TPF Users Group Spring 2006

Secure Key Management and Data Privacy on z/TPF

Name : Mark Gambino
Venue : Communications Subcommittee

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Agenda

- Why are we here?
- What support exists now
- New requirements that have come up
- Proposed solution
  - Statement of direction - details are subject to change
- What you can do now regarding application design and database design

Why Cryptography? To Protect the Data

- Data in flight (flowing across the network)
  - Data flowing across public networks
  - Even some data flowing in your private network
- Data at rest
  - Data in the TPF database
  - Data on tape
- Data in use
  - Data being used by an application program while processing the transaction

What Are My Options Today?

- Data in flight
  - Use Secure Sockets Layer (SSL)
  - Encrypt data at the application layer
    - When using private protocols or middleware that does not have encryption built in
- Data at rest
  - Encrypt data at the application layer
- Data in use
  - User modifications to prevent certain data from being displayed or included in dumps

Existing Support on TPF

- SSL
  - RC4, DES, Triple-DES (TDES) algorithms for data encryption
  - MD5 and SHA-1 hash algorithms for data integrity
  - Hardware acceleration for RSA operations
    - Allows you to start thousands of SSL sessions per second
  - Hardware acceleration for DES, TDES, and SHA-1
    - Uses Central Processor Assist for Cryptographic Functions (CPACF)
    - Allows you to send/receive tens of thousands of messages per second across SSL sessions
- User data encryption
  - APIs that allow you to encrypt/decrypt data using DES or TDES
  - Uses CPACF if installed on the processor
    - Can scale up to hundreds of thousands of crypto operations per second

Advanced Encryption Standard (AES)

- New crypto algorithm designed to be the replacement for DES
- How secure is AES?
  - U.S. government uses AES to protect classified information up to *"TOP SECRET"* level
- CPACF on System z9 added support for AES
- z/TPF plans to:
  - Add support for AES to SSL
  - Add APIs to allow applications to encrypt/decrypt data using AES
  - Use hardware acceleration for AES if available

New Regulations and Requirements

- Today, key management is responsibility of the user/application
  - Customers would like TPF to provide key management functions:
    - Create keys, change keys, store keys, and so on
  - Customers need secure key management:
    - Protect the keys from applications, operators, programmers, coverage staff, and so on
- Customers would like to be able to detect if data has been corrupted
  - Corrupted by accident or intentionally
- Sensitive data is unencrypted (in the clear) while being processed by an application
  - Customers would like the ability to prevent this data from being displayed, including being displayed in dumps
- New regulations, internal audit policies, and business partner agreements are requiring that more and more data be encrypted
  - The amount of data that requires encryption will continue to increase

Secure Key Management

- PCIXCC on z990 and Crypto Express2 coprocessor (CEX2C) on z9 are secure crypto cards (keys reside in the crypto cards)
- Problem:
  - Throughput on secure crypto cards is limited
    - For example, using TDES and 4K data size, one PCIXCC can do only 450 operations per second
- Solution:
  - z/TPF plans to continue to use CPACF for performance reasons
    - Using TDES and 4K data size, each CPACF on a z990 can do 28,760 operations per second
  - z/TPF plans to create a *Key Store* where crypto keys will reside
    - The Key Store will exist in "hidden memory" in core
    - Encrypted copy of the Key Store will exist in the TPF database
    - New operator command to create a key used only by TPF
    - Ability to import keys from a remote Key Repository or Key Manager for keys shared by TPF and a remote node

