPSec security associations is called the Internet Key Exchange (IKE). Two versions of IKE are defined: IKEv1 and IKEv2. IKEv1 was defined in the late 1990s and is widely implemented. IKEv2 was more recently defined, and is not yet widely implemented.



Comparing IKEv1 and IKEv2

- Both versions provide authentication, integrity and data privacy
  - IKEv2 requires support of some newer, stronger algorithms
- Similar concepts, different terminology
  - Use IKE flows to establish one or more tunnels to securely carry user data
    - IKEv1: “Phase 1 SA” or “ISAKMP SA”, “Phase 2 SA” or “IPsec SA”
    - IKEv2: “IKE SA”, “CHILD SA”
- Format and sequences of network flows for security association activation and deactivation are different
  - IKEv2 has preserved much of the IKEv1 header format
  - IKEv2 requires fewer network flows in most cases

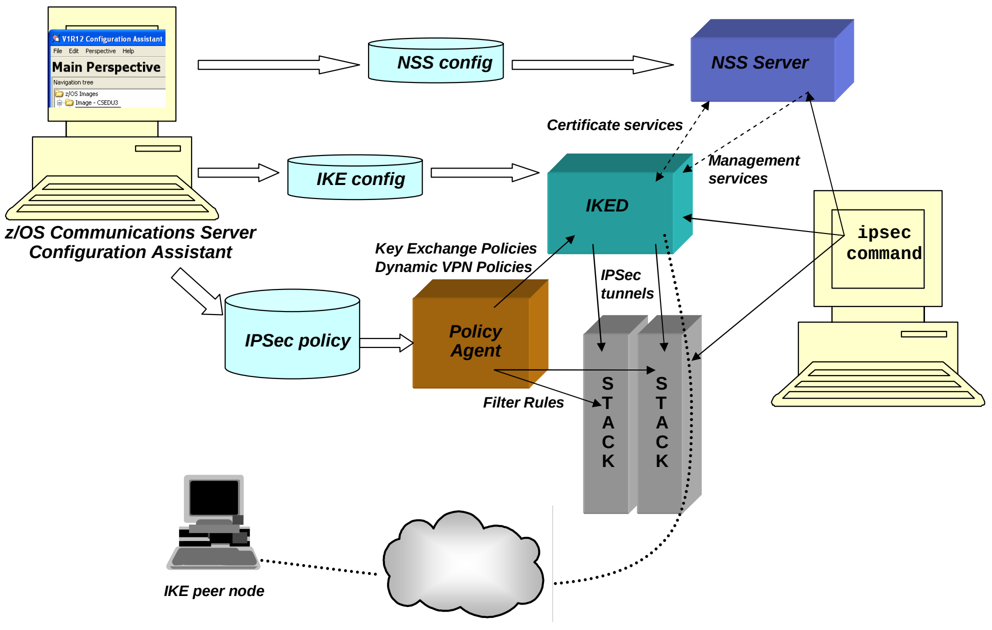
Both versions of IKE provide authentication, and both create security associations that carry data with integrity and privacy. Because it was defined more recently, the IKEv2 architecture requires base implementations to support some of the newer, stronger cryptographic algorithms.

IKEv2 is similar to IKEv1 in concept, but in some cases the terminology has changed. Both protocols first establish a tunnel to securely carry the IKE flows to the peer IKE node, then subsequently send IKE flows over that tunnel to establish secure tunnels for the data traffic. However, the terms used to describe those tunnels has changed. The z/OS Communications Server publications use the terms “phase 1” and “phase 2” to see those tunnels when the IKE version used to activate them is not important to the context. With both protocols, tunnels can be refreshed, and have limited life span based on time or bytes.

The format and sequences of network flows used to dynamically activate, rekey, and deactivate IKEv2 security associations is different than that for IKEv1. Because the IKEv2 header is very similar to the IKEv1 header, both protocols can be run over the same UDP port simultaneously. This allows one IKE daemon to support both IKEv1 and IKEv2, instead of having a separate IKE daemon for the IKEv2 protocol. In most cases, the IKEv2 protocol requires fewer flows, so it is more efficient than IKEv1. However, interoperability is not supported: IKEv2 requests need IKEv2 responses.

Another difference is that IKEv2 requires fewer network flows in most cases. IKEv1 has IKE negotiation modes (Main, Aggressive, Quick). IKEv2 has “initial exchanges” and subsequent exchanges.

Diagram of z/OS processes in IPSec



z/OS communications server supports IKE using a collection of processes and daemons, each one serving a specific role.

z/OS Communications Server Configuration Assistant is the graphical user interface (GUI) for defining daemon configuration files and IPSec policies.

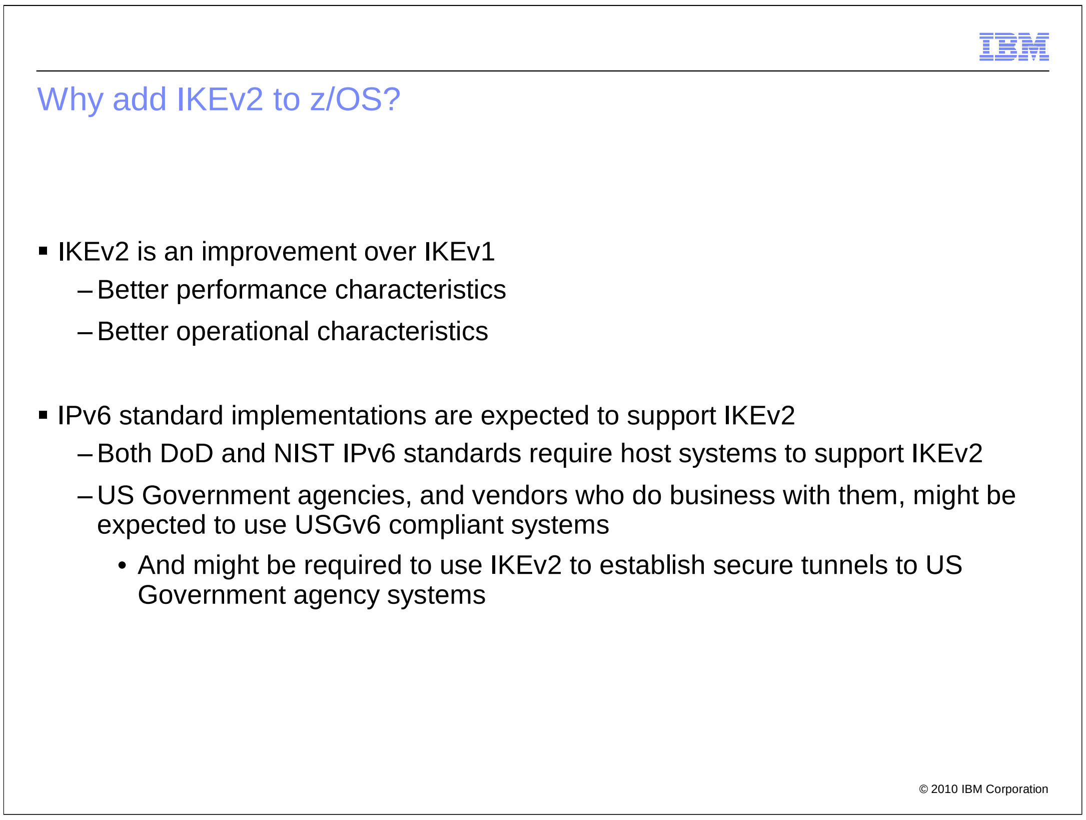
Policy Agent (PAGENT) parses and installs IPSec policies (filter rules, key exchange rules, and others)

TCP/IP stack enforces filter rules, and encrypts and decrypts data flowing on a tunnel.

IKE daemon (IKED) performs dynamic management of security associations using the IKE protocol. It controls activation, deactivation, and rekeying of the tunnels, by exchanging IKE messages with IKE peer nodes. It receives IPSec policies from PAGENT, and negotiates and installs tunnels into the stacks. It performs some basic certificate services itself, or can request those services from a Network Security Services server.

The Network Security Services (NSS) server daemon (NSSD) provides centralized management and certificate services to multiple IKE daemons.

The IPSec command allows operator management (display, activation, deactivation, and refresh) of security associations.

