Existing Collection Mechanisms, Future Changes and ADI Education

Josh Wisniewski
Tooling Architect

IBM z/TPF
April 13, 2016

25 Minutes	**Resource Usage Collections Education**
10 Minutes	**Other features in plan**
45 Minutes	**Guidelines for Implementing ECB Owner Names and Name-Value Pairs**

Resource Usage Collections Education

Agenda

- Existing Mechanisms
  - Business Value
  - ECB Resource Monitor and Named Limit Sets
  - ECB Owner Names
  - Block Owner Names
  - Name-Value Pairs
- Review
- Deliverable details

Business Value of Existing Mechanisms

- An operator can use the **ECB Resource Monitor** to prevent a single task or group of tasks from using too many system resources that would otherwise result in an outage.
- A coverage programmer can use **ECB owner names** to understand what system resources are used by individual code packages to easily identify where to start investigating a resource consumption problem.
- A coverage programmer can use **block owner name** information to identify the cause of running low on memory and take action before an outage occurs.
- An application programmer can use **name-value pairs** to pass credentials, request details, and more without having to change program interfaces or worry about how and where the data is stored.

Existing Mechanisms

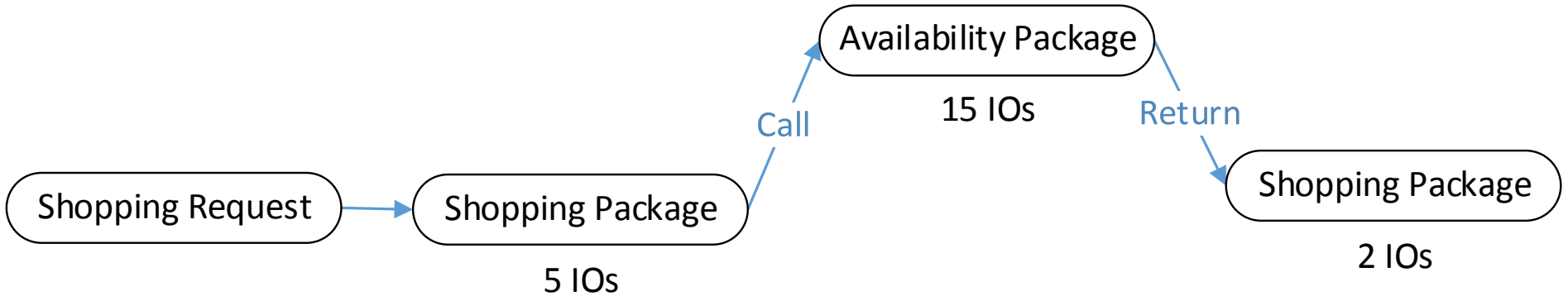
- ECB Resource Monitor and Named Limit Sets.
  - Why this mechanism exists.
  - How to use this mechanism.
  - Guidelines for usage.
  - More information.

ECB Resource monitor: Why it exists?

- The ECB Resource monitor is a policing mechanism that can take actions when an ECB exceeds resource limit thresholds as set by the system admin (such as killing the ECB).
- The ECB Resource monitor should be used to prevent run away conditions that could take the processor down.
- The ECB Resource monitor leverages named limit set counts in the following ways:
  - By the admin to determine where to set the resource limit thresholds.
  - By the z/TPF system to determine when a resource limit threshold has been exceeded by an ECB.

Named limit sets: Why it exists?

- Allows you to determine how much system resources are used to process a request.
- Conceptual example:



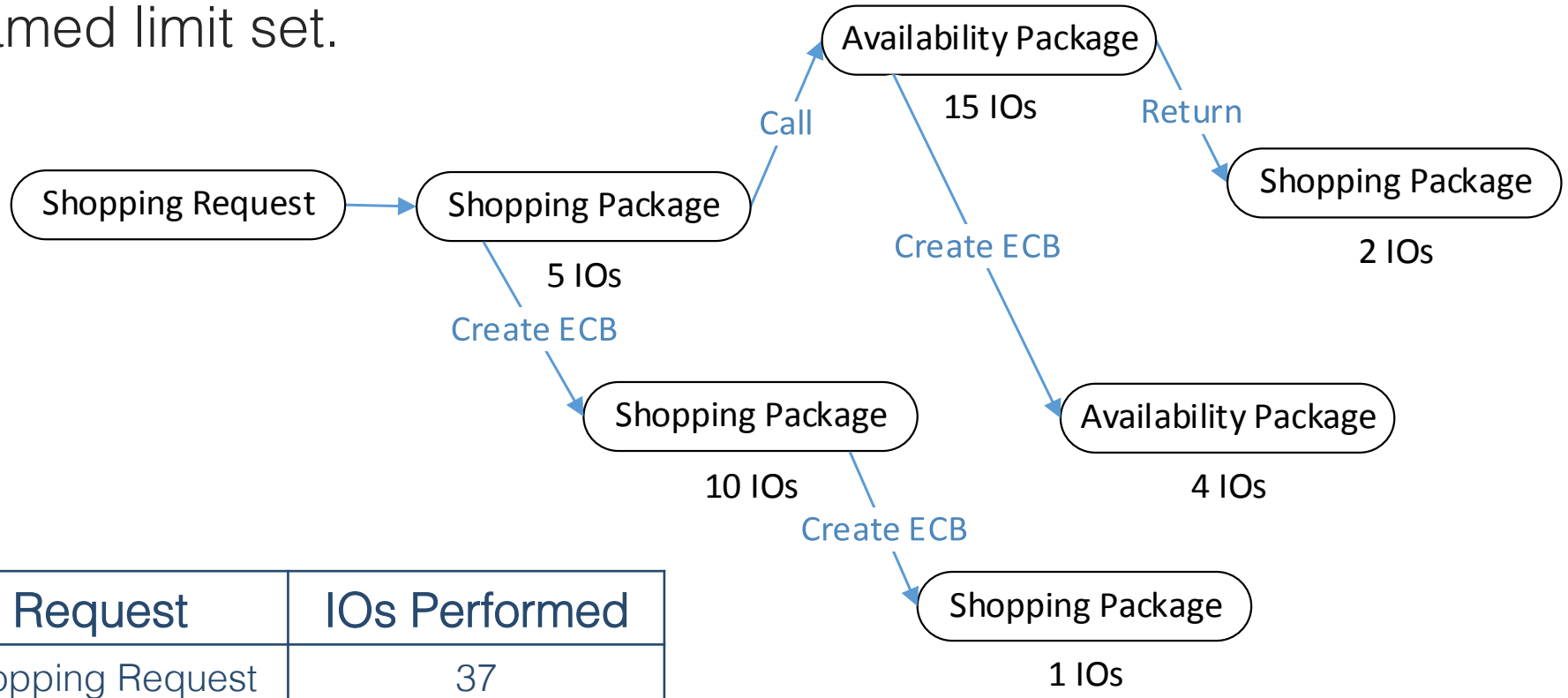
Request	IOs Performed
Shopping Request	22

ECB named limit sets: Why it exists?

- The named limit set counts allows you to see:
  - High water mark of the use of resources when processing a request type.
  - The average usage of resources when processing a request type.

ECB named limit sets: Why it exists?

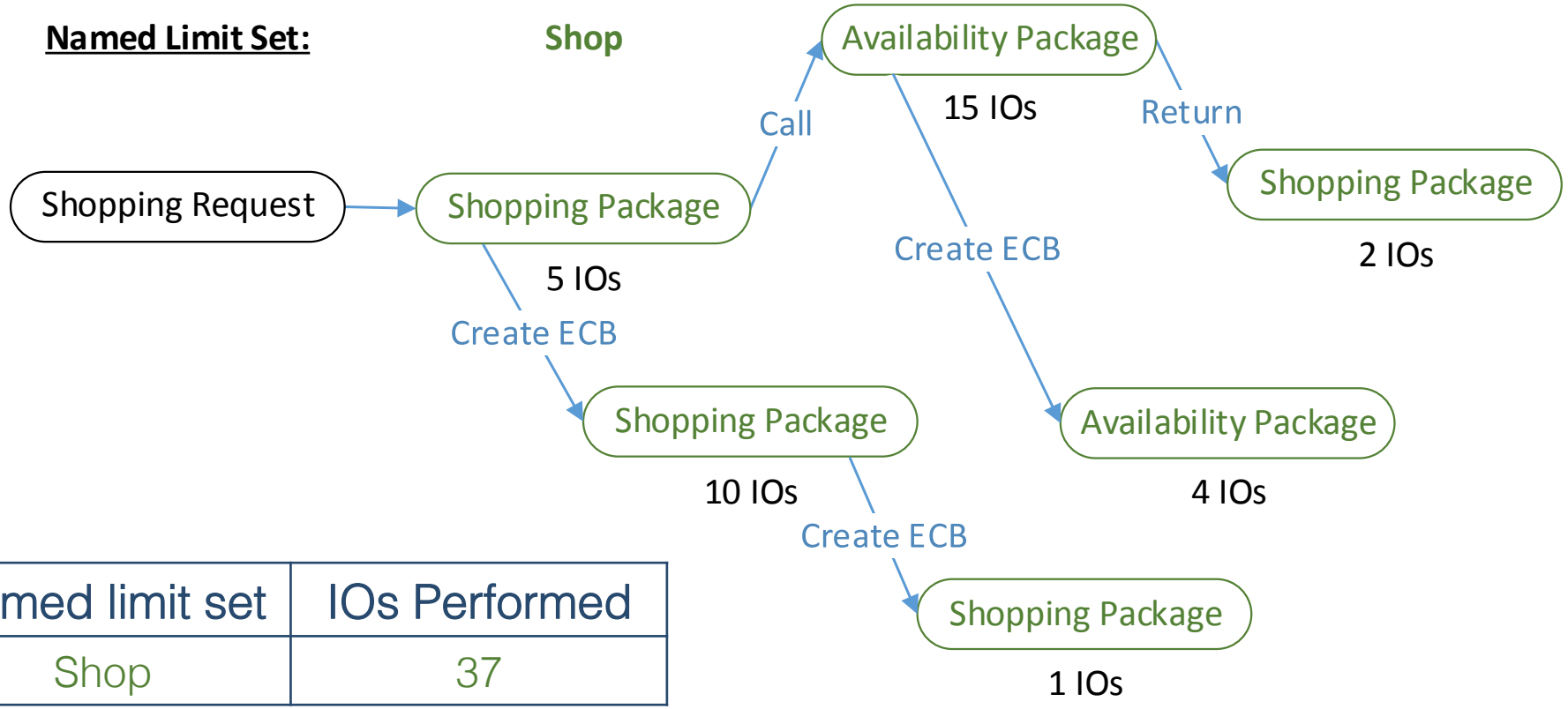
- Named limit sets can be specified as groups allowing the system resources to be counted for the parent and all created children ECBs. The children ECBs will inherit the named limit set.



Request	IOs Performed
Shopping Request	37

ECB named limit sets: How to use it?

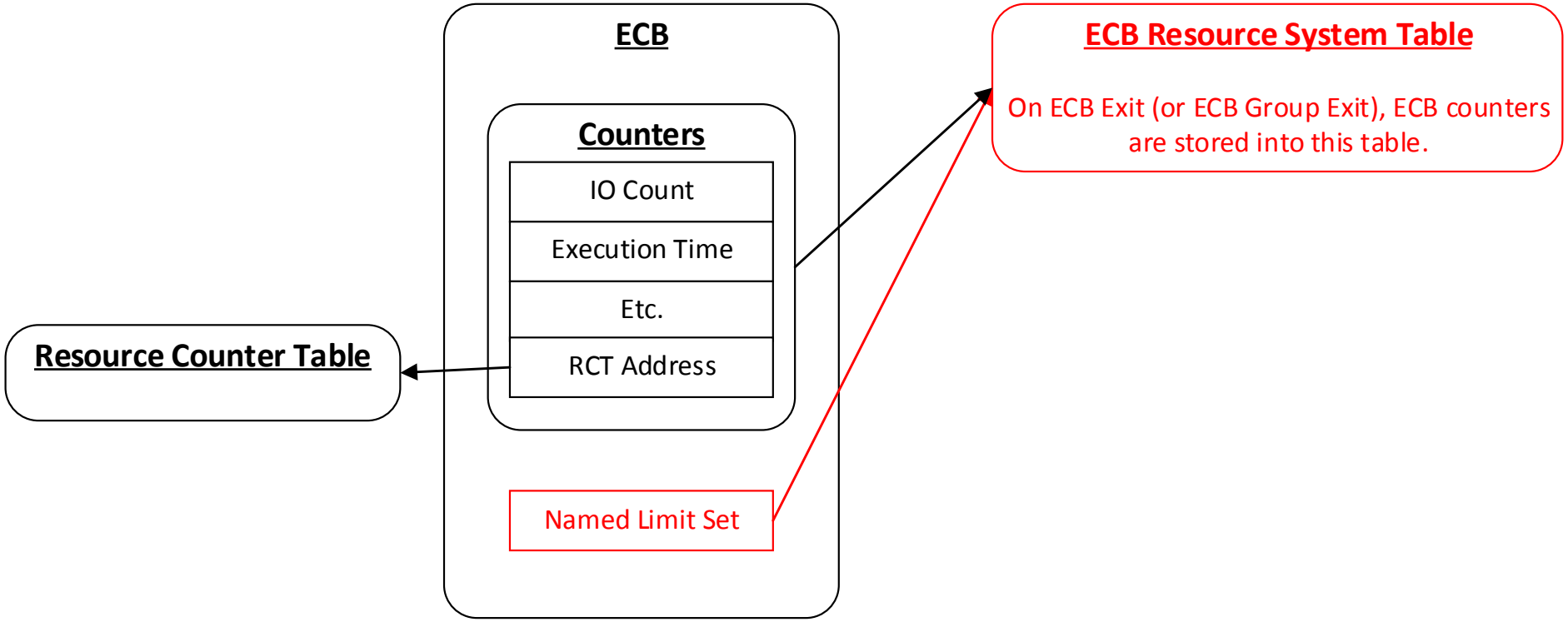
- Use the ECBMC macro or tpf_ecbmc_set_lsetname C API to set the named limit set for the ECB at the beginning of processing a request.



Named limit set	IOs Performed
Shop	37

ECB named limit sets: How to use it?

- A variety of counters are associated with the named limit set.
- At ECB exit, these counters are accumulated into a table.



ECB named limit sets: How to use it?

- ZECBM is used to:
  - Define named limit sets and groups. There is no limit to how many can be defined.
  - Perform a collection of the counters.
  - Manage named limit set thresholds for the ECB resource monitor.
- Use ECBMC, tpf_ecbmc_query, and tpf_ecbmc_set to manage exceptions to the ECB resource monitor policing.