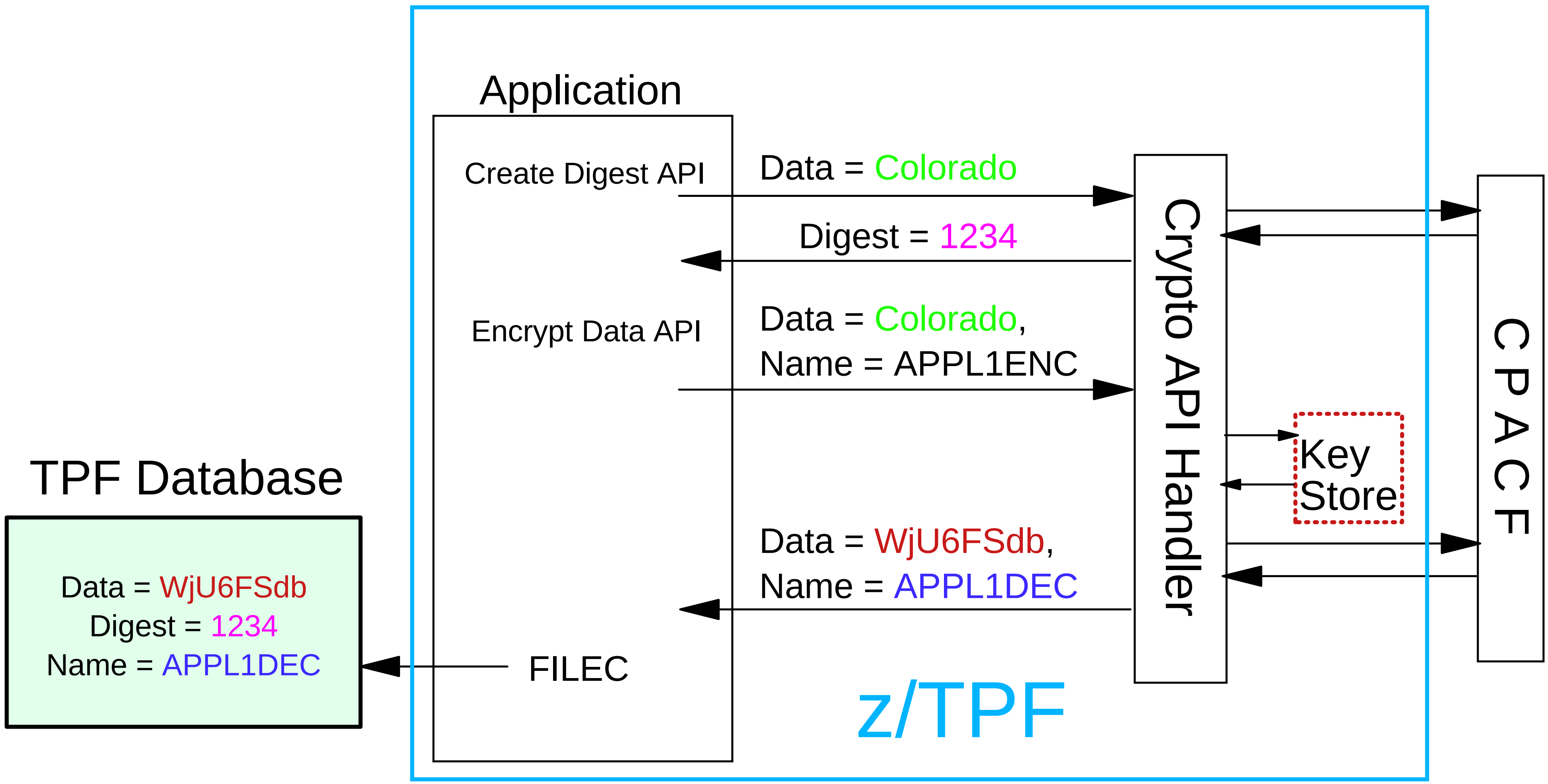
Data Encryption by Applications

- z/TPF plans to create new APIs to encrypt/decrypt data using keys in the TPF Key Store
- Applications reference a key by its token/name
  - Applications only know the name of the key, not the value of the key itself or what cipher (like DES, TDES) is associated with the key
- Applications pass the key name as input to the "encrypt data" API
  - Encryption name remains constant and can be hardcoded into the application program
  - Returns the key name to use to decrypt the data
    - Application needs to save the decryption key name in the record along with the encrypted data
    - Application program uses the saved decryption key name in the record to decrypt the data
- You can change the key value (or even the cipher) without having to modify the application programs

Data Integrity

- Secure Hash Algorithm (SHA)
  - One-way hash algorithm that takes variable length data as input and produces a fixed length digest as output
- To detect data corruption (accidental or malicious):
  1. Create a digest of the regular (unencrypted) data
  2. Save a copy of the digest along with the encrypted data
  3. Recalculate the digest after decrypting the data and compare this digest to the saved digest value
- z/TPF plans to:
  - Create new APIs allowing application programs to create and verify digests
  - Support both SHA-1 and SHA-256
  - Use CPACF to do the hash operation