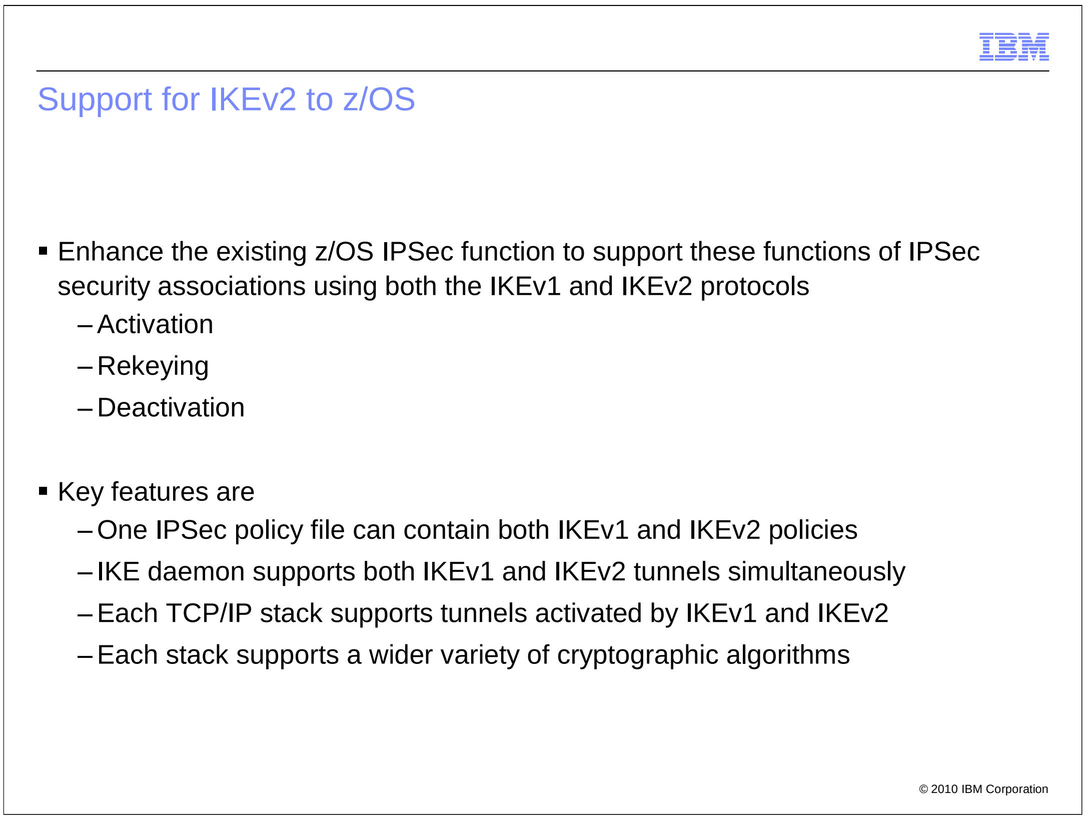


Why add IKEv2 to z/OS?

- IKEv2 is an improvement over IKEv1
  - Better performance characteristics
  - Better operational characteristics
- IPv6 standard implementations are expected to support IKEv2
  - Both DoD and NIST IPv6 standards require host systems to support IKEv2
  - US Government agencies, and vendors who do business with them, might be expected to use USGv6 compliant systems
    - And might be required to use IKEv2 to establish secure tunnels to US Government agency systems

IKEv2 is an improvement over IKEv1 in many ways. It has better performance characteristics, because it was designed to use fewer messages to establish and rekey tunnels. IKEv2 allows rekeying without reauthentification. It also has better operational characteristics than IKEv1, because its designers had the experience of IKEv1 to build on. It was designed to solve some of the problems that plagued IKEv1. Minimal IKEv2 implementations must support newer cryptographic algorithms. IKEv2 avoids duplicate or orphaned tunnels. IKEv2 has built-in dead-peer detection.

Industry standards for IPv6 implementations include IKEv2 support. Publications from both the U.S. Department of Defense (DoD) and the National Institute of Standards & Technology (NIST) require compliant systems to support IKEv2. US Government agencies, and vendors who do business with them, might be expected to use USGv6 compliant systems, and to use IKEv2 to establish secure communications with them.


Support for IKEv2 to z/OS

- Enhance the existing z/OS IPSec function to support these functions of IPSec security associations using both the IKEv1 and IKEv2 protocols
  - Activation
  - Rekeying
  - Deactivation
- Key features are
  - One IPSec policy file can contain both IKEv1 and IKEv2 policies
  - IKE daemon supports both IKEv1 and IKEv2 tunnels simultaneously
  - Each TCP/IP stack supports tunnels activated by IKEv1 and IKEv2
  - Each stack supports a wider variety of cryptographic algorithms

In order to be able to include z/OS in bids for government IT projects, the z/OS IPSec function is enhanced to support IKEv2, in addition to its current IKEv1 support. The IKEv2 support for z/OS affects multiple z/OS components. Here are the key features of IKEv2 support.

One IPSec policy file can contain both IKEv1 and IKEv2 policies.

IKE daemon supports both IKEv1 and IKEv2 tunnels simultaneously.

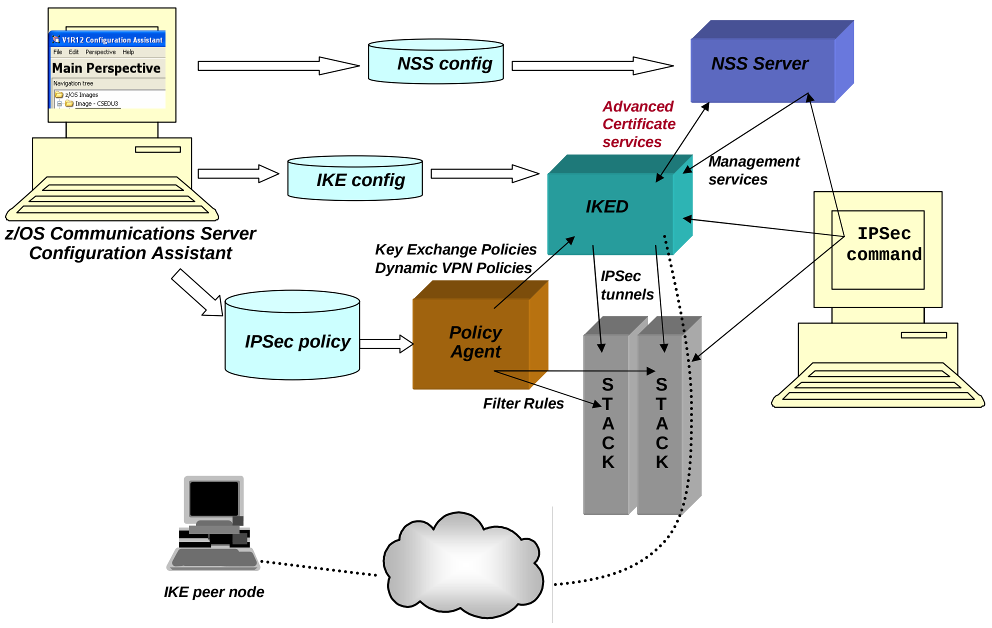
Each TCP/IP stack can support tunnels activated by IKEv1 and IKEv2, and supports a wider variety of cryptographic algorithms.

The IPSec command can display, activate, refresh, and deactivate both IKEv1 and IKEv2 tunnels.

The NSS provides several functions including advanced certificate services required by IKEv2. The NSS also provides HTTP retrieval of certificates and certificate bundles and trust chain and certificate revocation checking.

The NSS is required for IKEv2 tunnels that use any certificate-based authentication method.

Diagram of z/OS processes in IPSec with IKEv2



© 2010 IBM Corporation

Everything on this chart changed for IKEv2, but the overall structure and interaction of the components stayed very much the same.

New IKEv2 configuration options and policies are configured by the z/OS Communications Server Configuration Assistant. They are parsed by the Policy Agent (PAGENT), but the basic nature of the policies (filter rules, key exchange policies, dynamic VPN policies) did not change.

TCP/IP stacks now support IKEv2 tunnels in addition to IKEv1 tunnels. It also supports a wider variety of cryptographic algorithms for hashing and encryption of data.

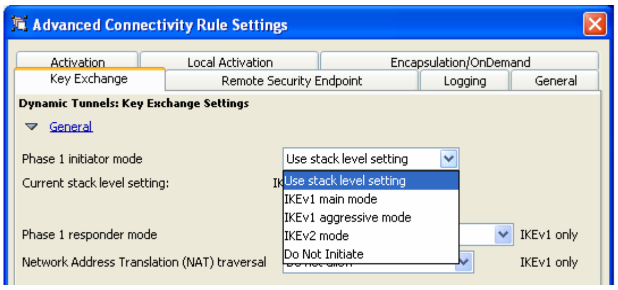
The IKE daemon (IKED) supports the IKEv2 protocol in addition to the IKEv1 protocol, for dynamic management of security associations. However, it has a stronger dependence on an Network Security Services (NSS) Server for certificate services.

The NSS server daemon (NSSD) plays a bigger part for IKEv2, by providing advanced certificate services to the IKE daemons. It now performs HTTP retrieval of certificates and certificate bundles, and trust chain and certificate revocation processing.

The IPSec command allows operator management (display, activation, deactivation, and refresh) of security associations.