ECB named limit sets: Guidelines for usage.

- We recommend this facility only be used for setting thresholds for the ECB Resource monitor to prevent run away conditions that could take the processor down.
- We do not recommend using this facility for the collection of general performance metric data.
- When an aberrant application situation is discovered, set the named limit set once when processing a request.
- If using name limit set group, set the name limit set group prior to creating child ECBs.
- IBM code does not use named limit sets.

ECB named limit sets: More Information.

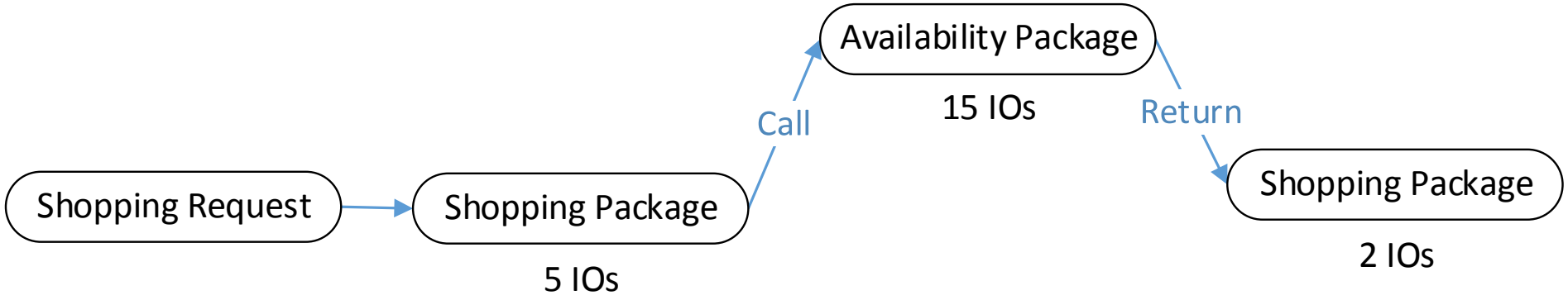
- ECB resource monitor: https://www-01.ibm.com/support/knowledgecenter/SSB23S_1.1.0.12/com.ibm.ztpf-ztpfdf.doc_put.12/gtpa2/apecbrm.html?lang=e
- Resource usage by named limit set: https://www-01.ibm.com/support/knowledgecenter/SSB23S_1.1.0.12/gtpa2/a2corubnls.html?lang=en
- Resource usage by named limit set collection overview: https://www-01.ibm.com/support/knowledgecenter/SSB23S_1.1.0.12/gtps3/corunlscovw.html?lang=en

Existing Mechanisms

- ECB Owner Names
  - Why this mechanism exists.
  - How to use this mechanism.
  - Other usage.
  - Guidelines for usage.
  - More information.

ECB Owner Names: Why it exists?

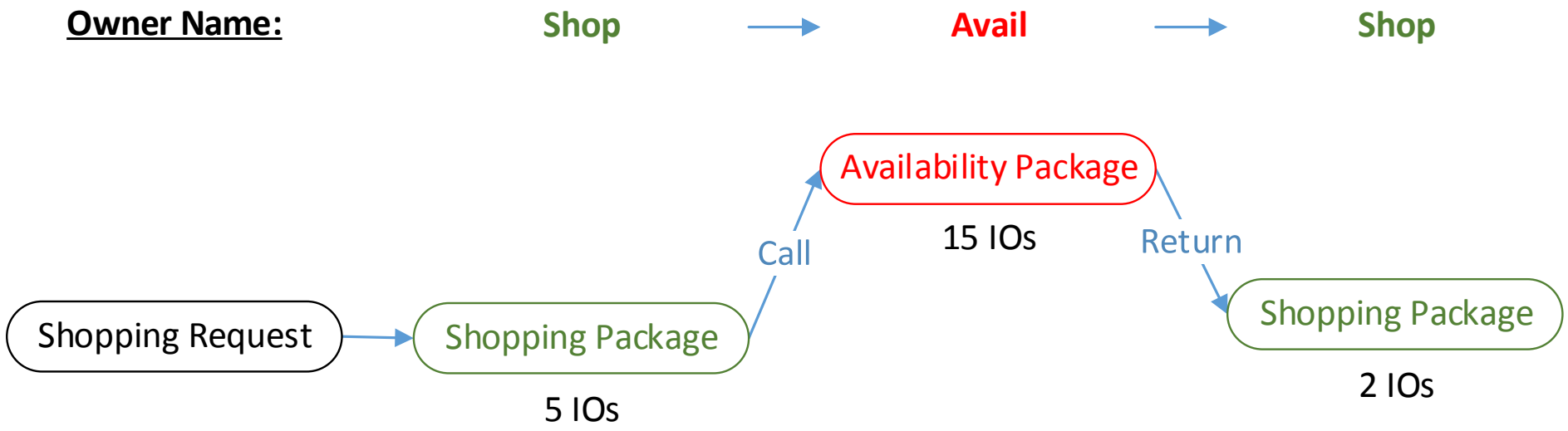
- Allows you to determine what system resources are being used and by which code packages.
- Conceptual example:



Package	IOs Performed
Shopping	7
Availability	15

ECB Owner Names: How to use it?

- Use EOWNRC macro or tpf_eownrc C API to set the owner name for the ECB when a package is called and upon return.



Owner Name	IOs Performed
Shop	7
Avail	15

ECB Owner Names: How to use it?

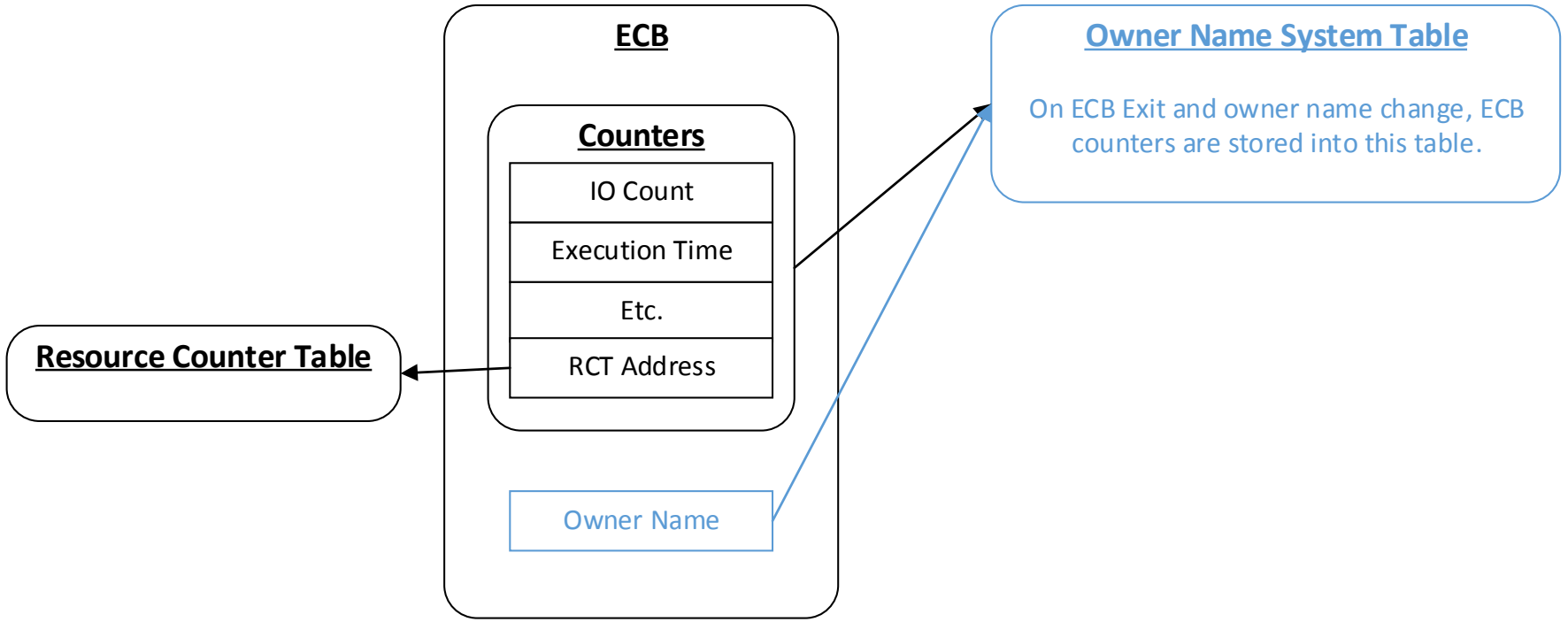
- Owner names are composed of 3 levels
  - High level qualifier – Should be used to indicate the package (the code that is being executed).
  - Mid level and low level qualifier – Can be used to provide additional information.

    SHOP.AIR.MONSTERtrvlSITE

    SHOP.CAR.MOMandPOPtrvl

ECB Owner Names: How to use it?

- A variety of counters are associated with the owner name.
- ZMOWN manages owner name collection.
- Upon changing the owner name or ECB exit, the counters are accumulated into a table.



ECB Owner Names: How to use it?

- Using a configuration file, ZMOWN can be used to define up to 64 owner name buckets for collection. A bucket may encompass multiple owner names.

```
CONFIGURATION_VERSION=1
1,SHOP*
2,AVAIL*,PRICING*
```

- Mid level qualifiers can be specified in the configuration file to break down owner name counters further

```
CONFIGURATION_VERSION=1
1,SHOP*.AIR*
2,SHOP*.CAR*
```

- Low level qualifiers are ignored for owner name collection.

ECB Owner Names: How to use it?

- Owner name usage report from ZMOWN usage.

```
OWNER NAME RESOURCE USAGE REPORT,OUTPUT_VERSION=2
PROCESSOR,2094-728
CPUID,B
NUMBER OF I-STREAMS,1
SUBSYSTEM NAME,BSS
START,Tue Jan  5 16:50:12 2016
END,Tue Jan  5 16:51:12 2016
ELAPSED TIME IN SECONDS,60
CONFIGURATION FILE LAST MODIFICATION TIME, Tue Jan  5 16:49:45 2016
Bucket number,Owner Name List,ECBs counts,CPU time consumed,MAX CPU time, ...
0            ,               ,664        ,152529           ,15268       , ...
1            ,SHOP*          ,10000      ,999888777        ,99999       , ...
2            ,AVAIL*PRICING* ,20000      ,112233445566     ,115555      , ...
```

ECB Owner Names: Other usage

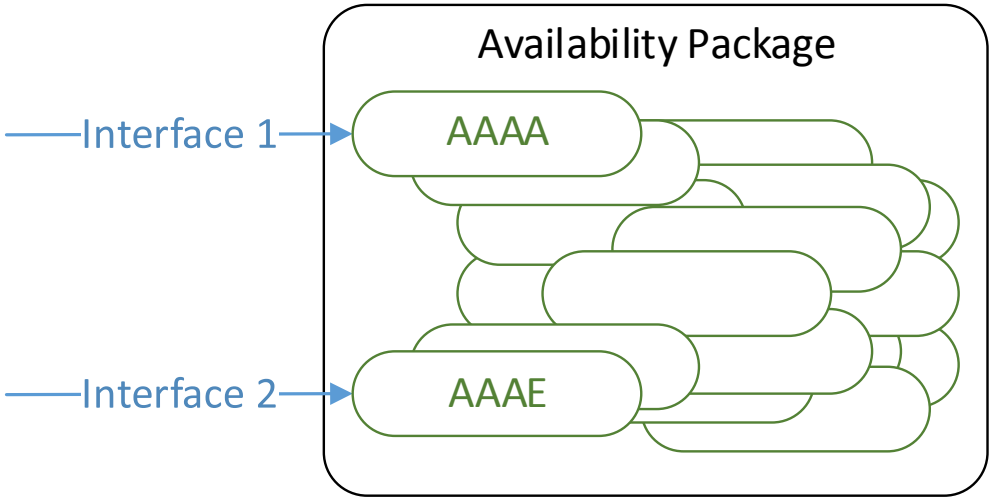
- The ZDECB filter and displays have been enhanced show ECB owner names.

```
User:   ZDECB USER OWNER-SHOP*
System: DECB0014I 02.28.59 DISPLAY ECB SUMMARY
        ECB ADDR IS  PGM     TRC MIN SC LSETNAME OWNER NAME
        1010C000  1 SAAA    SAAA  14 45 DEFAULT  SHOPPING.AIR
        TOTAL       1
        END OF DISPLAY
```

- The Software Profiler (ZTRAP) can restrict collection to the ECB owner name specified.

ECB Owner Names: Guidelines for usage.

- Changing ECB owner names frequently can impact performance due to the overhead of writing out the counters. For example, do not change ECB owner names on entry to shared library routines.
- We advise changing the ECB owner name at the main interface points to a package (ie AAAA entry point for interface 1 and AAAE entry point for interface 2 below).



ECB Owner Names: More Information.

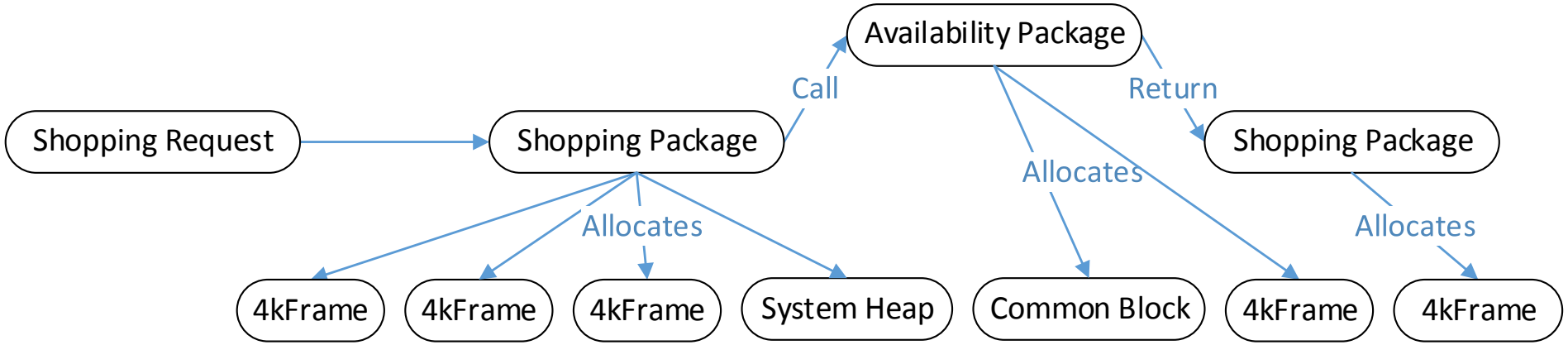
- Resource usage by owner name collection: https://www-01.ibm.com/support/knowledgecenter/SSB23S_1.1.0.12/com.ibm.ztpf-ztpfdf.doc_put.12/gtps3/coreusbonm.html?lang=en

Existing Mechanisms

- Block Owner Names
  - Why this mechanism exists.
  - How to use this mechanism.
  - Other usage.
  - Guidelines for usage.
  - More information.