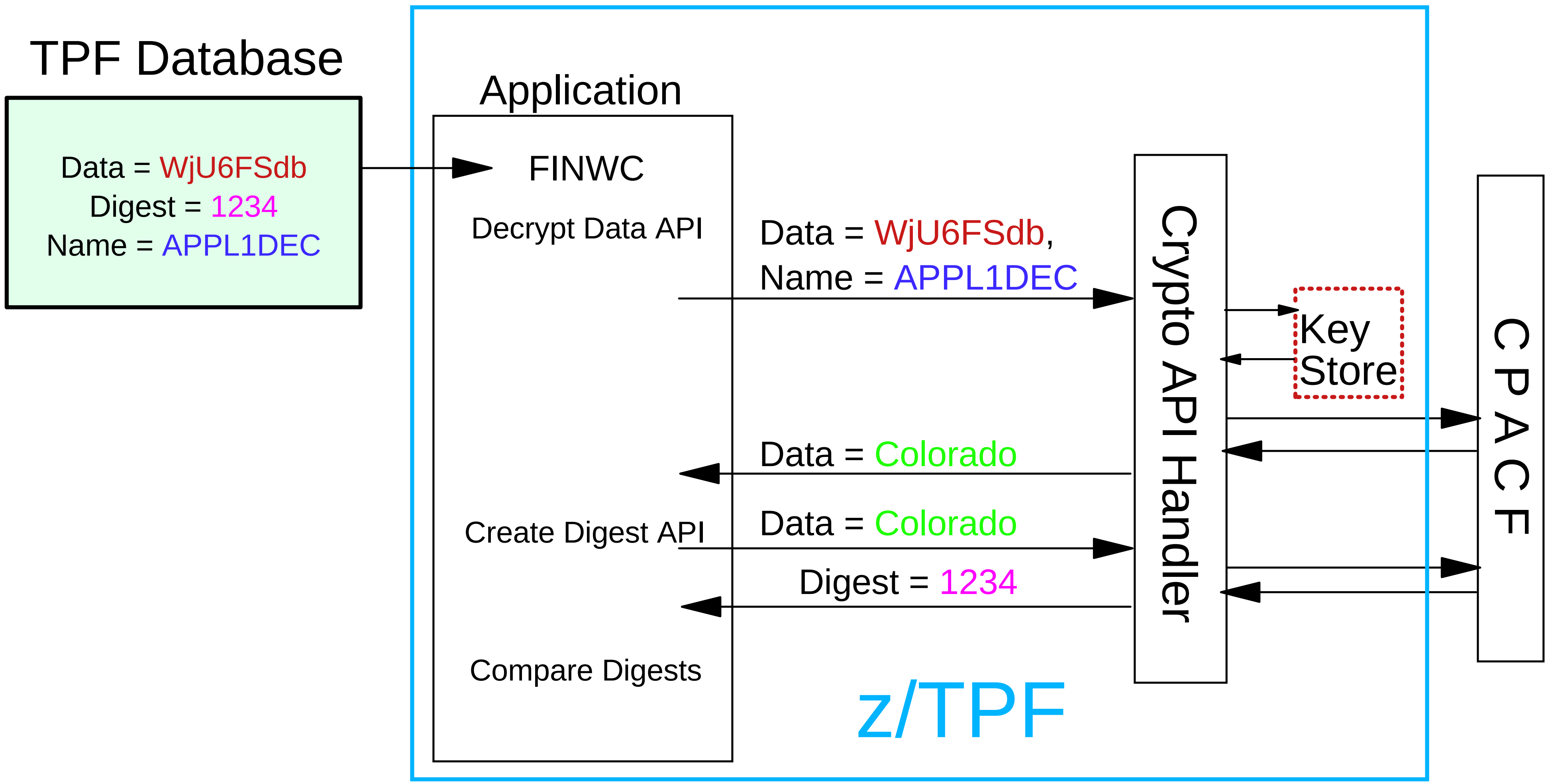
Encrypting Data in the TPF Database Example



Encrypting Data Steps

1. Application issues the "Create Digest" API to create a digest of the plain text (unencrypted data). CPACF is called to create the digest value.
2. Application issues the "Encrypt Data" API to encrypt the data passing the encryption key name of APPL1ENC as input.
3. TPF looks up key name APPL1ENC in the Key Store to find the key and cipher associated with this name, then calls CPACF to encrypt the data.
4. The output of the "Encrypt Data" API is the decryption key name (APPL1DEC) to use to decrypt this data.
5. The application program files the record containing the encrypted data, digest value, and the decryption key name.