Function externals: enabling IKEv2

- The **HowToInitiate** keyword of the KeyExchangeAction statement determines whether IKED will use IKEv1 or IKEv2 when acting as the *initiator* of a tunnel
- IKED will automatically respond to either IKEv1 or IKEv2 activation requests when acting as the responder
  - No additional configuration *required* to support IKEv2 as a responder!
  - The type of IKEv1 response is determined by HowToRespondIKEv1
    - Formerly HowToRespond, renamed to reflect IKEv1-only nature



Dynamic Tunnels: Key Exchange Settings

Phase 1 initiator mode: Use stack level setting
(Use stack level setting / IKEv1 main mode / IKEv1 aggressive mode / IKEv2 mode / Do Not Initiate)

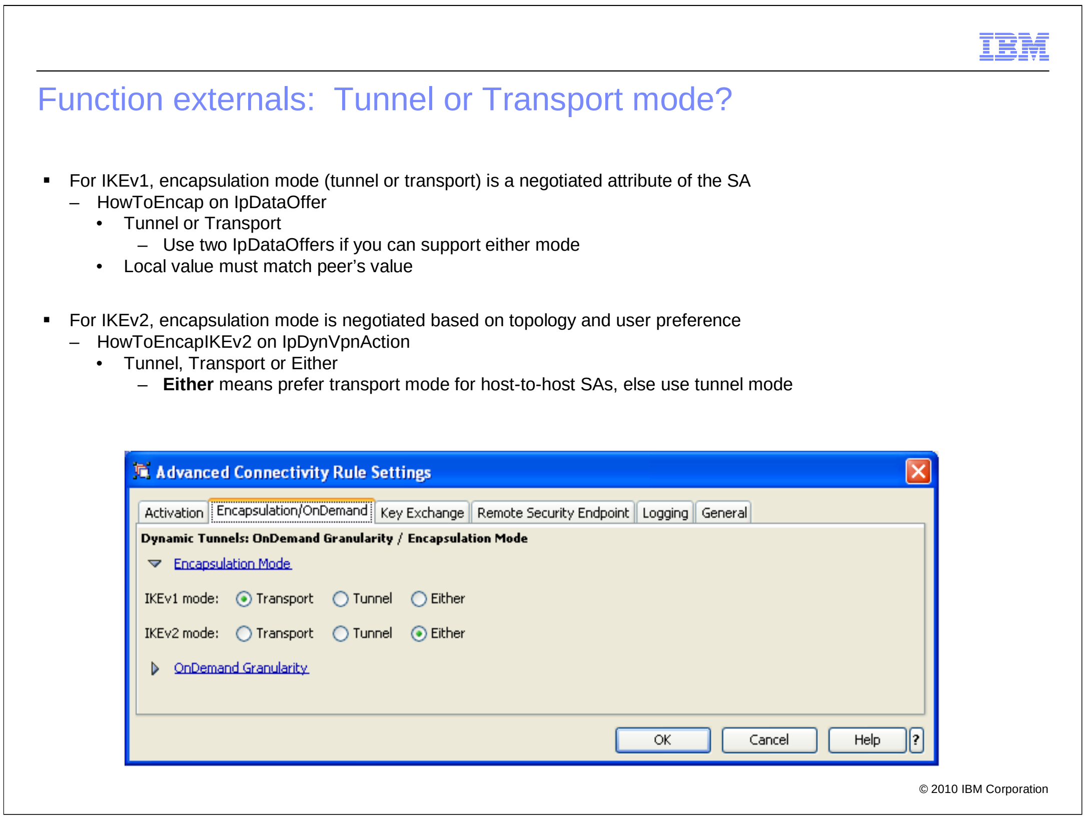
Current stack level setting: IK

Phase 1 responder mode — IKEv1 only

Network Address Translation (NAT) traversal — IKEv1 only

It is easy to enable IKEv2 on z/OS, particularly for users that have already implemented IKEv1. Changing the HowToInitiate keyword value to IKEv2 (or setting Phase 1 initiator mode to IKEv2) will cause z/OS IKED to use IKEv2 flows when initiating the security association. Specify HowToInitiate on KeyExchangePolicy to set the default for the stack.

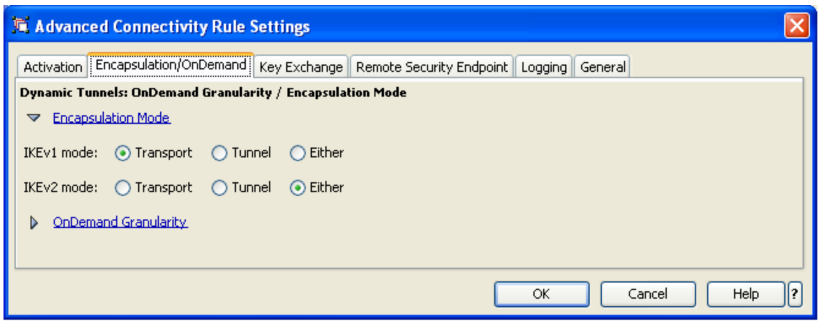
As a responder, z/OS IKED will respond to either IKEv1 or IKEv2 activation requests, so the HowToRespond (or Phase 1 responder mode) setting is used as an IKEv1-only parameter.



Function externals: Tunnel or Transport mode?

- For IKEv1, encapsulation mode (tunnel or transport) is a negotiated attribute of the SA
  - HowToEncap on IpDataOffer
    - Tunnel or Transport
      - Use two IpDataOffers if you can support either mode
    - Local value must match peer’s value

- For IKEv2, encapsulation mode is negotiated based on topology and user preference
  - HowToEncapIKEv2 on IpDynVpnAction
    - Tunnel, Transport or Either
      - **Either** means prefer transport mode for host-to-host SAs, else use tunnel mode



Dynamic Tunnels: OnDemand Granularity / Encapsulation Mode

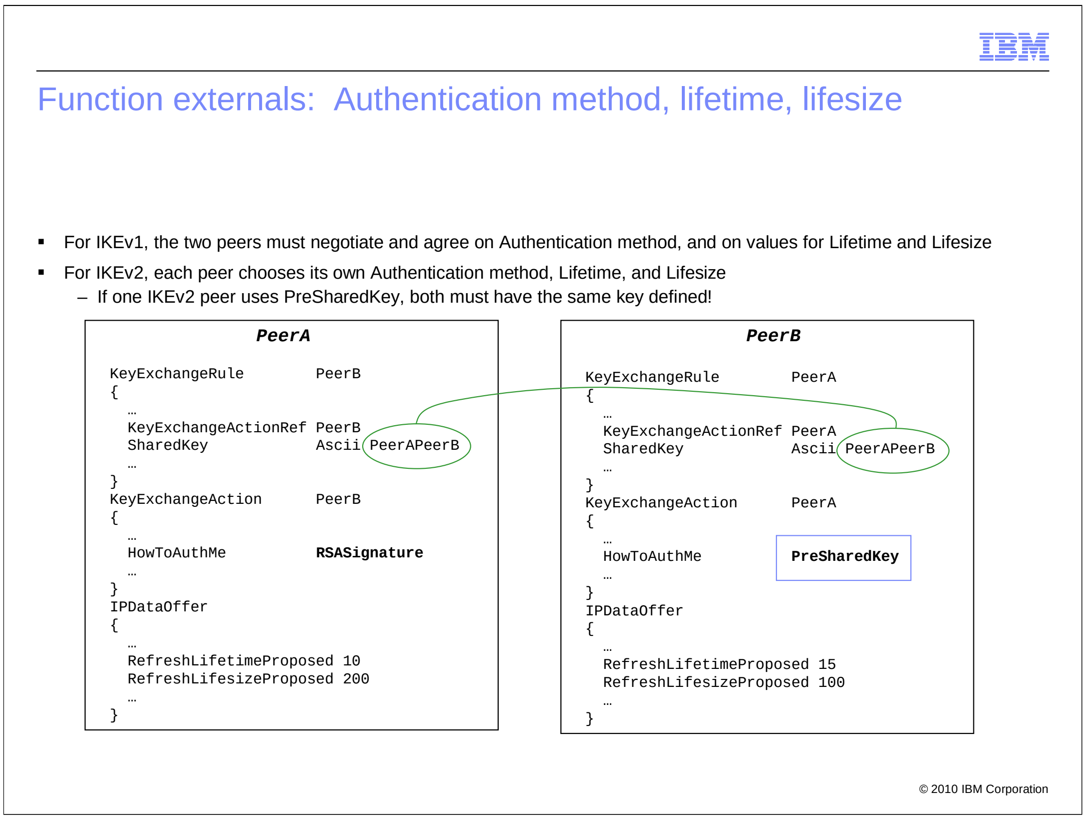
Encapsulation Mode

IKEv1 mode: [✓ Transport] Tunnel Either

IKEv2 mode: Transport Tunnel [✓ Either]

OnDemand Granularity

The encapsulation mode, tunnel or transport, is negotiated differently for IKEv2 than it is for IKEv1, so a separate setting is provided for IKEv2 encapsulation mode than for IKEv1, on the same rule.



Function externals: Authentication method, lifetime, lifesize

- For IKEv1, the two peers must negotiate and agree on Authentication method, and on values for Lifetime and Lifesize
- For IKEv2, each peer chooses its own Authentication method, Lifetime, and Lifesize
  - If one IKEv2 peer uses PreSharedKey, both must have the same key defined!