Block Owner Names: Why it exist?

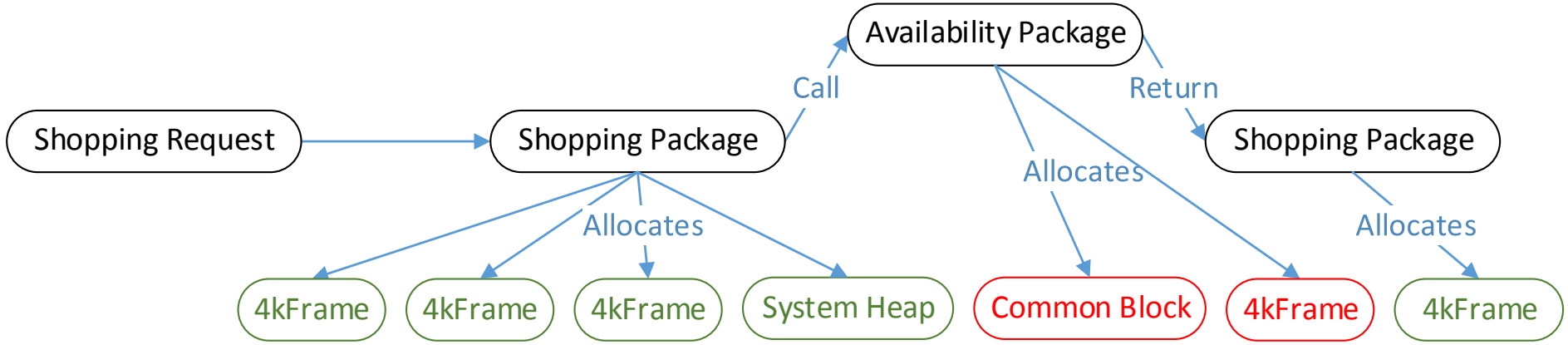
- Allows you to determine what block resources are being used and by which packages.
- Conceptual example showing allocation:



Block Type	Number Used
4kFrame	5
System Heap	1
Common Block	1

Block Owner Names: How to use it?

- Some block types (ie common blocks) inherit the ECB owner name. Other block types require setting the owner name explicitly such as GSYSC.



Owner Name: **Shop** **Avail** **Shop**

Owner Name	4kFrame	System Heap	Common Block
Shop	4	1	0
Avail	1	0	1

Block Owner Names: How to use it?

- Use ZSTAT U to understand overall block usage on the system.
- Then use ZSTAT OWNER to show a breakdown of block usage by owner name.

```
STAT0023I 14.34.11 BLOCK OWNER DISPLAY
                        IOB    FRAME  COMMON      SWB      ECB     FRM1MB
ALLOCATED              4456     6200     400     2504      600        400
AVAILABLE              4456     6153     395     2460      592        140
                        IOB    FRAME  COMMON      SWB      ECB     FRM1MB
SHOP                      0        0       0        4        0          1
AVAIL                     0        0       1        1        0          0
END OF DISPLAY
```

- Use ZSTAT SYSHEAP NAME-ownername to show a breakdown of system heap usage by owner name.

ECB Owner Names: Other usage.

- Display or modify dump override table (ZIDOT) leverages block owner names.

Block Owner Names: Guidelines for usage.

- Update application usage of APIs that require explicitly setting the owner name such as GSYSC.
- Setting owner names on block will not have any adverse effects.

Block Owner Names: More Information.

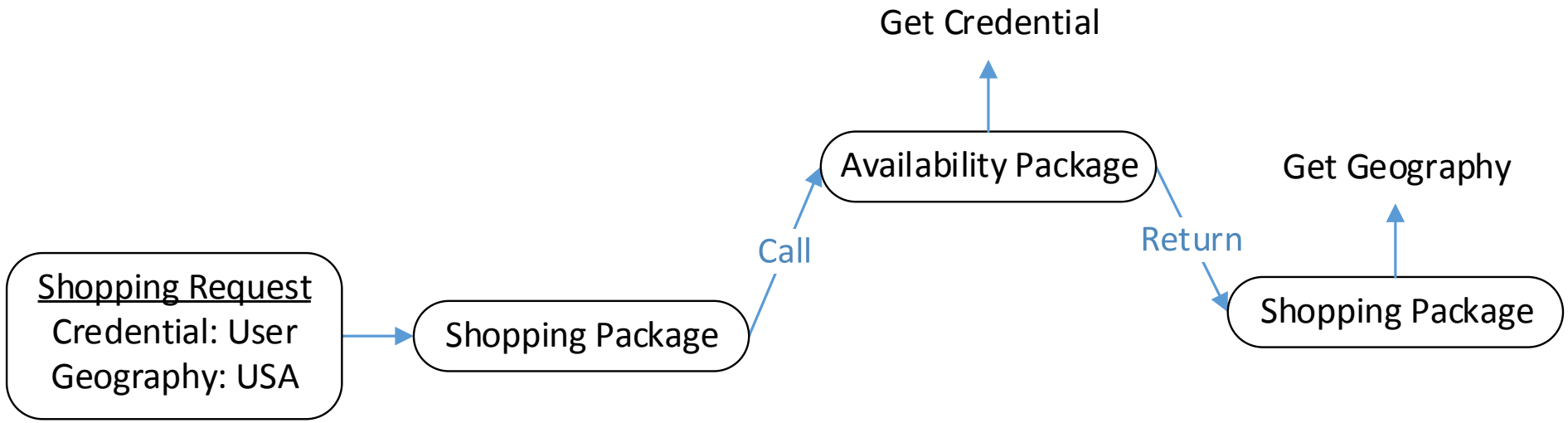
- Physical block owner name support: https://www-01.ibm.com/support/knowledgecenter/SSB23S_1.1.0.12/gtpm6/howner.html?lang=en

Existing Mechanisms

- Name-Value Pairs.
  - Why this mechanism exists.
  - How to use this mechanism.
  - Guidelines for usage.
  - More information.

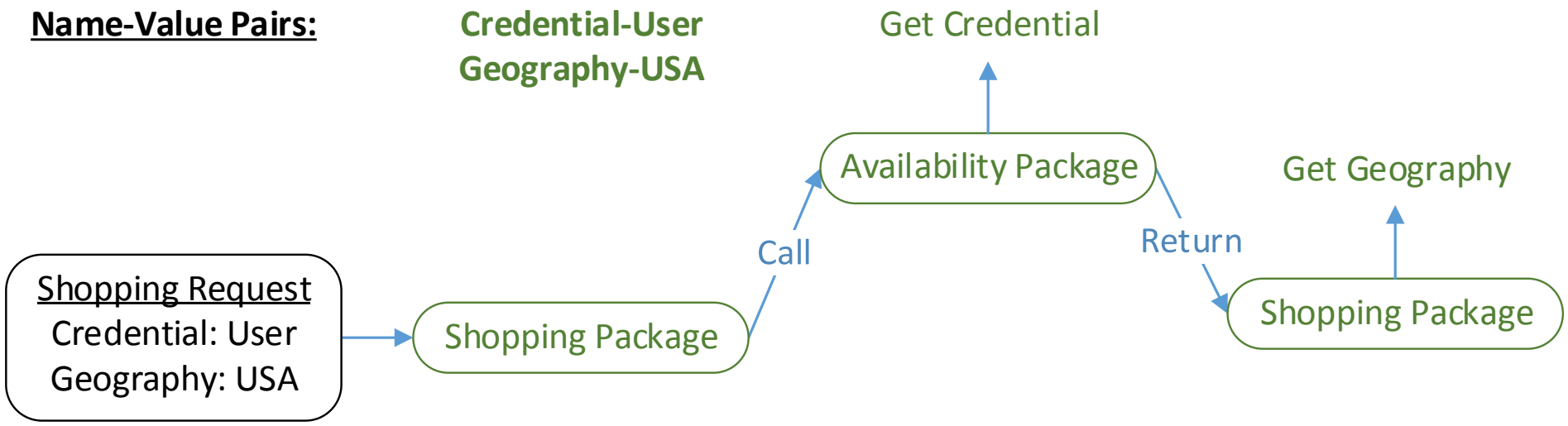
Name-Value Pairs: Why it exists?

- Allows you to make data available any where it is required in the application, for example: credentials, source of request, request details, and etc.
- Conceptual example:



Name-Value Pairs: How to use it?

- Use tpf_nameValueLocalSet to set a name-value pair, tpf_nameValueLocalGet to get the value for a name, and tpf_nameValueLocalRemove to remove a name-value pair.
- Name-value pairs are passed to created ECBs.
- Name-value pairs can be set, get and removed at any time.



Name-Value Pairs: Guidelines for usage.

- All of the name-value pairs are stored in a single 4k block. If you intended to utilize many different name-value pairs per ECB, consider using concise names and values.

Name-Value Pairs: More Information.

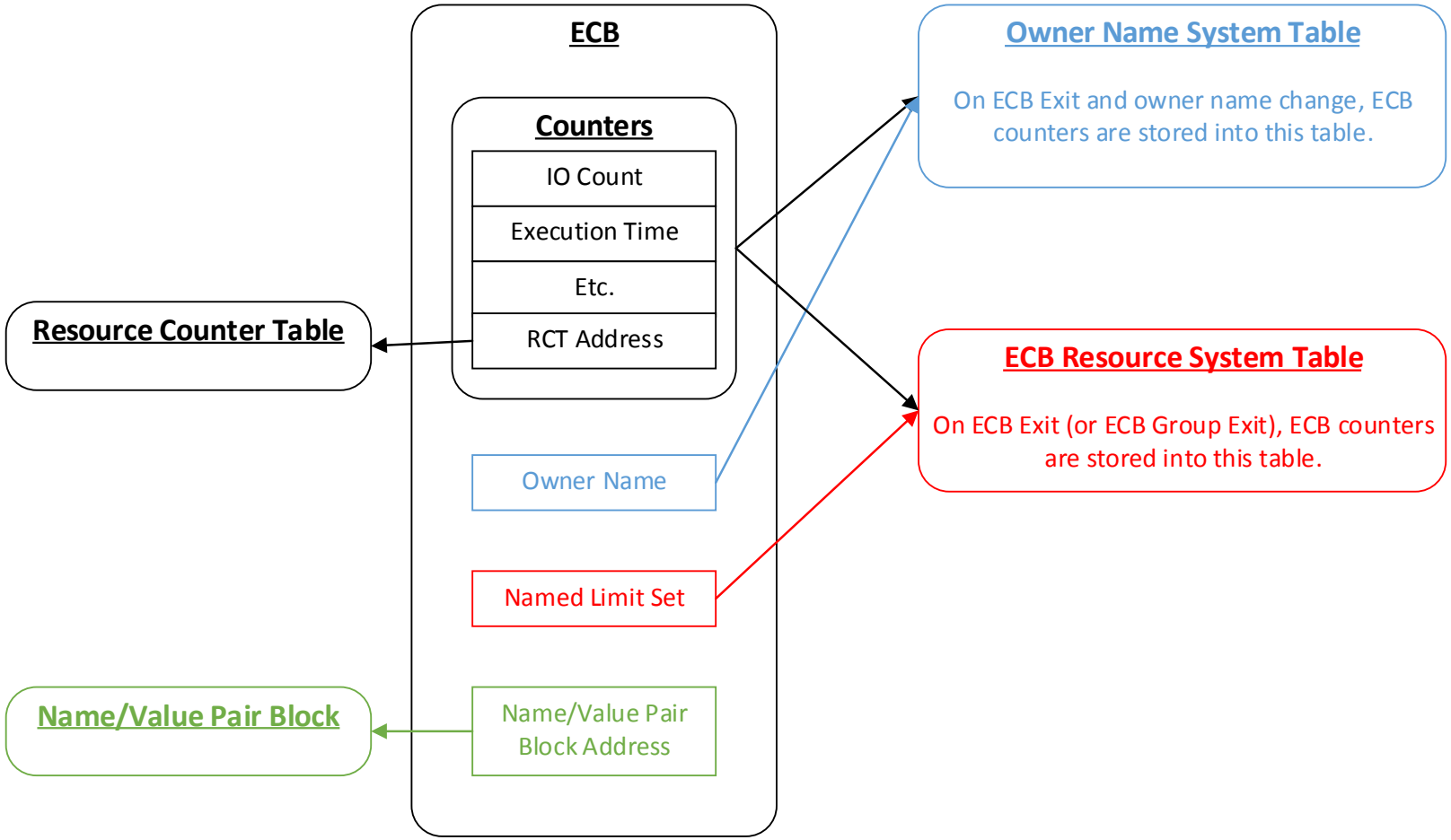
- Name-value pair support: https://www-01.ibm.com/support/knowledgecenter/SSB23S_1.1.0.12/gtpa2/namevaluepair.html?lang=en

Existing Mechanisms

- Summary.
  - Overview of existing mechanism.
  - Usage.

Summary: Overview of existing mechanism

- ECB owner name and named limit set use the same counters found in the ECB. However, they provide differing breakdowns of the counter data.



Summary: Usage

- Named limit set: Understand what resources request processing is using for the ECB Resource monitor to prevent outages.
- ECB owner names: Understand what resources code packages are using.
- Block owner names: Understand what blocks code packages are using.
- Name-value pairs: Make data available throughout the application (credentials, request source, request details, etc).

Deliverable details

Description	z/TPF PUT Level
Owner name support	GA/10
ECB resource monitor named limit set	10/12
Name-value pair	11

Other features in plan

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Agenda

- Business value
- Other features in plan
- New Name-Value Pair APIs
- Owner name behavior changes
- Start request processing API

Business Value of Existing Mechanisms: To be

- ECB owner names: More accurate understanding of what resources code packages are using. More granularity of resources used in ADI analysis.

- Name-value pairs: Make data available throughout the application. Name-value pair collection in ADI provides greater depth of understanding of resource usage than existing tools.

