Key Management in an ICSF and non-CCA Environment

ICSF provides a capability of communicating with both CCA (Common Cryptographic Architecture) and non-CCA (Atalla, Racal, Jones Futurex, Eracom etc.) cryptographic engines.

The major method of accomplishing inter-system key management is to establish a common key between two communication end points (Zone Master Key, Importer and Exporter Key Encrypting Keys).
Once established, key changes can be communicated back and forth securely using keys that are now encrypted under these transport keys.

When ICSF communicates with another CCA node, no special considerations are required.
When ICSF communicates with non-CCA nodes, then the ICSF versions of the transport keys (IMPORTER Key Encrypting Key and EXPORTER Key Encrypting Key) need to be defined as not only an IMPORTER or an EXPORTER, but they must also have the special attribute of NOCV. Control Vectors, or CV's, are the CCA implementation of Key Variants.
The capability of allowing NOCV keys is defined at CKDS (ICSF Cryptographic Key Data Set) initialization time. From the ICSF ISPF (TSO) main menu, option 6 (CKDS Functions) is selected and from the subsequent menu, the selection to add NOCV keys is selected.

Transport keys in a CCA system are uni-directional only. That is, although the clear values are the same, a key defined as IMPORTER can only be used to receive keys encrypted with the transport key, and via a Key Import (CSNBKIM), securely translate that key to encryption under the ICSF Master Key. Similarly, the associated EXPORTER transport key (with the same clear text value as its complimentary IMPORTER key) can only be used to translate keys from encryption under the ICSF Master Key to encryption under the transport key (CSNBKEX).

Normally, the translated key type associated Control Vector (Key Variant) is used to securely modify the clear value of the transport key that is used to protect this translated key.
As an example, if a PINVER key is exported, the clear text value of the EXPORTER key is securely EXCLUSIVE ORed with the Control Vector of a PINVER key. The resultant exclusive ORed transport key is then used to encrypt that PINVER key (the translated one).
In this way, even if multiple key types of the same clear value are translated, their resulting translated values will still not be the same encrypted value. Variant encryption also ensures proper key separation and use.

When the NOCV attribute is associated with a transport key (IMPORTER and/or EXPORTER), this EXCLUSIVE ORing of the protected keys Control Vector is not done. Only the clear text value of the transport key is used to then encrypt and protect the translated key.
For legacy systems, without Control Vectors, this is the easiest method to communicate keying information from ICSF to and from those non-CCA systems.

It can also be seen, that since CCA keys are uni-directional, for each transport key (Zone Master Key), two CCA keys must be defined (IMPORTER and also EXPORTER) if keying information is to flow in both directions.
Some legacy devices, such as ATM's or POS terminals do not send keying information to ICSF so they can be serviced by only an EXPORTER key type (IMPORTER not required).

Also, since CCA recognizes legacy systems, DATA keys (used to ENCIPHER, DECIPHER, MACGEN and MACVERify), also do not use Control Vectors and inherently act as if they were being IMPORTED/EXPORTED with NOCV keys, even if the transport key does not have the NOCV attribute.

This then leads to communication with non-CCA systems either via DATA keys or via NOCV transport keys.
DATA key types are applicable for enciphering information, deciphering information and can optionally be used to generate MACs and verify MACs.
PIN and KEY MANAGEMENT functions in ICSF can not use DATA keys.

Some devices or systems may also employ the use of KEY VARIANTS that are used to securely modify (exclusive OR) their transport keys (ZONE MASTER KEYS).
In this case, although one set of clear text keys may be securely exchanged, those clear text values are never really used, instead, their varianted versions are used. This is the case for instance in the Canadian INTERAC and the international VISA or CIRRUS environment. Although clear transport keys are exchanged, only their First, Second and Third variants are ever used.

This still does not pose a problem in a CCA system. Instead of defining two uni-directional transport keys (IMPORTER/EXPORTER), whose clear text value is never used, we define the varianted versions of these transport keys.
Usually variant 1 means that a value of X'08' is exclusive ORed into byte 0 (leftmost) of a single length key and bytes 0 and 8 of a double length key.
To support variant 1, we enter the varianted version of the transport key into ICSF as two keys, Importer and Exporter.
The easiest way to do this is to enter the transport key in three (instead of two) key parts.
Parts one and two are the traditional two key parts (under dual custody). The third key part is then just the variant (as in X'0800 0000 0000 0000 0800 0000 0000 0000').
Similarly, if variants two (X'10') and three (X'18') are used, they too will be thusly defined eventually giving six transport keys (IMPORTER/EXPORTER for each of the variants).
And of course, these keys would have the NOCV attribute.