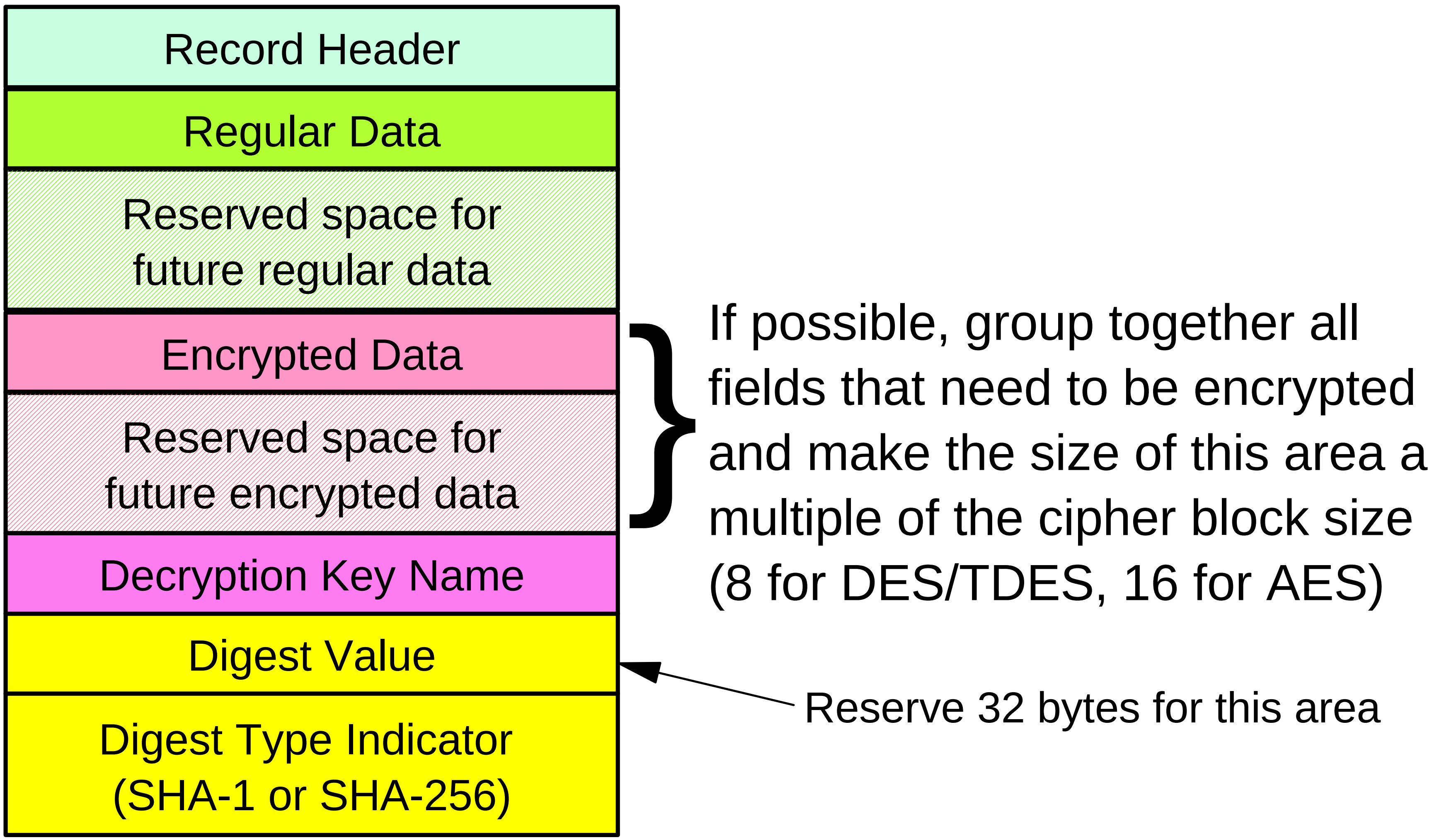
Decrypting Data in the TPF Database Example



Decrypting Data Steps

1. The application program reads the record from file.
2. Application issues the "Decrypt Data" API to decrypt the data passing the decryption key name (APPL1DEC) that was saved in the record.
3. TPF looks up key name APPL1DEC in the Key Store to find the key and cipher associated with this name, then calls CPACF to decrypt the data.
4. Application issues the "Create Digest" API to create a digest of the decrypted data. CPACF is called to create the digest value.
5. The application compares the digest calculated in step 4 to the digest saved in the record. The digest values are equal; therefore, the data in the file record is valid and can be processed.

Suggested Format of a Record in the TPF Database



Additional Items

- z/TPF plans to support IBM hardware tape drive encryption devices
- z/TPF plans to add capability for applications to mark certain core blocks and ECB heap areas as "private" such that their contents are not displayable (including in dumps)
  - For example, decrypted data being used by an application program

Statement of Direction Summary

- Add support for tape drive hardware encryption devices
- Provide secure key management capabilities
- Add APIs to enable applications to encrypt/decrypt data using secure keys
- Continue to support a high rate of crypto operations
  - For example, over 100,000 TDES operations per second on 4K of data on a single z/TPF processor
- Add support for the Advanced Encryption Standard (AES) algorithm
  - For use by SSL and user data encryption directly by applications
- Add APIs that enable application programs to detect if data has been altered by creating/verifying digests
  - Support both SHA-1 and SHA-256 digest/hash algorithms
- Add capability for applications to protect sensitive data in memory

Trademarks

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Notes

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.