PeerA

```
KeyExchangeRule        PeerB
{
  …
  KeyExchangeActionRef PeerB
  SharedKey            Ascii PeerAPeerB
  …
}
KeyExchangeAction      PeerB
{
  …
  HowToAuthMe          RSASignature
  …
}
IPDataOffer
{
  …
  RefreshLifetimeProposed 10
  RefreshLifesizeProposed 200
  …
}
```

PeerB

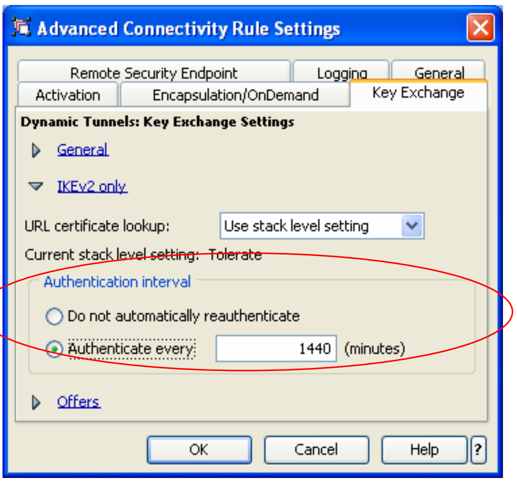
```
KeyExchangeRule        PeerA
{
  …
  KeyExchangeActionRef PeerA
  SharedKey            Ascii PeerAPeerB
  …
}
KeyExchangeAction      PeerA
{
  …
  HowToAuthMe          PreSharedKey
  …
}
IPDataOffer
{
  …
  RefreshLifetimeProposed 15
  RefreshLifesizeProposed 100
  …
}
```

For IKEv1, the authentication method, lifetime, and lifesize of a Security Association (SA) are negotiated, and both peer nodes must agree on the values. For IKEv2, each peer decides on its own authentication method, lifetime and lifesize. In this example, PeerA is authenticated to PeerB using RSA signature authentication, but PeerB is authenticated to PeerA using a PreShared key. Note that in order for PeerA to perform PreShared key authentication of PeerB, it must have PeerB's key defined.

Also note that PeerA is using a shorter Lifetime than PeerB, and PeerB is using a shorter Lifesize than PeerA. If the traffic flowing on the tunnel is light, PeerA might refresh the tunnel due to Lifetime expiration before PeerB has reached its Lifesize.

Function externals: IKEv2 rekey without reauthentication

- For IKEv1, refreshing a tunnel includes rekeying the tunnel and reauthenticating the peer.
  - Reauthenticating is expensive!
- IKEv2 supports rekeying a tunnel *without* reauthenticating the peer
  - For IKEv2, lifetime or lifesize expiration will cause a rekey only
    - New parameter **ReauthInterval** on KeyExchangeAction will periodically cause a rekey and reauthentication for IKEv2
  - “ipsec –k refresh” will cause rekey and reauthentication



Dynamic Tunnels: Key Exchange Settings

General

IKEv2 only

URL certificate lookup: Use stack level setting

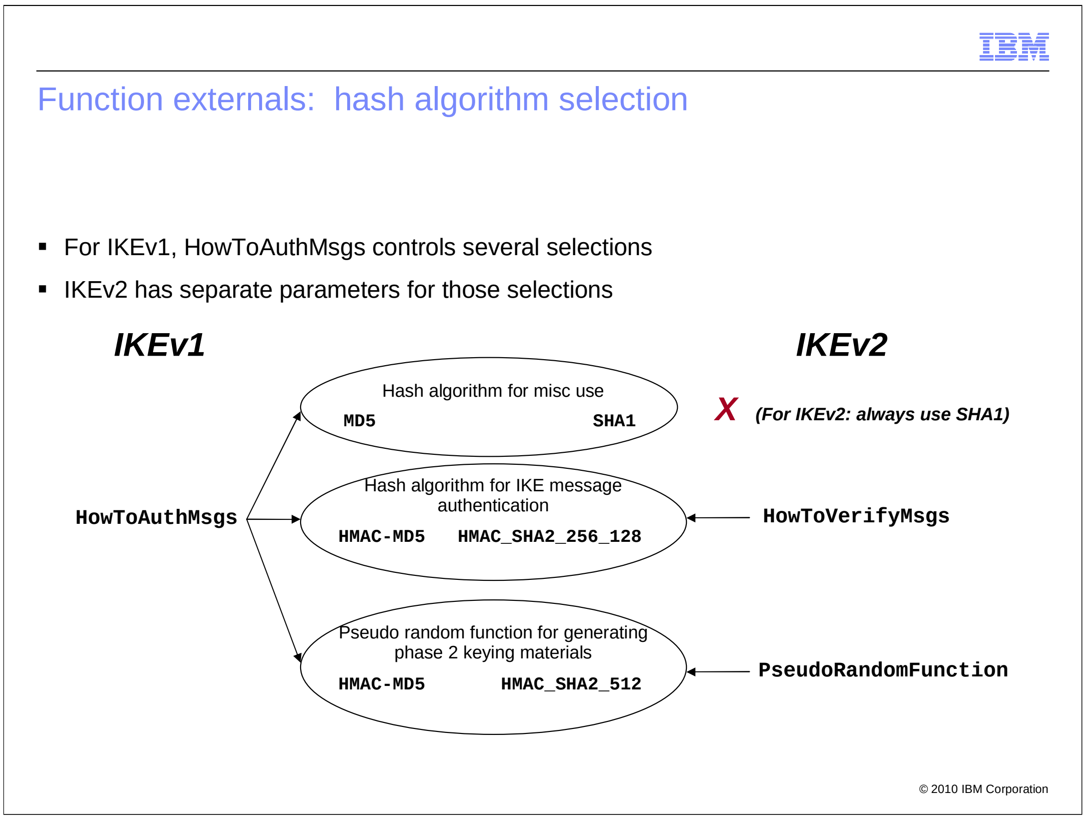
Current stack level setting: Tolerate

Authentication interval

Do not automatically reauthenticate [✓ Authenticate every: 1440 (minutes)]

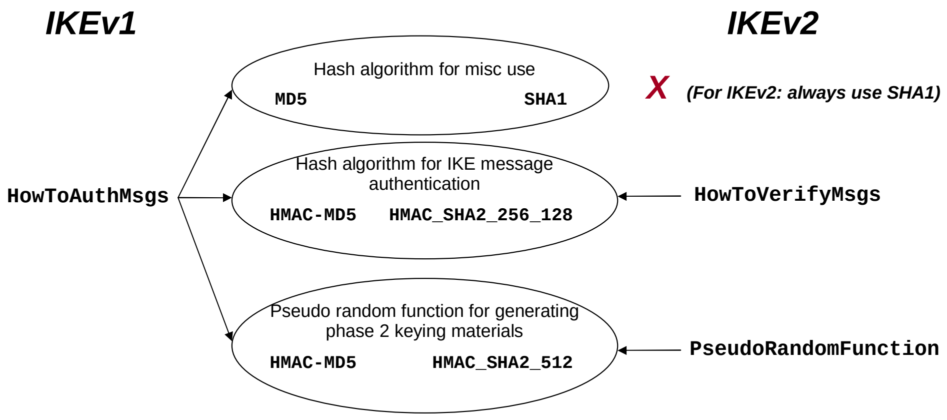
Offers

In IKEv1, every refresh of a tunnel includes rekeying the tunnel and reauthenticating the peer node. Reauthentication is a cycle-intensive process, so the IKEv2 architecture was defined to support rekeying of a tunnel without requiring reauthentication, when lifetime or lifesize expires. z/OS users can force reauthentication using the ipsec command. A new parameter was introduced for IKEv2 to allow z/OS users a way to automatically reauthenticate after a user-chosen interval.

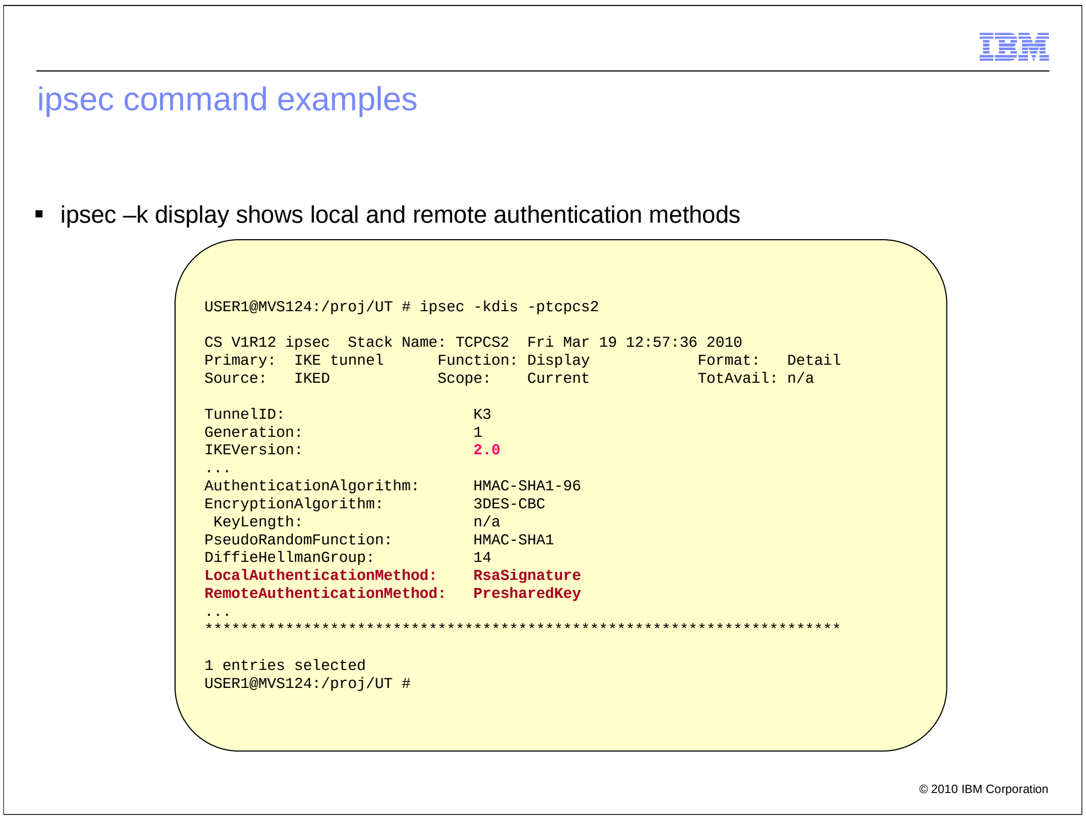


Function externals: hash algorithm selection

- For IKEv1, HowToAuthMsgs controls several selections
- IKEv2 has separate parameters for those selections