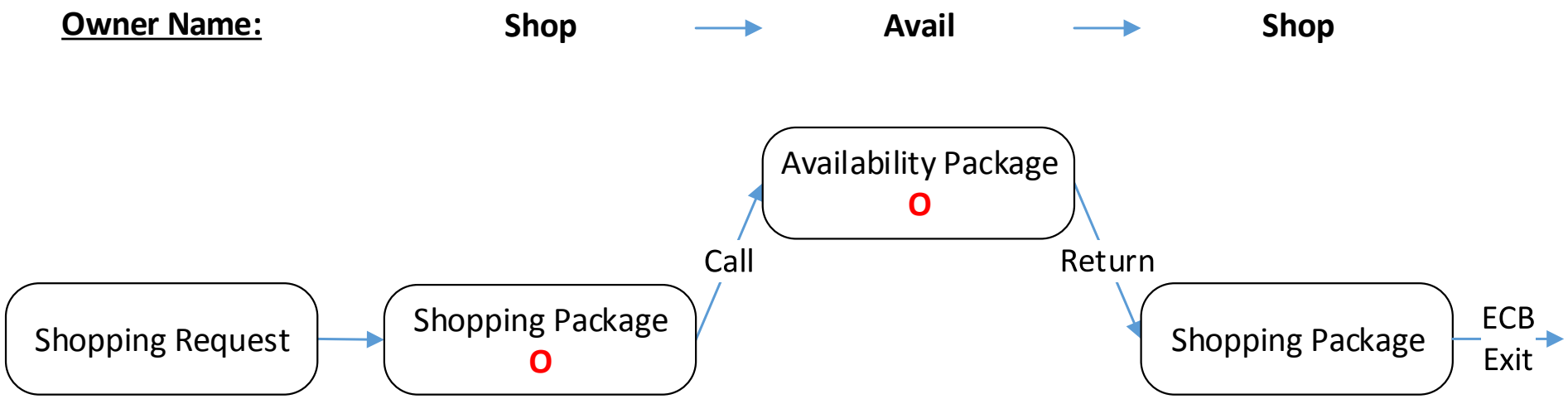
Automatic Owner Name Restore

Problem Statement

- Owner names are not automatically restored when the code returns to the caller of a package.

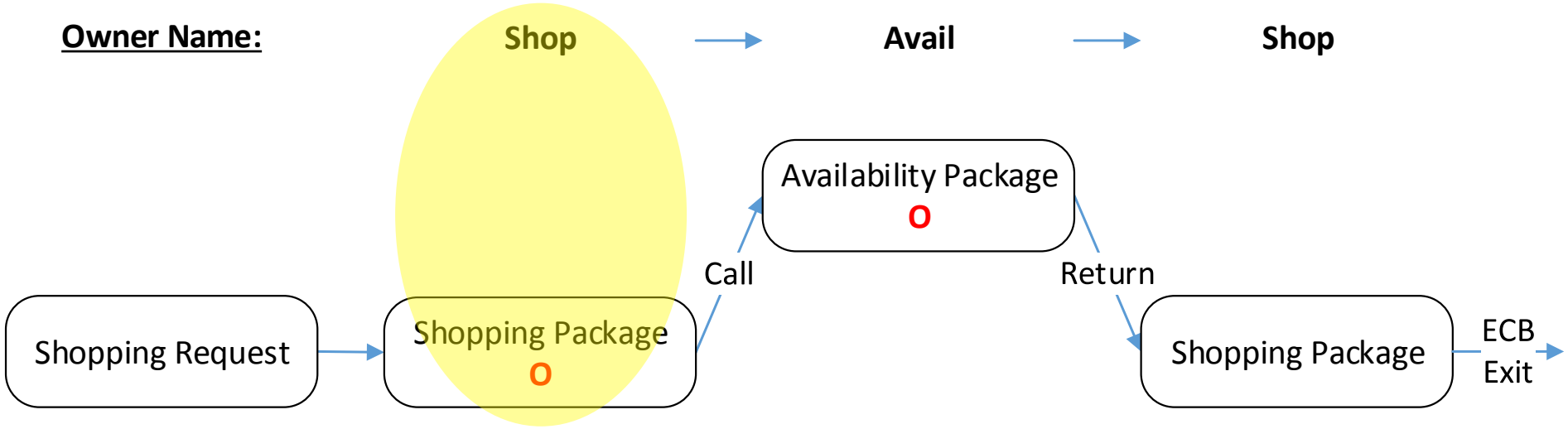
Automatic Owner Name Restore

- As previously discussed, the application is changed to set the owner name at the main interface points to a package. This is represented by the red O in the example below.
- In this example, the shopping package interface will set the owner name to Shop and the availability package interface will set the owner name to Avail.



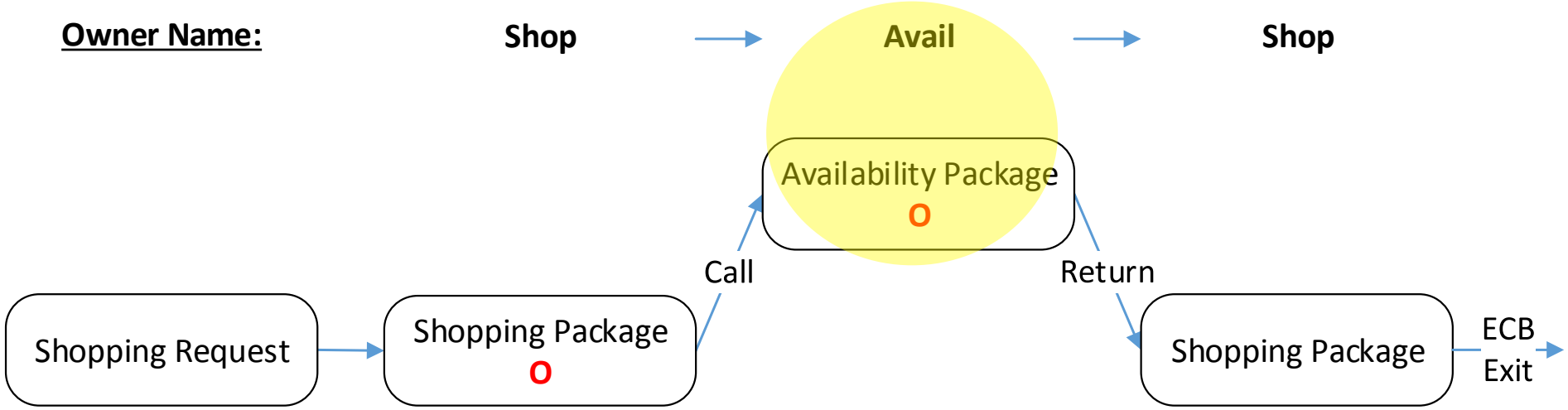
Automatic Owner Name Restore

- At the main interface to the shopping package, the owner name is set to Shop.
- Since this is the first time the owner name is set, the counters are not written out.



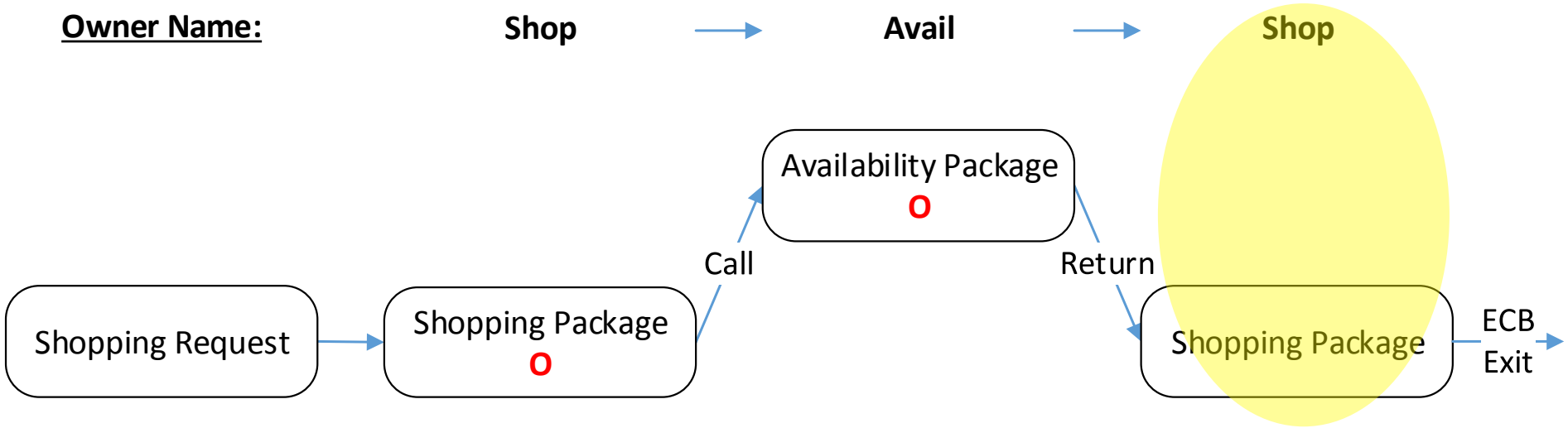
Automatic Owner Name Restore

- At the main interface to the availability package, the owner name is set to Avail.
- The state of the counters is written into the ZMOWN bucket that subsumes the Shop owner name.



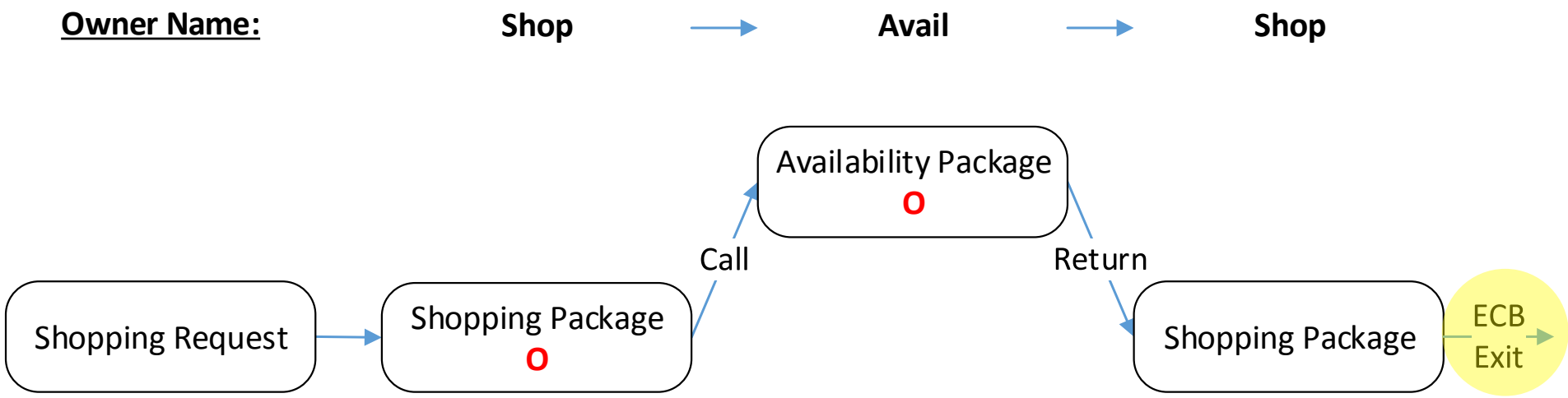
Automatic Owner Name Restore

- Currently, upon return to the shopping package, any resources used will continue to be attributed to the availability package.
- When this feature is delivered, the system will now automatically restores the owner name to Shop.
- The state of the counters is written into the ZMOWN bucket that subsumes the Avail owner name.



Automatic Owner Name Restore

- The ECB exits.
- The state of the counters is written into the ZMOWN bucket that subsumes the Shop owner name.



Automatic Owner Name Restore

- When you change the owner name, the previous owner name will be associated with the stack frame where the change occurred.
- Return or BACKC from that stack frame will:
  - Write out counters for the current owner name.
  - Reset the owner name to previous owner name.
- Automatic owner name restore will only occur on cross module calls.
- No interface/application changes will be required.
- If desired, you may reset the owner name manually until these changes are in place. However the ECB Count will be artificially inflated until these changes are delivered.

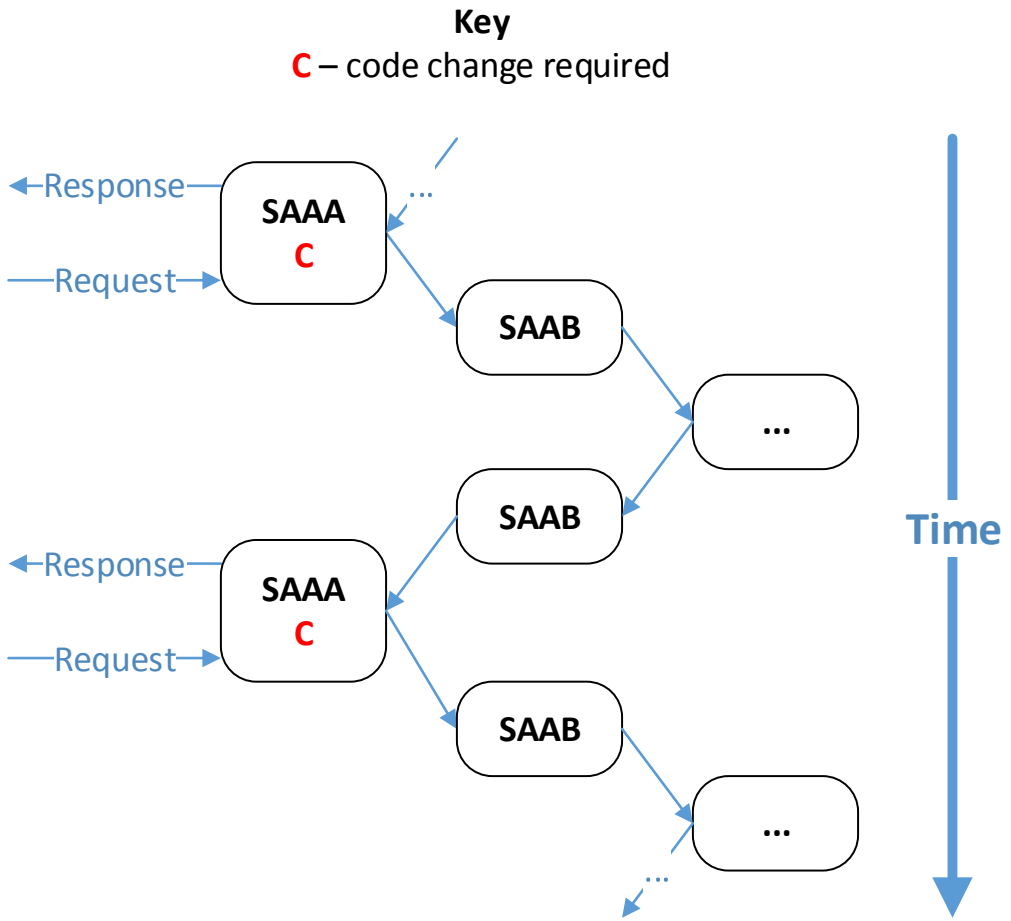
Other Owner Name Changes

- Setting owner name to same owner name will have no effect.
- Setting the owner name twice in an ECB for the same stack frame will result in an error.