Regardless of the use of variants or not, IMPORTING a non-CCA key into ICSF requires that the non-CCA key be put into a structure that ICSF can work with. In this case, we take the 8 or 16 byte key value supplied by the non-CCA system and place it into a NULL key token.
A NULL key token is a 64 byte construct that is initialized to hex zeroes.
The key value is then moved into bytes 16 to 23 (0 being leftmost) for an 8 byte key, or into bytes 16 to 31 for a 16 byte key.
This construct can then be used as the SOURCE KEY value for the Key Import function (CSNBKIM).
The target for the Key Import is also a 64 byte construct. If the target is to be a DATA key, that construct can be initialized to hex zeroes.

If other than a DATA key is required (e.g.. Input Pin Encrypting Key or KPE), then the target construct is first initialized to hex zeroes and further prepared with the Key Token Build (CSNBKTB) function.
As an example, if a KPE (Input PIN Encryption Key or IPINENC) is desired, CSNBKTB is run with the target token being the 64 byte hex zero area, the KEY TYPE is set to IPINENC, a RULE ARRAY COUNT of 2, and RULE ARRAY set to two eight byte character fields containing "INTERNAL " and "NO-KEY ".
CSNBKIM is then run with the output of CSNBKTB as the target key token.
Since the input transport key has the NOCV attribute, the CSNBKIM function will not apply the IPINENC Control Vector to the source transport key, but when re-encrypting the IPINENC key under the Master Key, will apply the Control Vector (IPINENC) to the Master Key.

A non-CCA key value (8 or 16 bytes) has now been securely translated into the ICSF system for ICSF use. If the non-CCA system provides a single length key (8 bytes) that is required to be imported as a CCA double length key (16 bytes), then before the IMPORT (CSNBKIM), the supplied 8 byte encrypted key is placed into both bytes 16 to 23 and 24 to 31 of the NULL key token.

Similarly, if a CCA key is to be exported out of the CCA system or also if it is to be generated for use by ICSF and also a non-CCA system, the associated functions (CSNBKEX or CSNBKGN) use a NOCV EXPORTER transport key as the key to encrypt the target key value.
The Control Vector is again applied to the Master Key to retrieve the clear text key value, but since the target employees a NOCV key, that Control Vector is not applied to the associated transport key.
The resultant 64 byte construct will contain an 8 or 16 byte (bytes 16 to 23 or 16 to 31) encrypted value using just the clear text of the transport key (no variant applied). This 8 or 16 byte value can then be extracted and sent directly to the non-CCA system.
Again, this transport key can be either varianted or not. The resultant key value will then be in a form and encryption method understood by the non-CCA system.

Any non-CCA system, varianted or not can be handled in the same fashion.

If an existing file of encrypted keys is desired to be translated and input into a CCA system, the same technique can still be used.
A transport key (NOCV) is established that matches the key used to encrypt those keys.
The keys are put into a NULL construct (NULL TOKEN). The target token can be set to zeroes and KEY TOKEN BUILt as required (if not DATA).
CSNBKIM is then used to translate the key to encryption under the ICSF Master Key.
Any keys desired to be stored in the ICSF must have a key record entry prepared for them.
First a KEY RECORD CREATE (CSNBKRC) is executed. Next the NULL token is constructed (containing the encrypted 8/16 byte key). A KEY TOKEN BUILD (CSNBKTB) is run to prepare the target key type. CSNBKIM is run to translate the key into the target (KEY TOKEN BUILt) area. That target (64 byte) is then written (KEY RECORD WRITE - CSNBKRW) into the record that was created (CSNBKRC).

Automated Teller Machines (ATM's) are handled somewhat differently.
Some ATM's have a concept of using a cryptographic key for two functions.
In this case, sometimes the Communications (or SESSION) key can also be used to exchange subsequent session keys. In this case, that same key which acts like a DATA key (session), can also be used as a transport key (EXPORTER).
This provides an additional challenge, but once the aforementioned concept (NOCV) is understood, falls easily into place.

Usually an A, M (Master) or B (backup) key are defined as long lived ATM keys (transport keys, or EXPORTERs), again with attribute NOCV.
If no session key (C) exists, one is generated using the Master or Backup key.
Since this session key will encipher/decipher data, it can be generated as a DATA key.
To do this, a Key Generate (CSNBKGN) function is used.
Two session keys are required, one under the ICSF Master Key, the other under the ATM M or B key.
The parameters for Key Generate would then be:
KEY FORM of OPEX
KEY LENGTH of SINGLE
KEY TYPE 1 and 2 as DATA
KEK IDENTIFIER 1 as a NULL TOKEN
KEK IDENTIFIER 2 as the value or label of the pre-established M or B NOCV key.
GENERATED KEY 1 is then retained as the ICSF session key
GENERATED KEY 2 then has bytes 16 to 23 extracted and sent to the ATM as the new session key.
This accomplishes phase one, establishing a session key to decipher information coming from the ATM and to subsequently encipher information being sent to the ATM.

We must now transform that session key into an EXPORTER (transport key) for any subsequent key changes that use this C key.