IKEv1
IKEv2
Hash algorithm for misc use
MD5
SHA1
X
(For IKEv2: always use SHA1)
HowToAuthMsgs
Hash algorithm for IKE message authentication
HMAC-MD5
HMAC_SHA2_256_128
HowToVerifyMsgs
Pseudo random function for generating phase 2 keying materials
HMAC-MD5
HMAC_SHA2_512
PseudoRandomFunction

For IKEv1, a single policy setting determines the hash algorithm and its HMAC-ed variants that are used for several purposes. IKEv2 is designed to allow you to select independent algorithms for those purposes.



ipsec command examples

- ipsec –k display shows local and remote authentication methods

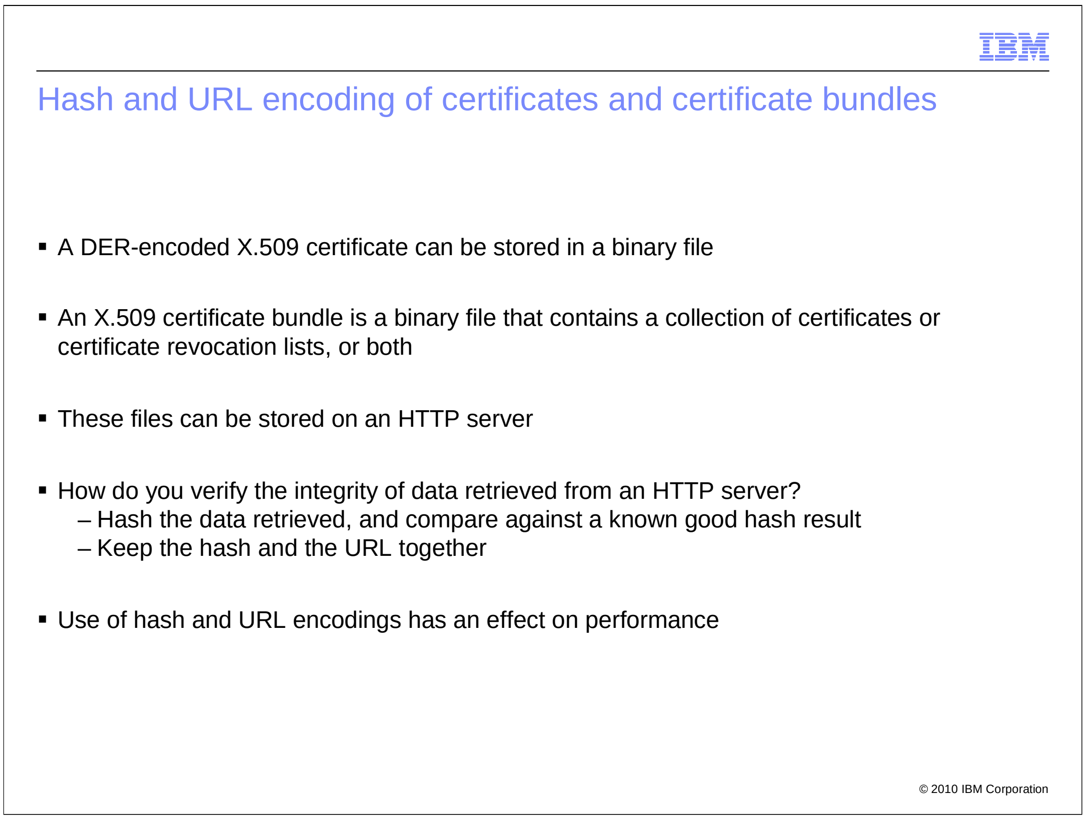
```
USER1@MVS124:/proj/UT # ipsec -kdis -ptcpcs2

CS V1R12 ipsec  Stack Name: TCPCS2  Fri Mar 19 12:57:36 2010
Primary:  IKE tunnel      Function: Display            Format:   Detail
Source:   IKED            Scope:    Current            TotAvail: n/a

TunnelID:                      K3
Generation:                    1
IKEVersion:                    2.0
...
AuthenticationAlgorithm:       HMAC-SHA1-96
EncryptionAlgorithm:           3DES-CBC
 KeyLength:                    n/a
PseudoRandomFunction:          HMAC-SHA1
DiffieHellmanGroup:            14
LocalAuthenticationMethod:     RsaSignature
RemoteAuthenticationMethod:    PresharedKey
...
***********************************************************************

1 entries selected
USER1@MVS124:/proj/UT #
```

IKEv2 allows the local and remote peers to use different authentication methods. Both methods are shown in the ipsec –k display.



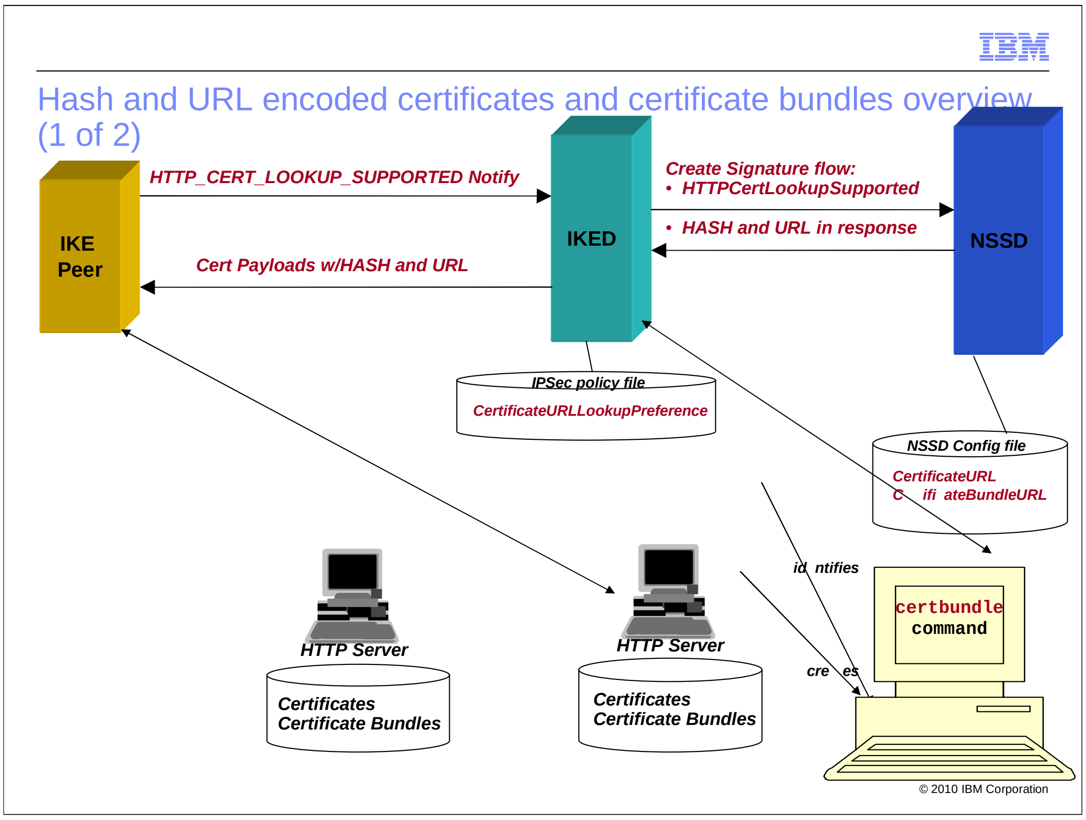
Hash and URL encoding of certificates and certificate bundles

- A DER-encoded X.509 certificate can be stored in a binary file
- An X.509 certificate bundle is a binary file that contains a collection of certificates or certificate revocation lists, or both
- These files can be stored on an HTTP server
- How do you verify the integrity of data retrieved from an HTTP server?
  - Hash the data retrieved, and compare against a known good hash result
  - Keep the hash and the URL together
- Use of hash and URL encodings has an effect on performance

A X.509 certificate encoded using the Distinguished Encoding Rules (DER) can be stored in a binary file. An X.509 certificate bundle is a binary file that contains a collection of certificates or certificate revocation lists, or both.

These files can be stored on an HTTP server, and they can be retrieved using HTTP protocols by anyone who knows the URL and has network access to the HTTP server. The question is, how do you verify the integrity of the data retrieved from an HTTP server? The common way of authenticating data is to hash the data retrieved, and compare against a known good hash result.

IKEv2 architecture uses certificates for digital signature authentication, like IKEv1 does. However, IKEv2 allows Hash and URL encoding of certificates, while IKEv1 does not. Use of hash and URL encodings can reduce the size of IKEv2 messages, but has additional overhead of retrieval of the certificates from the HTTP server. IKEv2 peers indicate their support (and preference) for hash and URL encodings by sending a notify payload of type HTTP_CERT_LOOKUP_SUPPORTED.