Start request processing API

Problem Statement

- In some cases, a single long running ECB may be used to process multiple requests. Currently, resource usage metrics may not be written out until the ECB exits.



Start request processing API

- A new tpf_startRequestProcessing() API will be created to accomplish the following:
  - Write out counters for the owner name, named limit set, and name-value pairs.
  - Reset all ECB counters to zero, as if it's a new ECB.
  - Resets owner names and named limit set to the system defaults. All name-value pairs are unset.
  - ZMOWN reports will treat this ECB as a new ECB.
- The tpf_startRequestProcessing() API should be used for cases where a single ECB is used to process many requests. This will be useful for server-client models like Java or other thread pool worker applications.

New Name-Value Pair APIs

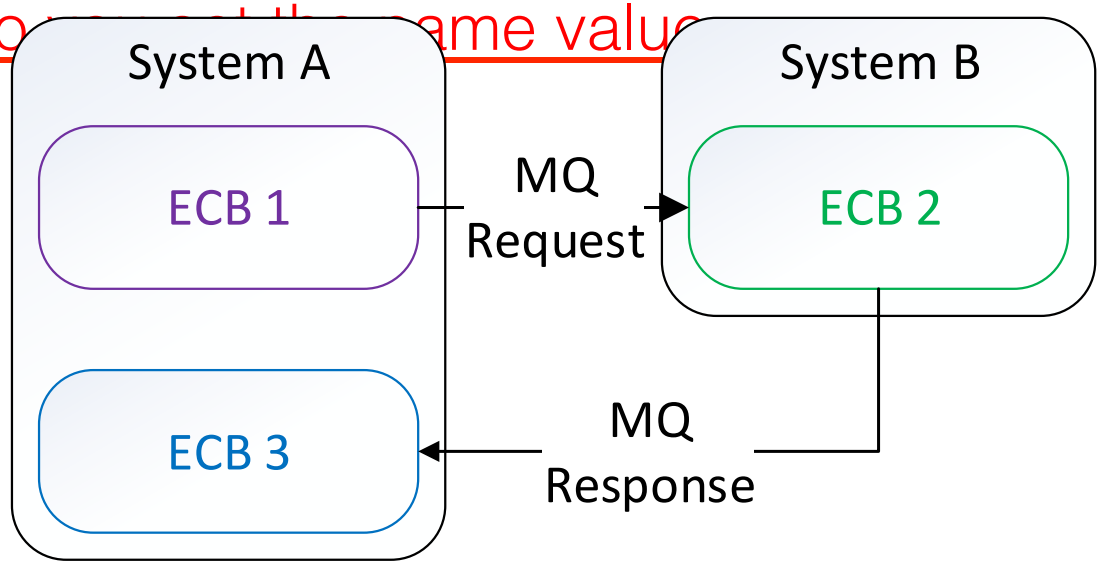
Problem Statement

- TPF request processing often spans calls to other systems. How are name value pairs passed to called systems? How are name-value pairs re-established upon resuming processing on the initial system?

New Name-Value Pair APIs

Example (problem statements underlined in red). The resources consumed by all 3 ECBs constitute a single unit of work.

- ECB 1: Is created on system A. Name-value pairs are set.
- ECB 1: Sends an MQ request to system B.
- ECB 1: Exits.
- ECB 2: Is created on system B. How do you set the name value pairs?
- ECB 2: Completes processing. Sends an MQ response to system A.
- ECB 2: Exits.
- ECB 3: Is created on system A. How do you set the name value pairs?
- ECB 3: Resumes request processing.
- ECB 3: Exits.



New Name-Value Pair APIs

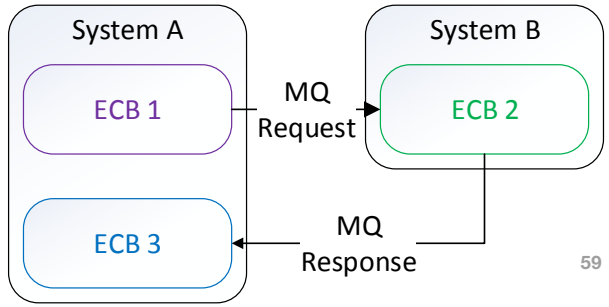
Two new APIs will facilitate saving and restoring name-value pairs

- tpf_NameValueLocalSave() will be created to save the name-value pairs set in the ECB into a buffer. This API will be useful in saving name-value pairs for later use, sending over communications protocols and etc.
- tpf_NameValueLocalRestore() will be created to set the name-value pairs in the ECB from a buffer where the tpf_NameValueLocalSave() API was used to previously save. This API will be useful in setting name-value pairs that were saved for later use, sent over communications protocols and etc.

New Name-Value Pair APIs

Example (New customer added code underlined in black)

- ECB 1: Is created on system A. Name-value pairs are set.
- ECB 1: Calls tpf_NameValueLocalSave() to get the name-value pairs.
- ECB 1: Saves the name-value pairs to a pool record for ECB 3.
- ECB 1: Adds the name-value pairs to the MQ request for ECB 2.
- ECB 1: Sends an MQ request to system B.
- ECB 1: Exits.
- ECB 2: Is created on system B. Uses tpf_NameValueLocalRestore() to set the name-value pairs from the MQ request sent by ECB 1.
- ECB 2: Completes processing. Sends an MQ response to system A.
- ECB 2: Exits.
- ECB 3: Is created on system A. Uses tpf_NameValueLocalRestore() to set the name-value pairs from the pool record saved by ECB 1.
- ECB 3: Resumes request processing.
- ECB 3: Exits.



Guidelines for Implementing ECB Owner Names and Name-Value Pairs

For use today and preparing for the new Application Delivery Intelligence (ADI) Name-value pair collection mechanism.

Agenda

- Business value
- Overall guidelines
- Examples
  - Multiple ECBs processing one request, calls out to other systems, and etc.
  - One ECB for multiple requests

Business value

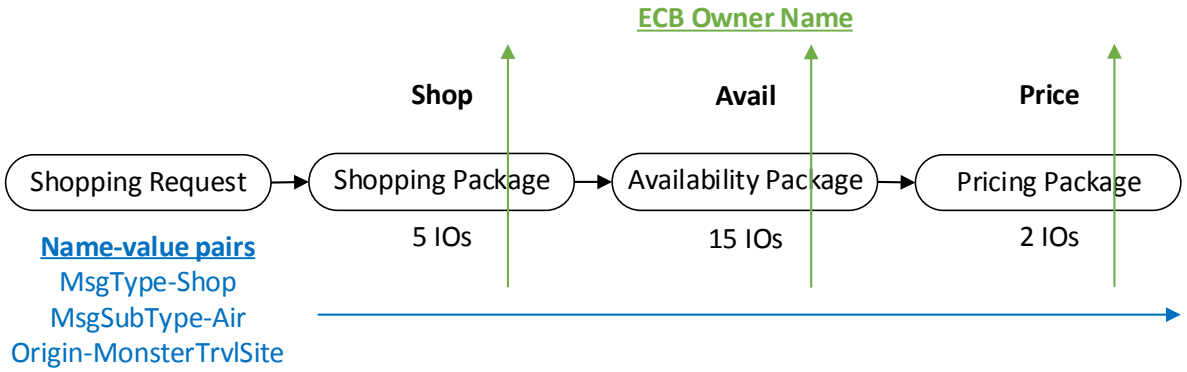
- The following guidelines are intended to be implemented by customers today.
- The immediate benefit is leveraging owner name collection to understand resource usage by code package and name-value pairs for communicating data.
- The future benefit is being able to readily leverage the name-value pair collection, ADI, owner name changes and etc when these features become available.

Overall Guidelines

- The name limit set mechanism should only be used for the ECB Resource monitor to prevent run away conditions. As such, it will not be discussed further in this presentation.

Overall Guidelines

- Develop and implement a unified strategy across all of your applications for using the owner names and name-value pairs. Consider the following:
  - How ECB owner names (vertical/code packages) and name-value pairs (horizontal/request categorization) can be used together to understand resource usage metrics.



  - Define a methodology for saving/restoring and/or transferring name-value pairs across off board requests.

Overall Guidelines

- ECB Owner name strategy.
  - Define a consistent owner name naming scheme.
  - Standardize owner name usage for all 3 levels.
  - Define when and where owner names will be set.
  - Use owner names to delineate code packages.
  - Set owner name on main interface points to a code package.

- Note: If you have previously used ECB owner names and you change your ECB owner name usage your block owner name will also be affected.

Overall Guidelines

- ECB Owner name naming strategy.
  - Consider owner name level relationships. How will you use the mid level to subdivide the high level? How will you use the low level to subdivide the mid level? For example, you may want to use a scheme such as:
    - Code package.Code subpackage.Further refinement (useful for ZMOWN and etc)
      - SHOPA.AIR.Domestic
    - Code Package.User.Database or Package.Database.User (useful for different views from ZMOWN)
      - SHOPA.USERA.AIRDB

Overall Guidelines

- Name-value pair naming strategy.
  - Define a consistent name-value pair naming scheme for how you want to categorize your requests.
  - Standardize both names and values.
  - Define when and where name-value pairs will be set.
  - Use name-value pairs to describe requests. Set name-value pairs for collection when you begin processing a request.
  - Define other uses of name-value pairs that will be ignored by name-value pair collection. Set name-value pairs for other purposes as needed.
  - All name-value pairs fit in a single 4k block and so don't use verbose names and values if you need to have a large number of name-value pairs.

Overall Guidelines

- Name-value pair naming strategy.
  - Define a unique request name-value pair token. Ensure the token is unique across complexes such as “complex + processor + getpid()” ie sprintf(value,“%8c.%1c.%X”, ctkiptr->ic0comp, tpf_cpuid_to_ordinal(ecbptr()->ce1cpd), getpid()); Alternatively, this unique value could be passed in on the request.
  - Define how the unique request name-value pair will be transferred across all external interfaces (AOR, MQ, TCP/IP, off board processing, and etc). Most TPF interfaces will propagate the name-value pair on your behalf within the system (ECB create and etc, exceptions such as internal AOR use).

Overall Guidelines

- Name-value pair naming strategy.
  - Consider defining a transaction id name-value pair that can be used to group requests together for a broader view of the collected data. For example, a transaction id could be used across shopping, booking, and ticketing requests to understand the complete cost of interacting with an end user.
  - Define how the transaction id name-value pair will be passed across all external interfaces (AOR, MQ, TCP/IP, off board processing, and etc). Most TPF interfaces will propagate the name-value pair on your behalf within the system (ECB create and etc, exceptions such as internal AOR use).

Overall Guidelines

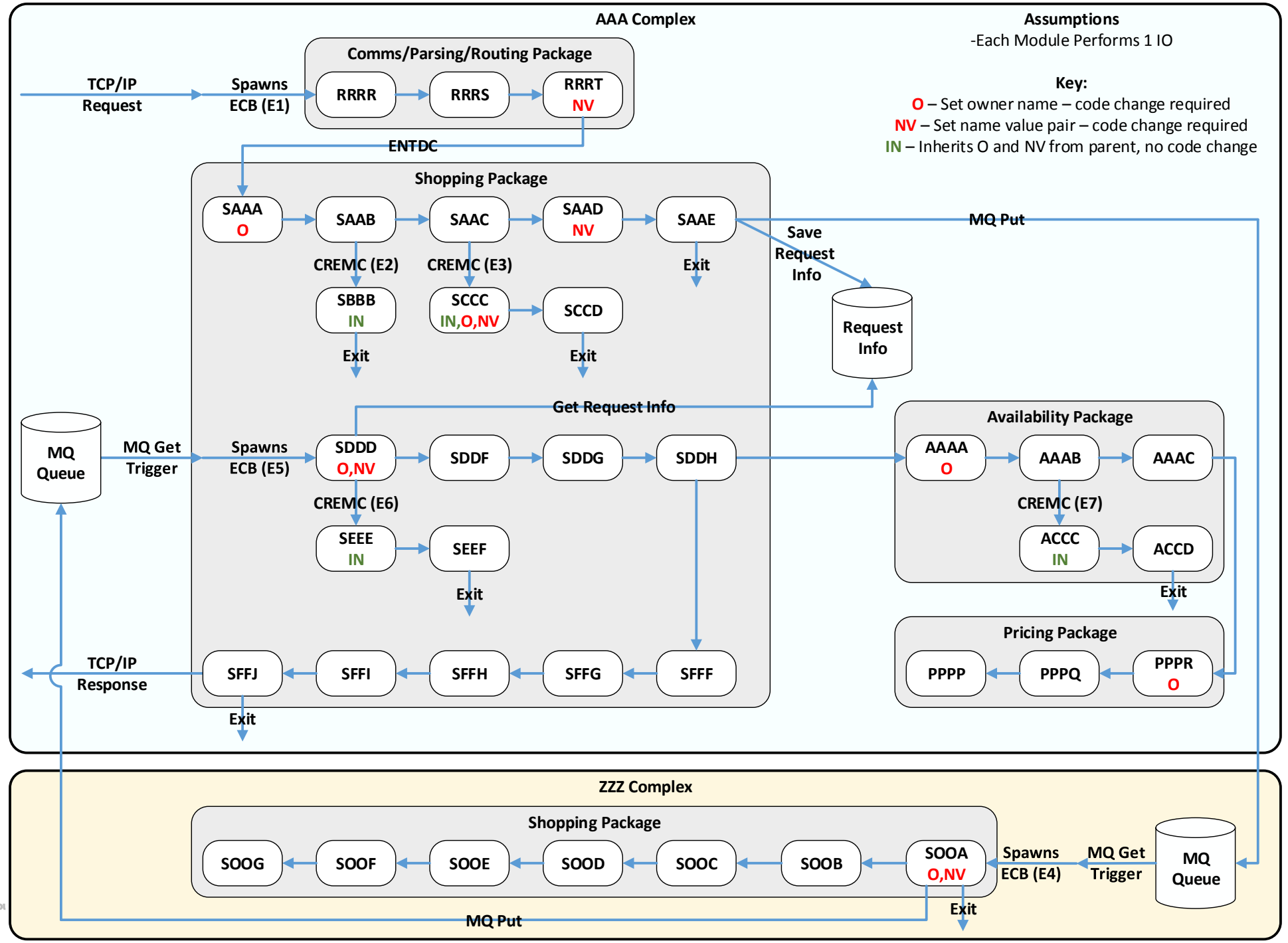
- Name-value pair naming strategy.
  - Consider name-value pair relationships. What name-value pairs will you want defined to subdivide another name-value pair? For example, consider the following hierarchy where each unique indented name-value pair name can be used to subdivide the one above.

```
Agency
   Geography
      Location
         Agent id
            Customer id
               Transaction Token
                  Message type
                     Message subtype
                        Unique Request Token
```

  - For example, a “shopping” message type can be subdivided into “Air Travel”, “Car Travel”, “Hotel Res” and etc.