In order to change how a key is used, we incorporate a concept of “flavour” keys.
These keys allow us to change the type, or flavour, of a key.
The main component of flavour keys is a NOCV transport (EXPORTER) and NOCV transport (IMPORTER) key. These two keys are defined as having the same clear text value. By exporting a key using the flavour EXPORTER, we strip the Control Vectors from the source key, thereby creating a legacy DATA type key, with no Control Vector (or variant) applied. The target of the EXPORT (CSNBKEX) will be a 64 byte construct with valid information in bytes 16 to 23 (or 16 to 31).
If the target key is to be changed into a double length key, but the source is single length, we replicate the values contained in bytes 16 to 23 also into bytes 24 to 31.
In this case we export (CSNBKEX) the above created C (DATA) key. We will be changing its flavour from DATA to a transport key (EXPORTER, double length). Once the export is successful, we replicate bytes 16 to 23 into bytes 24 to 31 all into a NULL token construct.
We prepare a new target token and customize it such that it can contain a NOCV transport (EXPORTER) key.

This customization is performed through a call to function CSNBKTB (Key Token Build) with options “INTERNAL”, “NO-KEY”, “NOCV-KEK” and “EXPORTER”.
Once complete, we now have a C (DATA or session) key used for encipher/decipher of ATM transaction data and an associated transport key (EXPORTER) that can be used for future key generation activity that requires a new session key to be generated encrypted with the current C (session) key.

The concept of flavour keys is very powerful, and therefor subject to certain restrictions. In order to protect against abuse, ICSF requires that an application that deals with NOCV key values as 64 byte constructs (as opposed to referencing them indirectly via their key label name), be strictly controlled and audited. As such, these applications must run authorized, and as the case with all user written authorized code, should be subject to strict review, audit and function evaluation.

Two further functions are associated with ATM transactions, verification of the CVV or CVC (VISA Card Verification Value) and validation of the entered PIN.
Once we have created the above C (session) key, we can decipher the transaction data.
To validate the CVV, we call the CSNBCSV (CVV Verify) function with associated magnetic card information and DATA keys (CVV-A and CVV-B). Once the card has been validated, we can then use the (still encrypted under the PIN encryption key) PIN with the CSNBPVR function call to validate the entered PIN.
When all validation is done, transaction authorization occurs and a reply is built that is then enciphered (CSNBENC) with the current C (session key).

A final consideration that some implementations rely on is the concept of a Key Check Value (KCV). This is a legacy implementation that provides some form of validation that an encrypted (with a transport key) key is as expected. The clear value of the key being exchanged is used to encipher or MAC (Message Authenticate) an eight byte value of hex zeroes. The resulting leftmost 4 digits of the encipher/MAC result is then appended to key management messages.
Since we have already explored flavour keys, we can use the same concept to change the flavour of a non-DATA or non-MAC key into a MAC key (export as DATA, import as MAC).
The now MAC key is then used to Message Authenticate (CSNBMGN) an eight byte hex zero string. It is suggested to use a MAC as opposed to DATA key so that the functionality of the flavour changed key can be restricted to non-decipher functions.

We can also make one further adjustment to our flavour keys so that the using application can only change the flavour of a key to that of MAC, thereby relaxing some of the associated security and integrity concerns.
When defining our flavour keys, we still define the EXPORTER and IMPORTER transport keys from their dual custody key parts. The EXPORTER is still defined as a NOCV transport key. Now, rather than also defining the IMPORTER transport key as a NOCV key, we add a third key part to the IMPORTER. That key part is the Control Vector associated with a MAC key. We add the Control Vector (concatenated into 16 bytes) thereby creating a Prepared IMPORTER transport key. Before exporting the subject key value, we extract its cryptogram (bytes 16 to 23) from its 64 byte construct and use that to Key Token Build (CSNBKTB) a source MAC construct

with options "INTERNAL", "KEY" and "MAC". That construct is now exported with the NOCV EXPORTER transport key. The encrypted key value (bytes 16 to 23) is then used to build a MAC construct. Since ICSF does not have a provision for Key Token Building an EXTERNAL token construct, we can build one with application code. First a NULL construct is built. Byte zero is set to hexadecimal 02, byte six is set to hexadecimal 0C and the encrypted key value is placed in bytes 16 to 23, and the Control Vector for a MAC key is placed into bytes 32 to 39. A Token Validation Value is now built and placed into bytes 60 to 63. This is then re-imported with the prepared IMPORTER transport key into a NULL construct as a MAC key. This MAC key can then be used to MACGEN eight bytes of hex zeroes to generate a Key Check Value.
The initial NOCV EXPORT serves to turn the key into a data key without Control Vectors. The application built MAC token will have its Control Vector applied to the prepared IMPORTER thereby nullifying the third key part (MAC Control Vector) and thusly still securely recovering the correct clear key value. The resulting IMPORT will build a MAC construct with the encrypted key now properly encrypted under the MAC variant of the Master Key.

Any key that is already DATA or MAC need not have its flavour changed, and can be used directly to MAC eight hex zeroes.