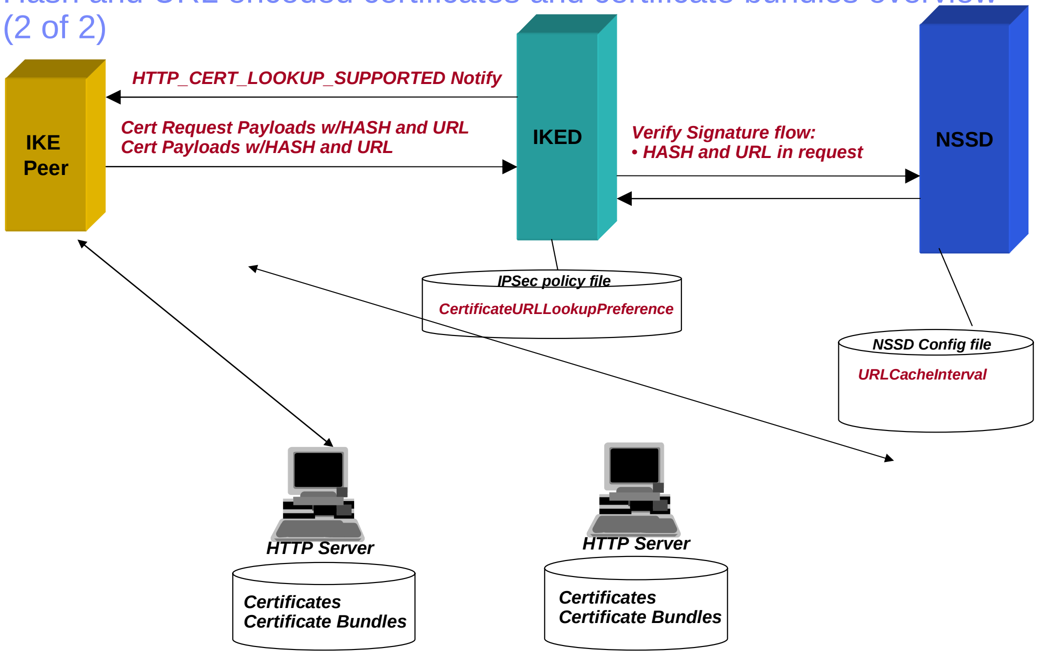


The IPSec policy file contains a new configuration option that controls whether IKED will try to send hash and URL encoded data to peers. Under certain conditions, IKED will inform NSS that it is valid to use the hash and URL certificate encoding types in the certificate payloads that is sent to the peer node along with the created signature. These conditions are if IKED is configured to try and if the IKE peer sends an HTTP_CERT_LOOKUP_SUPPORTED notify.

NSSD chooses the certificate to use to create the signature and chooses supporting certificates to send with the signature. The NSSD configuration file contains new definitions that associate URLs to certificates on the NSSD key ring. The certbundle command is a new command that will create certificate bundle files that can be placed on an HTTP server. The NSS daemon performs all HTTP retrieval of certificates and certificate bundles using the URLs. For create signature processing, it retrieves locally stored certificates and bundles so it can calculate the hash which is sent with the URL to the peer.

The peer retrieves the certificates and bundles using the URLs, and verifies the received data using the associated hashes.

Hash and URL encoded certificates and certificate bundles overview (2 of 2)



That same new IPSec policy option controls whether IKED will allow a peer to send hash and URL encoded data to z/OS. If it is allowed, IKED will send an HTTP_CERT_LOOKUP_SUPPORTED notify to that peer. As a result, the peer can send Certificate Request and Certificate payloads encoded using hash and URL encoding types.

IKED will send the send hash and URL encoded certificate data to NSSD in a verify signature request. NSSD performs all HTTP retrieval of certificates and certificate bundles using the URLs sent from the peer node.

NSSD can be configured to cache the retrieved data in memory, to reduce the frequency of HTTP retrievals. Caching of the URLs and the associated retrieved data is controlled by another NSSD configuration file option: URLCacheInterval.

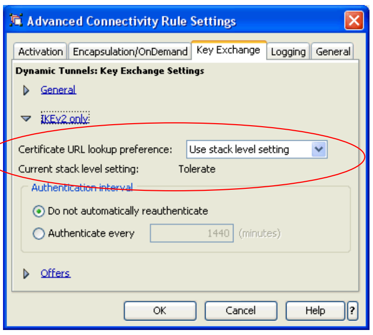
IKED and NSSD can send and receive hash and URL encoded certificates and bundles in both directions to each IKE peer.

Function externals: Enabling or disabling HTTP lookup

- **CertificateURLLookupPreference** keyword of KeyExchangeAction

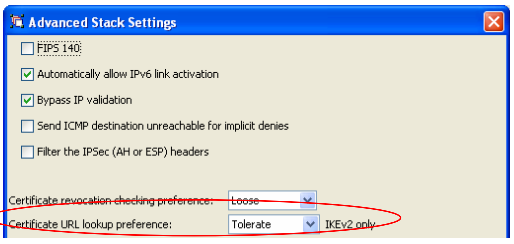
– *Allow*	send and request URLs
– *Tolerate*	send URLs, don't request
– *Disallow*	do not send or request

- Also available on KeyExchangePolicy
  - Sets the stack level default



Certificate URL lookup preference: Use stack level setting
Current stack level setting: Tolerate

Authentication interval
[✓ Do not automatically reauthenticate]
Authenticate every 1440 (minutes)



☐ FIPS 140
☑ Automatically allow IPv6 link activation
☑ Bypass IP validation
☐ Send ICMP destination unreachable for implicit denies
☐ Filter the IPSec (AH or ESP) headers

Certificate revocation checking preference: Loose
Certificate URL lookup preference: Tolerate IKEv2 only

To enable HTTP lookup of certificates and bundles, code the **CertificateURLLookupPreference** keyword of the KeyExchangeAction or KeyExchangePolicy.

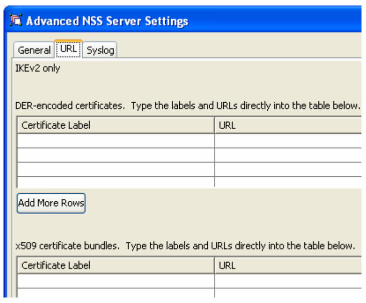
If you code Allow, that means IKED will send hash and URL encoded certificate and certificate request payloads to peer nodes, and will request that peer nodes use hash and URL encoding when sending to IKED.

If you code Tolerate, then IKED will send hash and URL encoded certificate and certificate request payloads to peer nodes, but will prevent peer nodes from sending hash and URL encodings in its requests to IKED. This is the default. It gives the system administrator the greatest control over which HTTP servers z/OS will attempt retrieval from, by limiting HTTP retrieval attempts to those URLs that are encoded in the local NSSD configuration file. See the next chart for details.

The CertificateURLLookupPreference option can be specified on a per-IKE-peer basis, and the stack-wide default value can be specified for each stack. These options are also available through the Configuration Assistant.

Function externals: NSSD configuration file changes

- **CertificateURL** and **CertificateBundleURL** statements
  - Define the label of a certificate on the key ring
  - Define the corresponding URL of the certificate or bundle file to be used by peer nodes for HTTP retrieval
- **URLCacheInterval** statement
  - Defines the number of minutes that retrieved URL data is cached
    - Default: 1 week
    - 0 means do not cache
- All of these statements can be modified dynamically
  - MODIFY *nssd*,REFRESH


Advanced NSS Server Settings

General | URL | Syslog

IKEv2 only

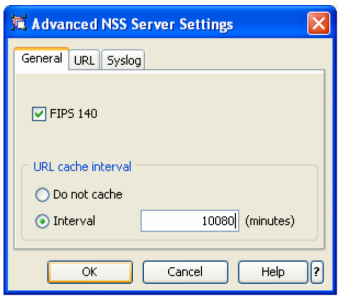
DER-encoded certificates. Type the labels and URLs directly into the table below.

Certificate Label	URL

Add More Rows

x509 certificate bundles. Type the labels and URLs directly into the table below.