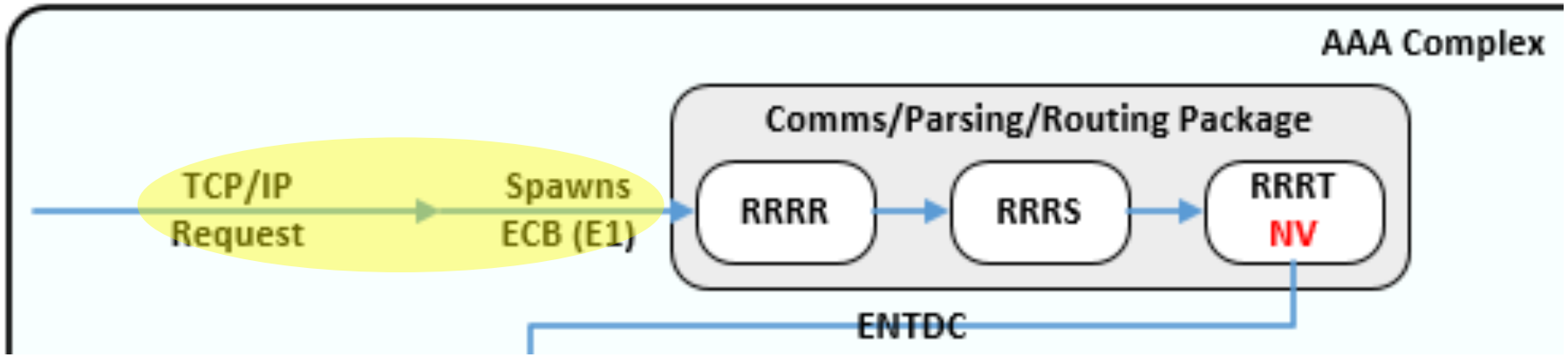
Real World Example

- This example will show a request being processed by multiple ECBs, calls between code packages, calls out to other systems, and etc.
- The purpose of this example is to demonstrate where customer code changes are required to set owner names and name-value pairs.
- This example also steps through how these mechanisms will work in detail when processing a request.



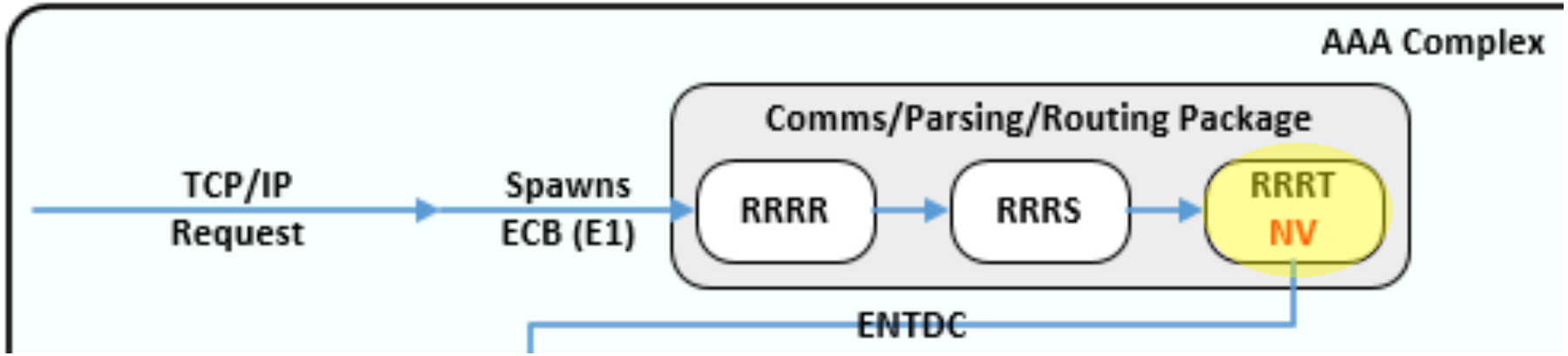
Real World Example

- A TCP/IP request spawns an ECB (E1).
- The ECB enters a code package that handles communications, parsing the input request and routing the ECB to the appropriate package.



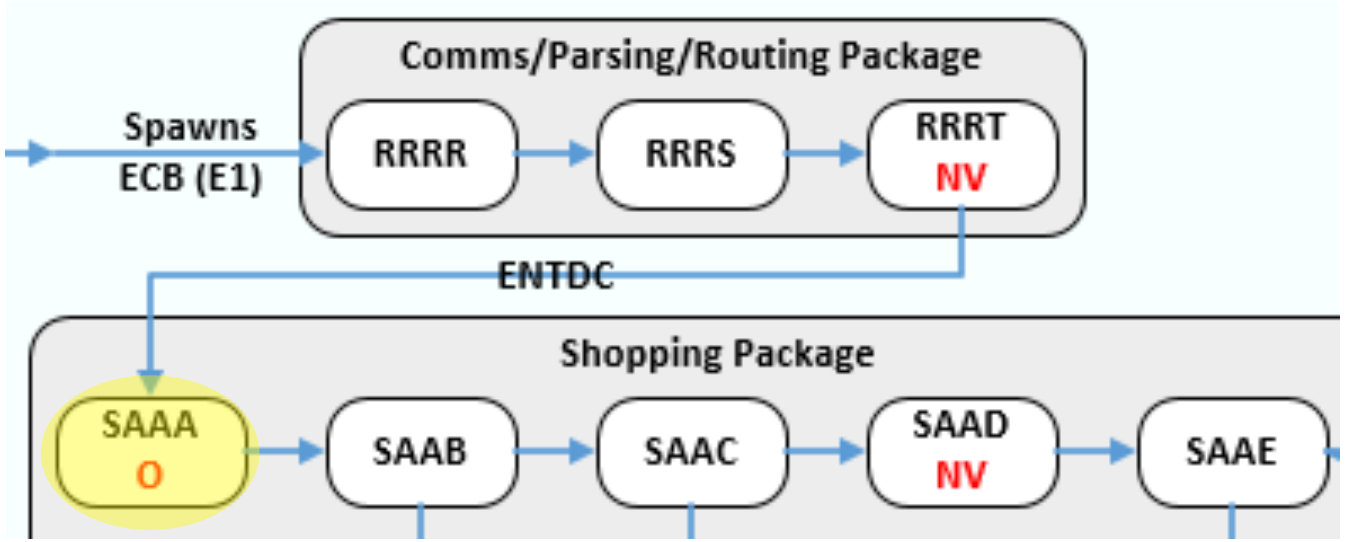
Real World Example

- As soon as the details of the request are understood (RRRT), the package sets a variety of name-value pairs that describe the input request:
  - name-value pairs: msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
  - Note that the msgId is the unique request identifier.



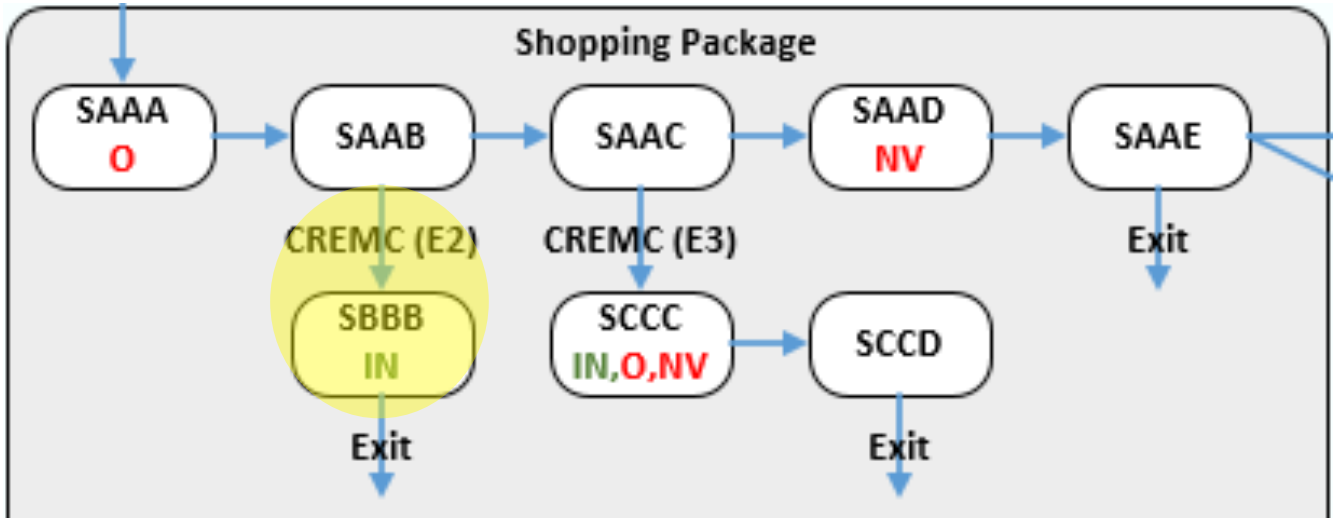
Real World Example

- The routing package (RRRT) does an ENTDC to an interface to the shopping package (SAAA).
- The shopping interface point sets the owner name
  - Owner: shop.air.domestic
- Since this is the first time the owner name is set on the ECB, the owner name counts are not written out. All resources used by the routing package are attributed to the shopping package.



Real World Example

- SAAB creates ECB (E2) executing program SBBB.
- ECB (E2) inherits the owner name and name-value pair settings.
  - name-value pairs: msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
  - Owner: shop.air.domestic

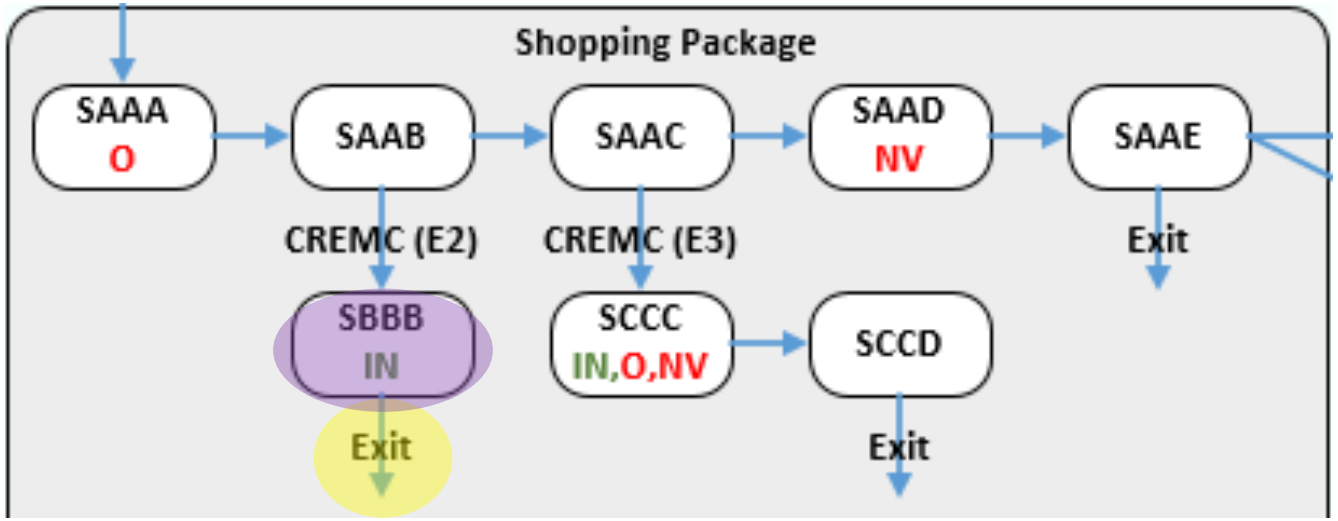


Real World Example

- ECB (E2) exit triggers collection

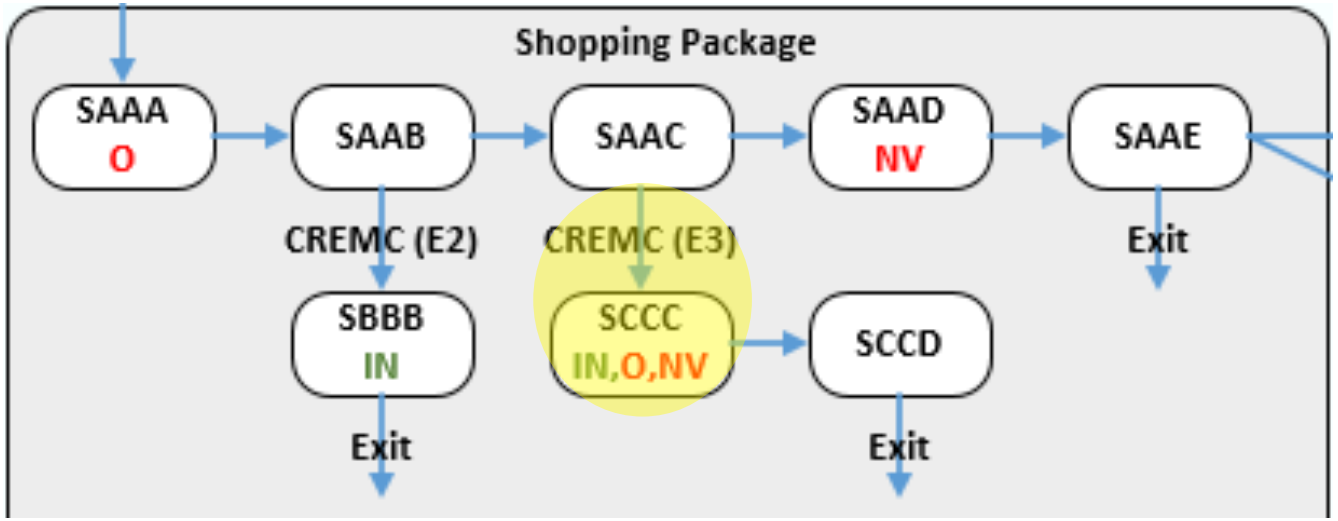
Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...