Certificate Label	URL


Advanced NSS Server Settings

General | URL | Syslog

[✓] FIPS 140

URL cache interval

( ) Do not cache

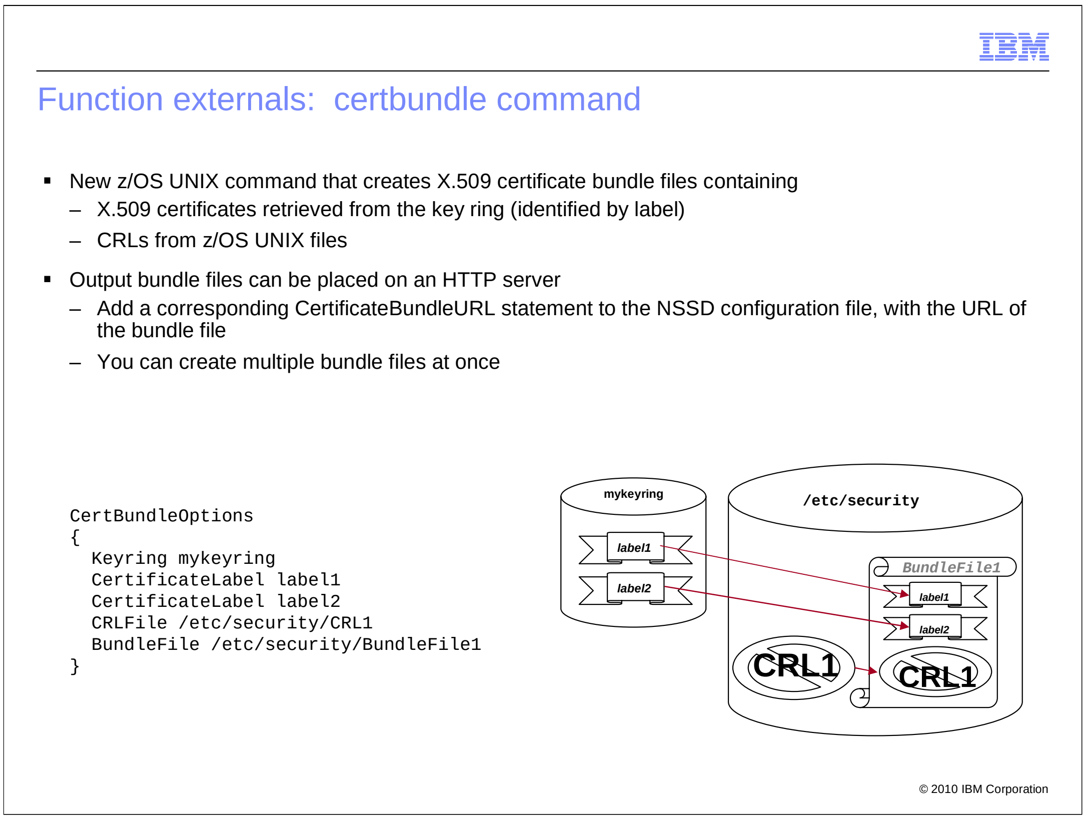
[✓ Interval] 10080 (minutes)

You define the URLs associated with certificates or bundles in the NSSD configuration file. These define the URLs that are sent back to peer nodes that request hash and URL encoded certificates.

You also control caching of URL data in the NSSD configuration file, using the URLCacheInterval statement. By default, NSSD will cache retrieved URL data for one week. You can disable caching completely by coding 0 on the URLCacheInterval statement.

Examples of configuration of these options using the Configuration Assistant are also shown.

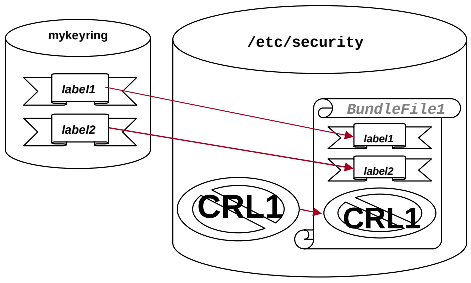
These options can be modified dynamically using the MODIFY REFRESH command.



Function externals: certbundle command

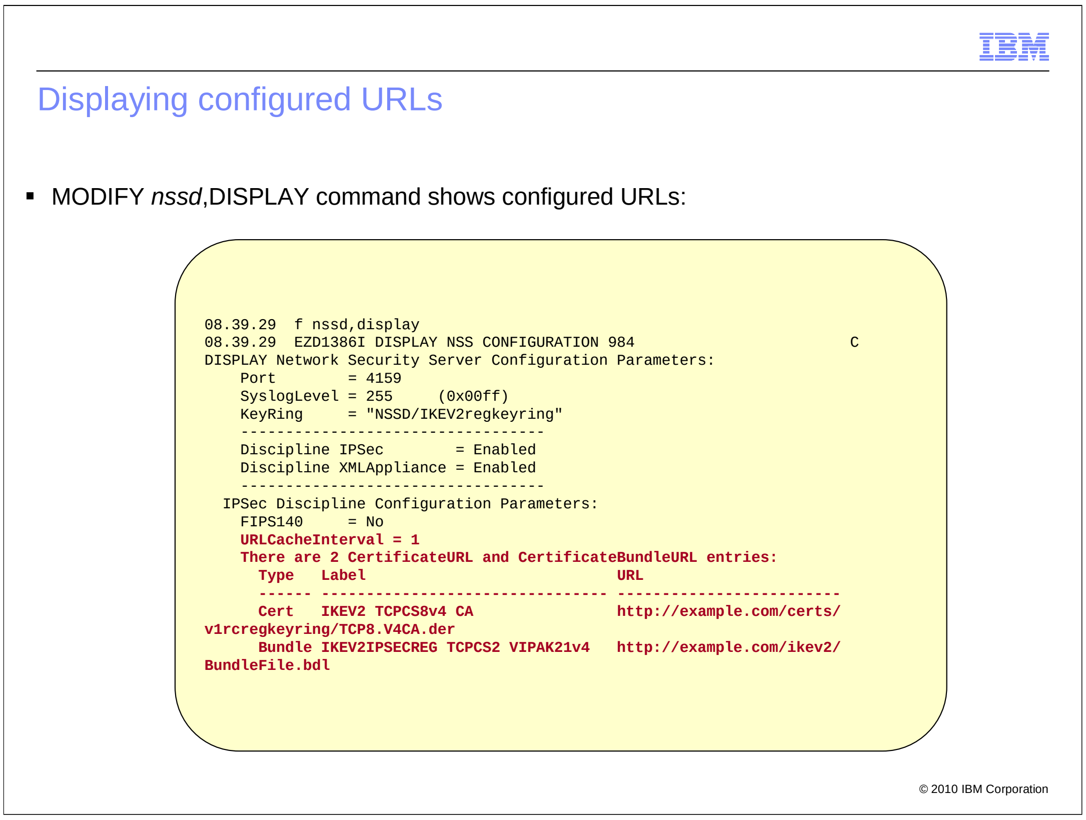
- New z/OS UNIX command that creates X.509 certificate bundle files containing
  – X.509 certificates retrieved from the key ring (identified by label)
  – CRLs from z/OS UNIX files
- Output bundle files can be placed on an HTTP server
  – Add a corresponding CertificateBundleURL statement to the NSSD configuration file, with the URL of the bundle file
  – You can create multiple bundle files at once

```
CertBundleOptions
{
  Keyring mykeyring
  CertificateLabel label1
  CertificateLabel label2
  CRLFile /etc/security/CRL1
  BundleFile /etc/security/BundleFile1
}
```



The **certbundle** command is a new z/OS UNIX command that allows you to create certificate bundles. The certificates to be included in the bundle must be in a SAF key ring, and the Certificate Revocation Lists (CRLs) to be included must be on the z/OS UNIX file system.

Create a text file containing one or more CertBundleOptions statements, describing where to get the input certificates and CRLs, and where to place the output bundle. An example is given on this chart. Then invoke the certbundle command, passing as input the name of the options text file. The certbundle command reads the options from the text file, and creates the output certificate bundle files, extracting the certificates from the specified key rings, and inserting the CRLs.

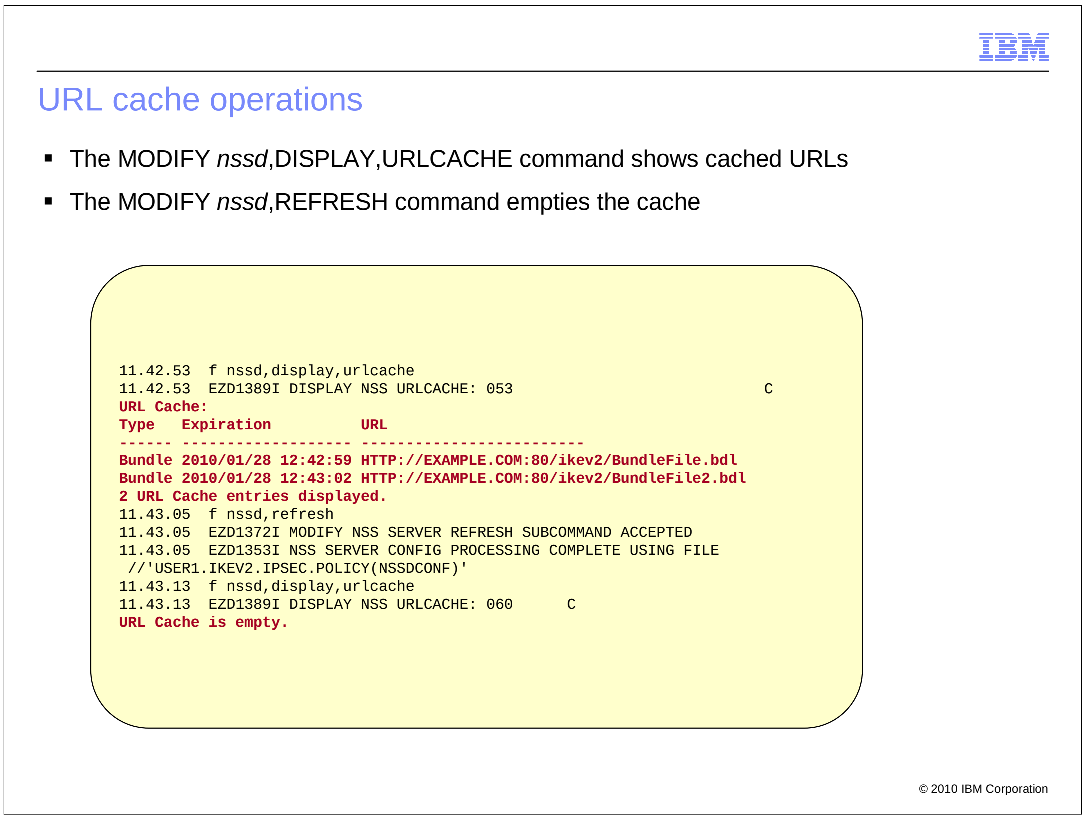


Displaying configured URLs

- MODIFY *nssd*,DISPLAY command shows configured URLs:

```
08.39.29  f nssd,display
08.39.29  EZD1386I DISPLAY NSS CONFIGURATION 984                        C
DISPLAY Network Security Server Configuration Parameters:
    Port        = 4159
    SyslogLevel = 255     (0x00ff)
    KeyRing     = "NSSD/IKEV2regkeyring"
    ----------------------------------
    Discipline IPSec        = Enabled
    Discipline XMLAppliance = Enabled
    ----------------------------------
  IPSec Discipline Configuration Parameters:
    FIPS140     = No
    URLCacheInterval = 1
    There are 2 CertificateURL and CertificateBundleURL entries:
      Type   Label                            URL
      ------ -------------------------------- -------------------------
      Cert   IKEV2 TCPCS8v4 CA                http://example.com/certs/
v1rcregkeyring/TCP8.V4CA.der
      Bundle IKEV2IPSECREG TCPCS2 VIPAK21v4   http://example.com/ikev2/
BundleFile.bdl
```

The MODIFY *nssd*,DISPLAY command allows users to display the values coded in the NSSD configuration file. New data on this display includes the values coded on CertificateURL and CertificateBundleURL statements. The key ring labels and the URLs are quite wide, so it is likely that the data will wrap, especially if you use an 80-column console.



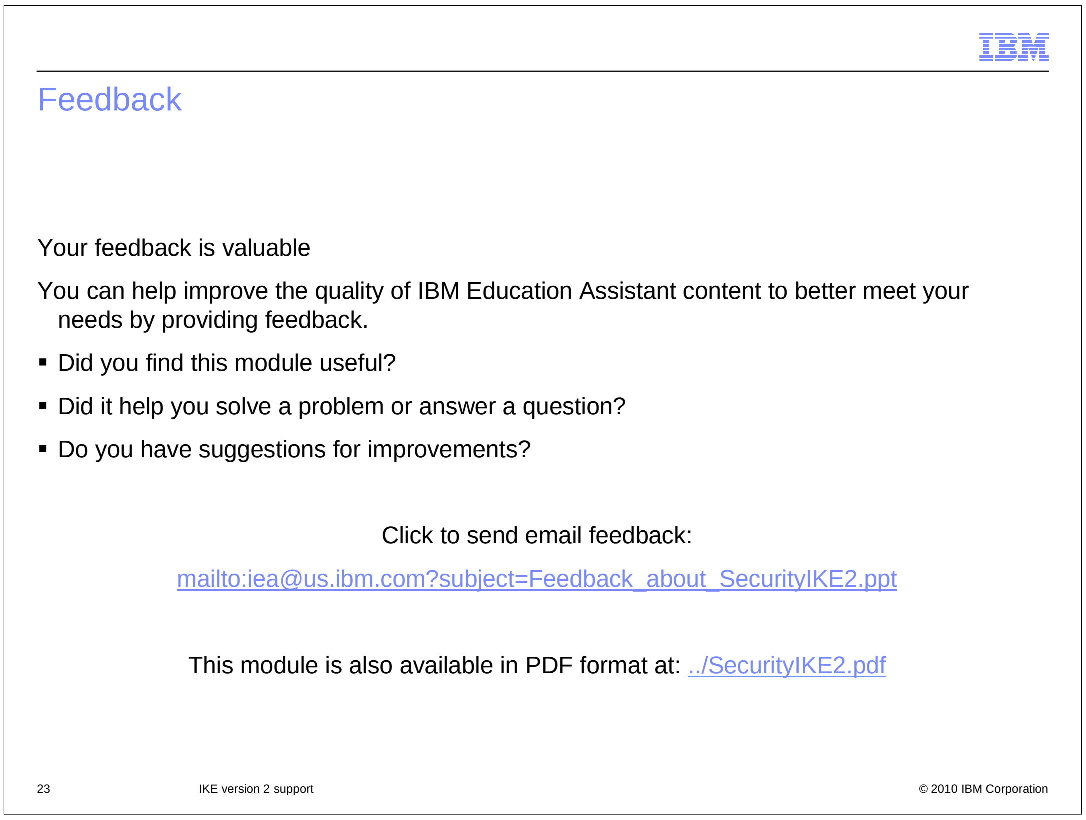
URL cache operations

- The MODIFY *nssd*,DISPLAY,URLCACHE command shows cached URLs
- The MODIFY *nssd*,REFRESH command empties the cache

```
11.42.53  f nssd,display,urlcache
11.42.53  EZD1389I DISPLAY NSS URLCACHE: 053                            C
URL Cache:
Type   Expiration          URL
------ ------------------- ------------------------
Bundle 2010/01/28 12:42:59 HTTP://EXAMPLE.COM:80/ikev2/BundleFile.bdl
Bundle 2010/01/28 12:43:02 HTTP://EXAMPLE.COM:80/ikev2/BundleFile2.bdl
2 URL Cache entries displayed.
11.43.05  f nssd,refresh
11.43.05  EZD1372I MODIFY NSS SERVER REFRESH SUBCOMMAND ACCEPTED
11.43.05  EZD1353I NSS SERVER CONFIG PROCESSING COMPLETE USING FILE
 //'USER1.IKEV2.IPSEC.POLICY(NSSDCONF)'
11.43.13  f nssd,display,urlcache
11.43.13  EZD1389I DISPLAY NSS URLCACHE: 060      C
URL Cache is empty.
```

A new option on the MODIFY *nssd*,DISPLAY command, URLCACHE, allows you to display the URLs that have been retrieved and cached by NSSD since it started, and the expiration date and time of each cache entry. The URLs are quite wide, so it is likely that the data will wrap, especially if you use an 80-column console.

The MODIFY *nssd*,REFRESH command empties the cache.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

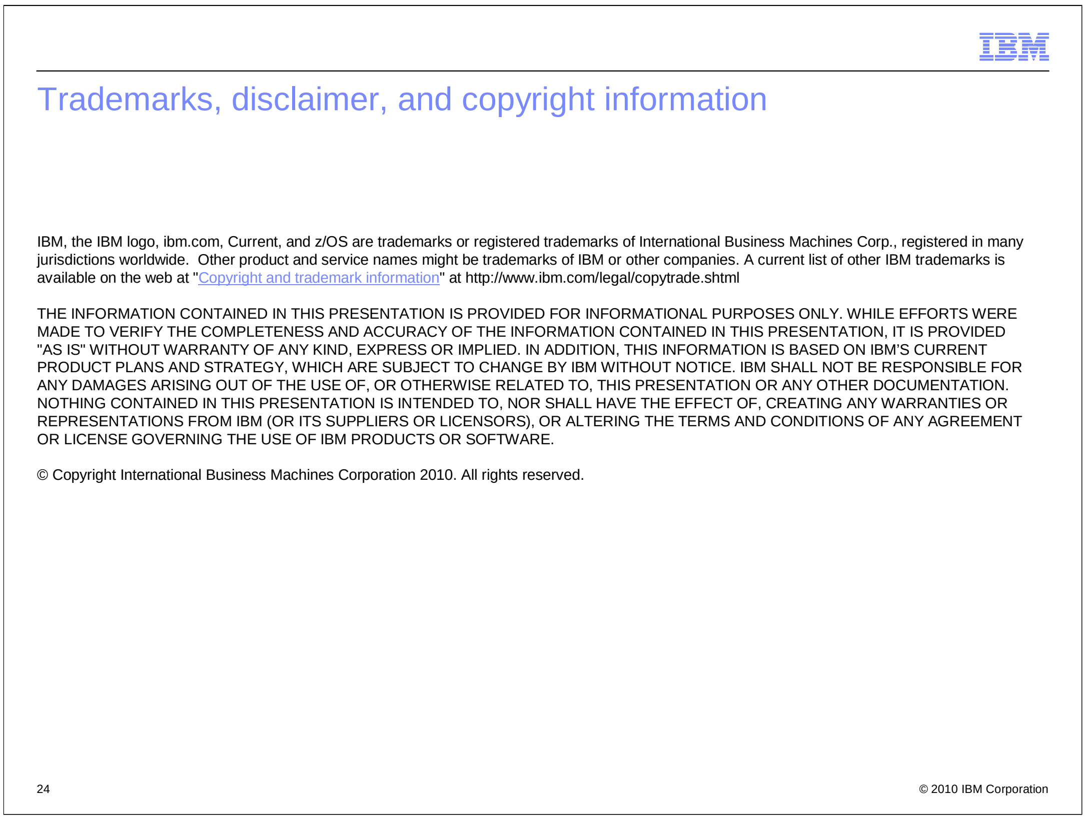
Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_SecurityIKE2.ppt

This module is also available in PDF format at: ../SecurityIKE2.pdf

23 IKE version 2 support © 2010 IBM Corporation

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, disclaimer, and copyright information