Complex AAA Owner Name Data	
Package	**IOs**
shop	1
avail	0
pricing	0
Total	**1**



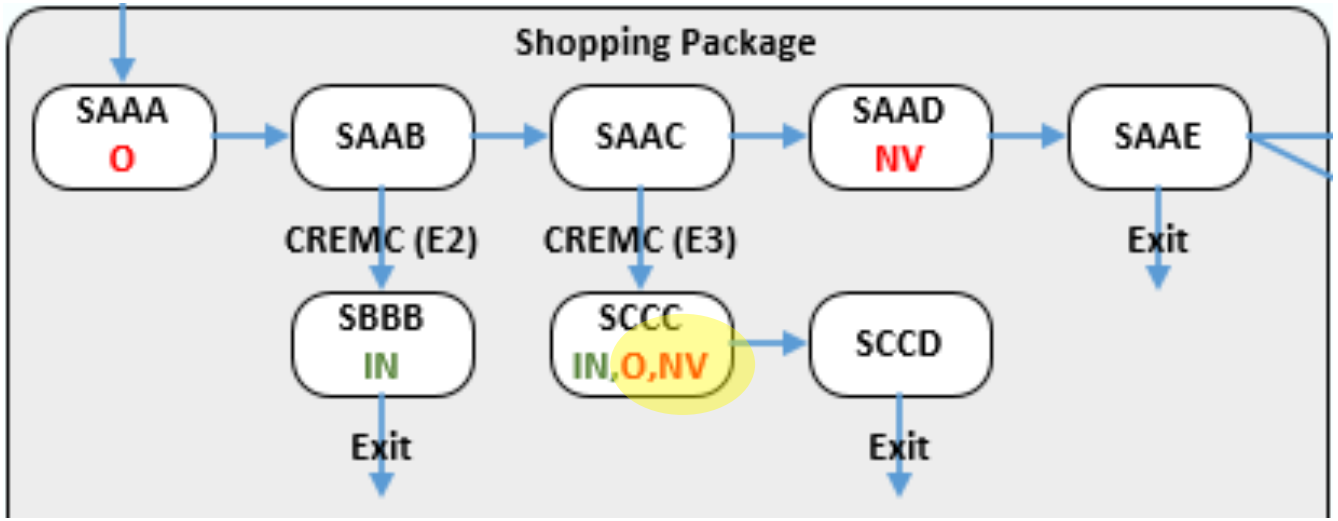
Real World Example

- SAAC creates ECB (E3) executing program SCCC.
- ECB (E3) inherits the owner name and name-value pair settings.
  - name-value pairs: msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
  - Owner: shop.air.domestic



Real World Example

- ECB (E3) executing program SCCC is a policing ECB that is not part of shopping message processing.
- ECB (E3) sets owner name and name-value pair settings.
  - name-value pairs: msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq, ...
  - Owner: shop.policing.

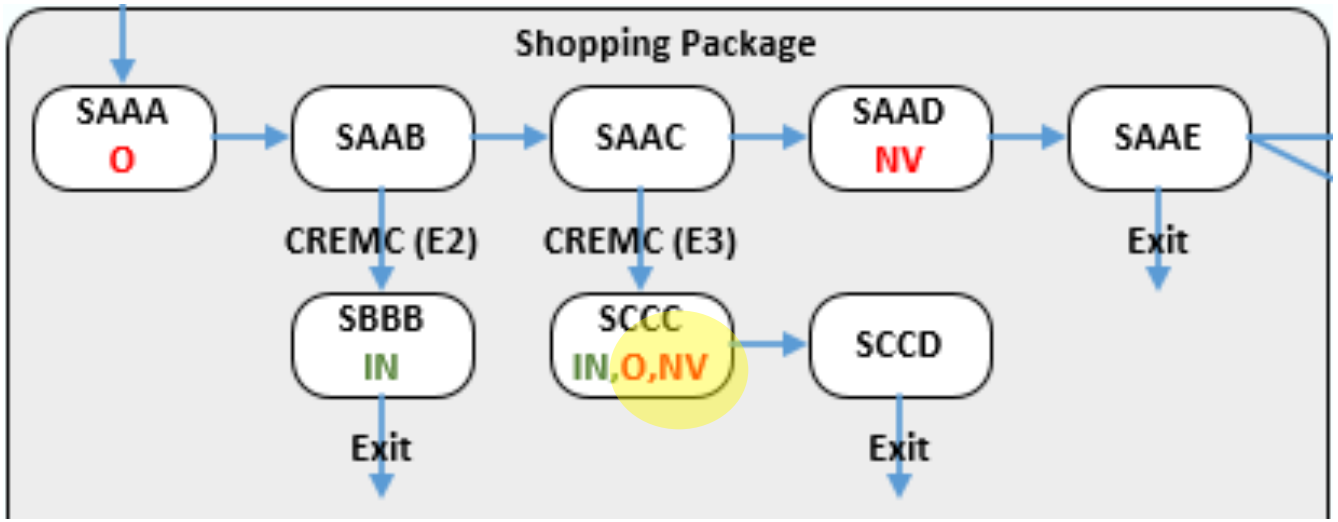


Real World Example

- Owner name change triggers collection. However, the owner name was set shortly after the ECB was created and minimal resources were used.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...

Complex AAA Owner Name Data	
Package	**IOs**
shop	1
avail	0
pricing	0
Total	**1**

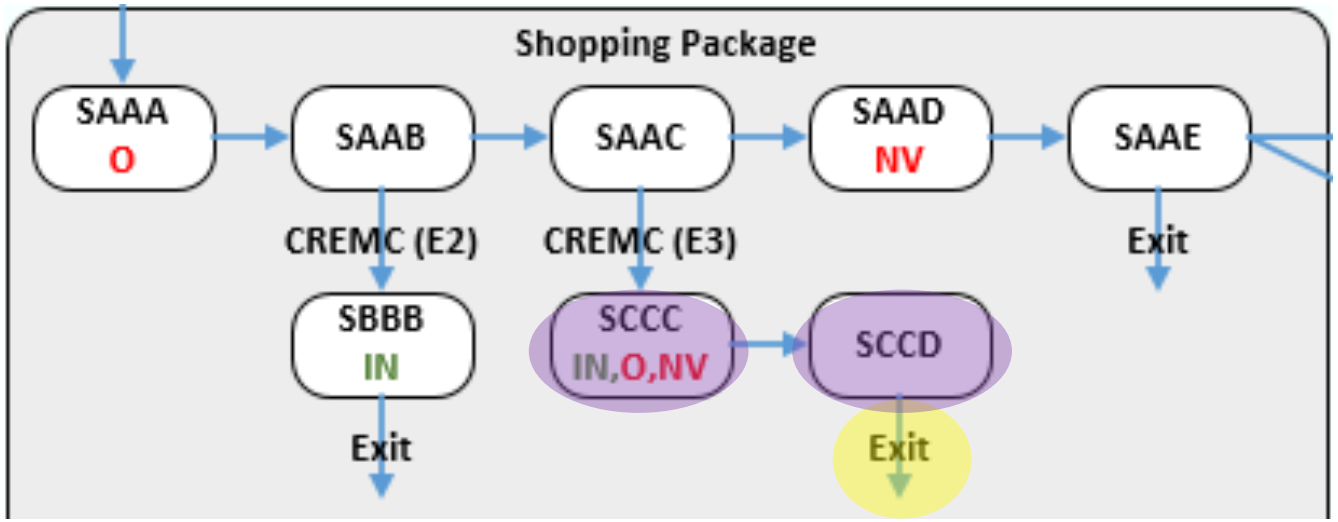


Real World Example

- ECB (E3) exit triggers collection

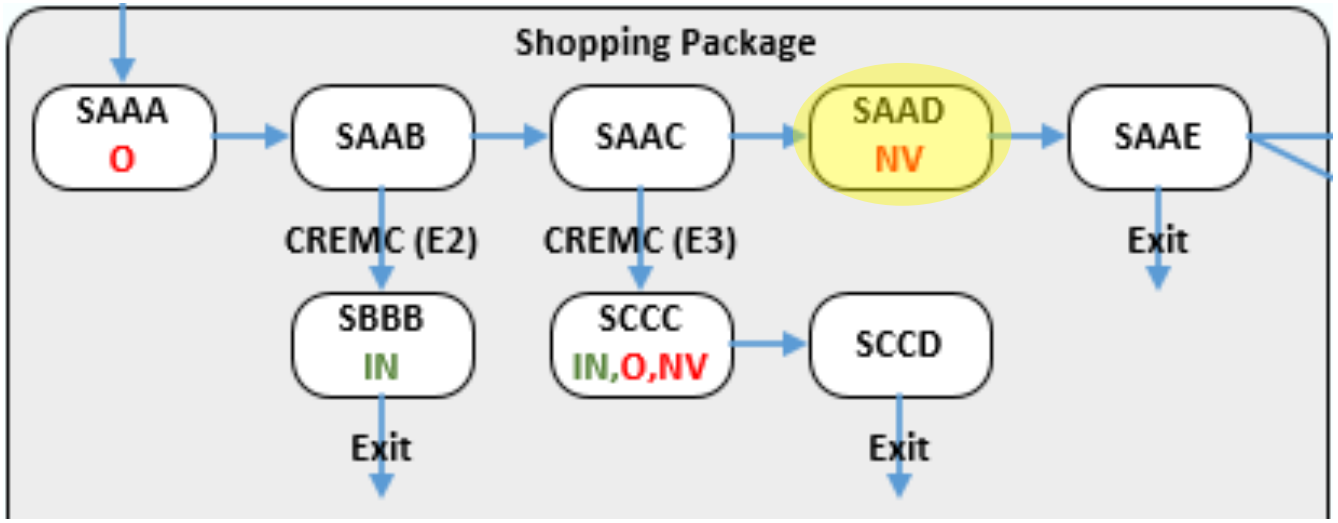
Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,

Complex AAA Owner Name Data	
Package	**IOs**
shop	3
avail	0
pricing	0
Total	**3**



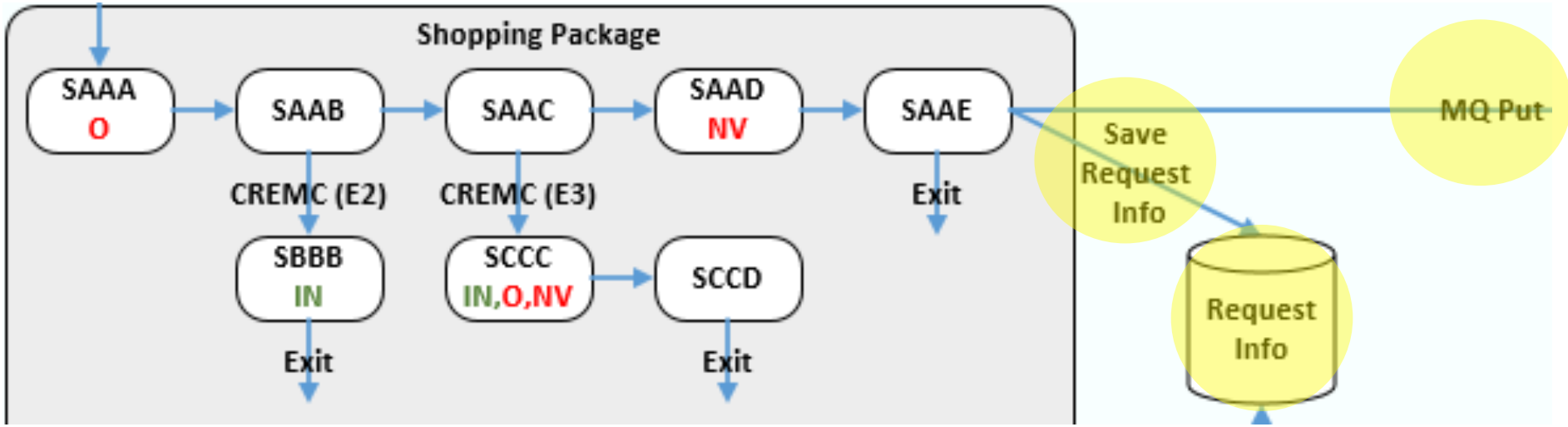
Real World Example

- SAAD sets a name-value pair for a user credential. This demonstrates how name-value pairs can be used for purposes other than name-value pair collection.
  - name-value pair: UserId-user1
- Note that child ECBs E2 and E3 were created prior to this name-value pair being set and will not have this additional name-value pair. Use caution setting name-value pairs of this type.
- It may be advisable to ensure these name-value pairs are not included in your name-value pair collection policy.



Real World Example

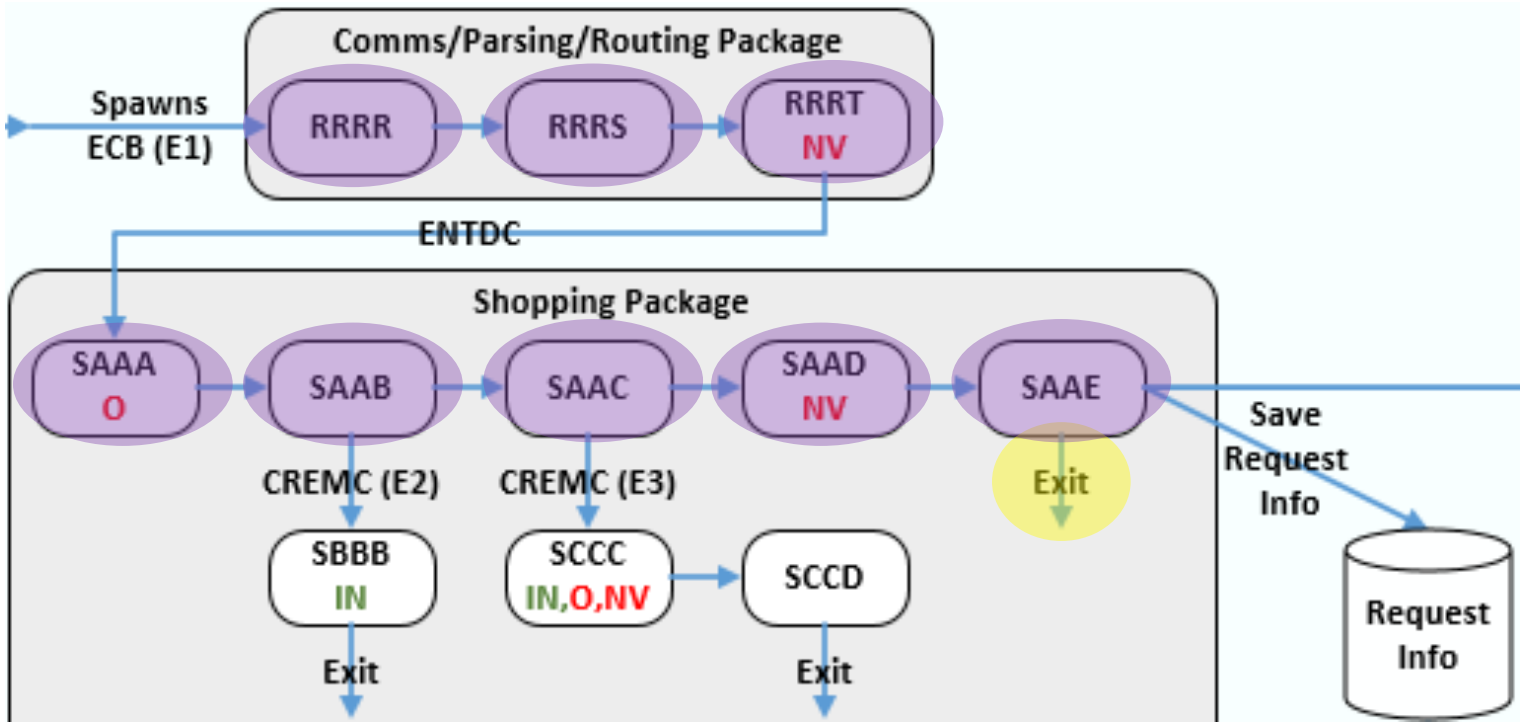
- Making an off box call.
  - SAAE saves name-value pairs by using the new tpf_NameValueLocalSave() API as well as other request information. An alternative to saving the name-value pair data is passing all of these details through the off box call.
  - SAAE sends name-value pairs in the MQ put request to another complex such as the unique request id name-value pair (msgId-AAA.B.F021). This allows for cross system name-value pair analysis.



Real World Example

- ECB (E1) exit triggers collection

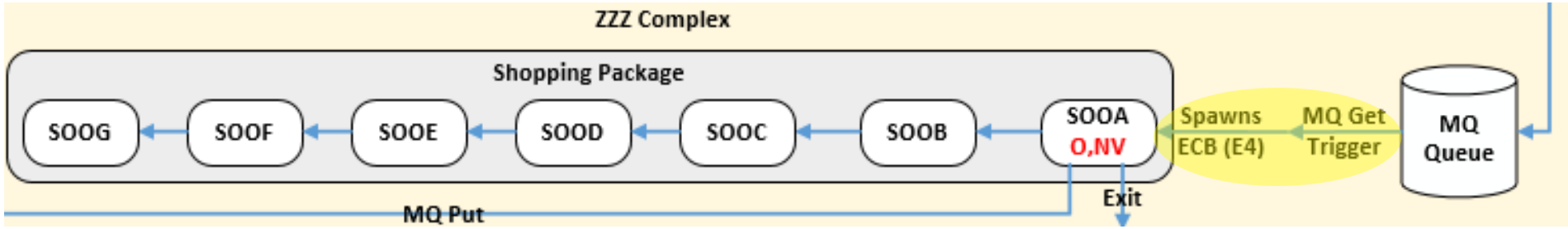
Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...



Complex AAA Owner Name Data	
Package	**IOs**
shop	11
avail	0
pricing	0
Total	**11**

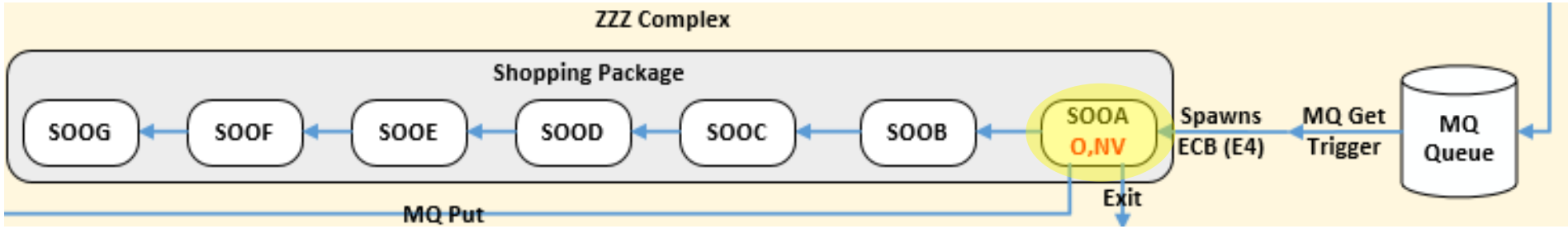
Real World Example

- Off box processing.
  - The message arrival on the queue triggers the creation of ECB (E4) which enters SOOA of the shopping package.



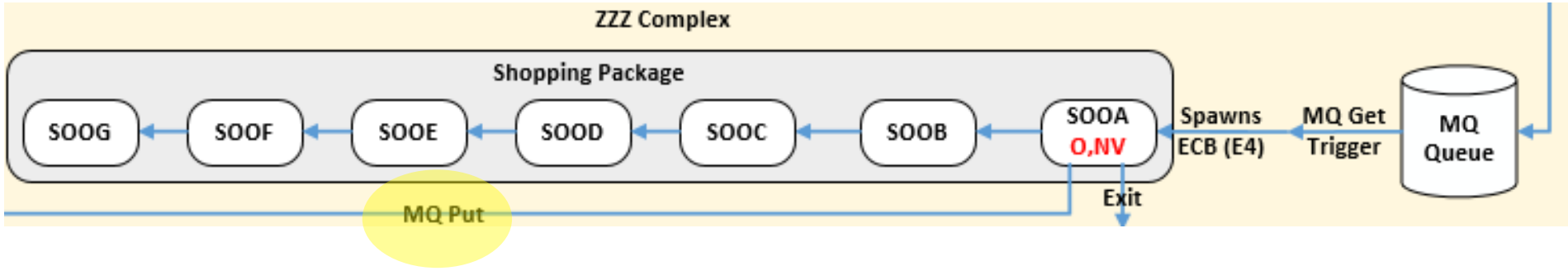
Real World Example

- Off box processing.
  - SOOA is an interface point to the shopping package and as such the owner name is set. Since this is the first time the owner name is set on the ECB, the owner name counts are not written out.
    - Owner name: shop.air.coOrd
  - The request read from the queue indicates a unique message name-value pair. These name-value pairs are set using the tpf_NameValueLocalRestore() API. (note the presence of unique message id and credential name-value pair):
    - Name-value pairs: msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA, UserId-user1 ...



Real World Example

- Off box processing.
  - The other processing in the package is completed.
  - SOOA sends critical name-value pairs in the MQ put request to origin complex such as the unique request id name-value pair (msgId-AAA.B.F021). This allows for cross system name-value pair analysis.



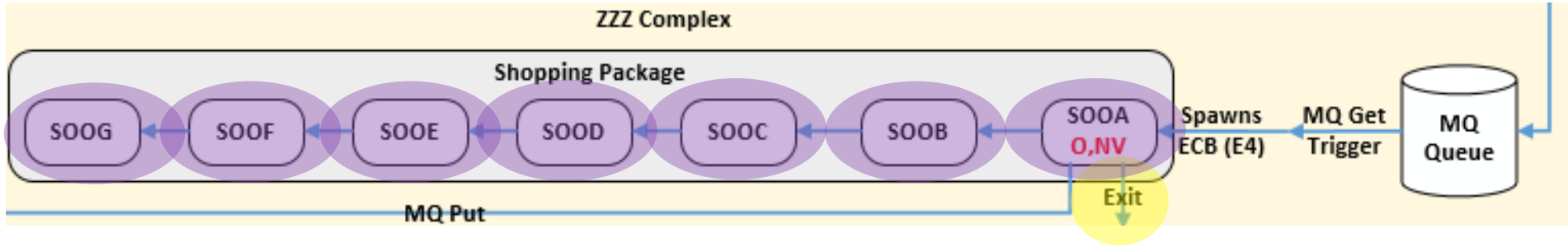
Real World Example

- ECB (E4) exit triggers collection. Note that since the user credential name-value pair was not in the policy, it is not included in the name-value pair collection data.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E4 (External SOOA)	Exit	7	shop.air.coOrd	msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA

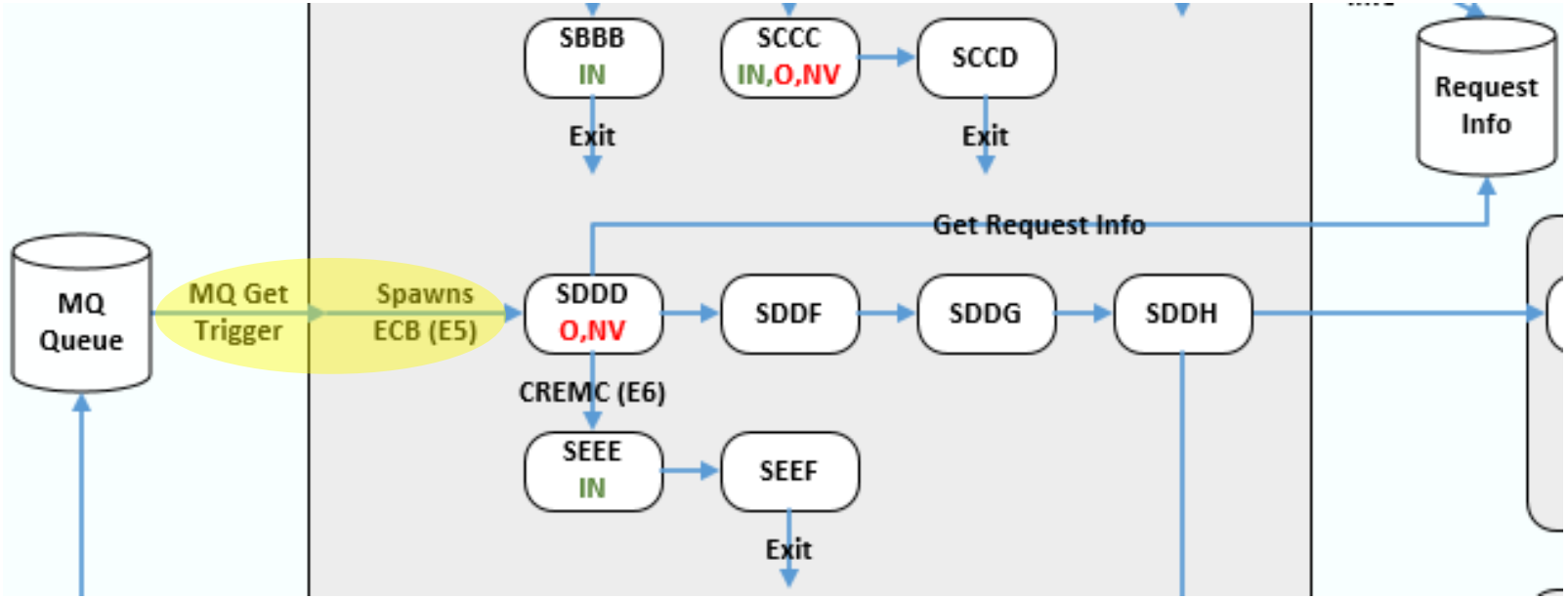
Complex AAA Owner Name Data	
Package	**IOs**
shop	11
avail	0
pricing	0
Total	**11**

Complex ZZZ Owner Name Data	
Package	**IOs**
shop	7
Total	**7**



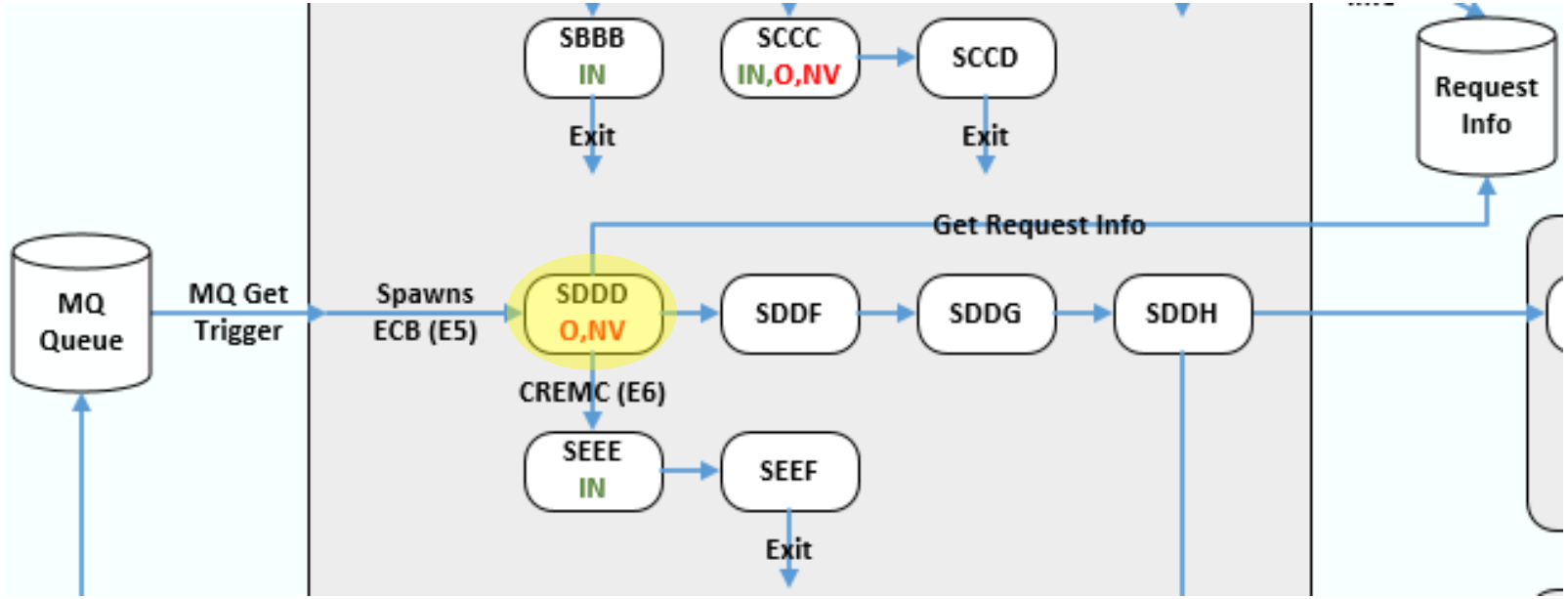
Real World Example

- The message arrival on the queue triggers the creation of ECB (E5) which enters SDDD of the shopping package.



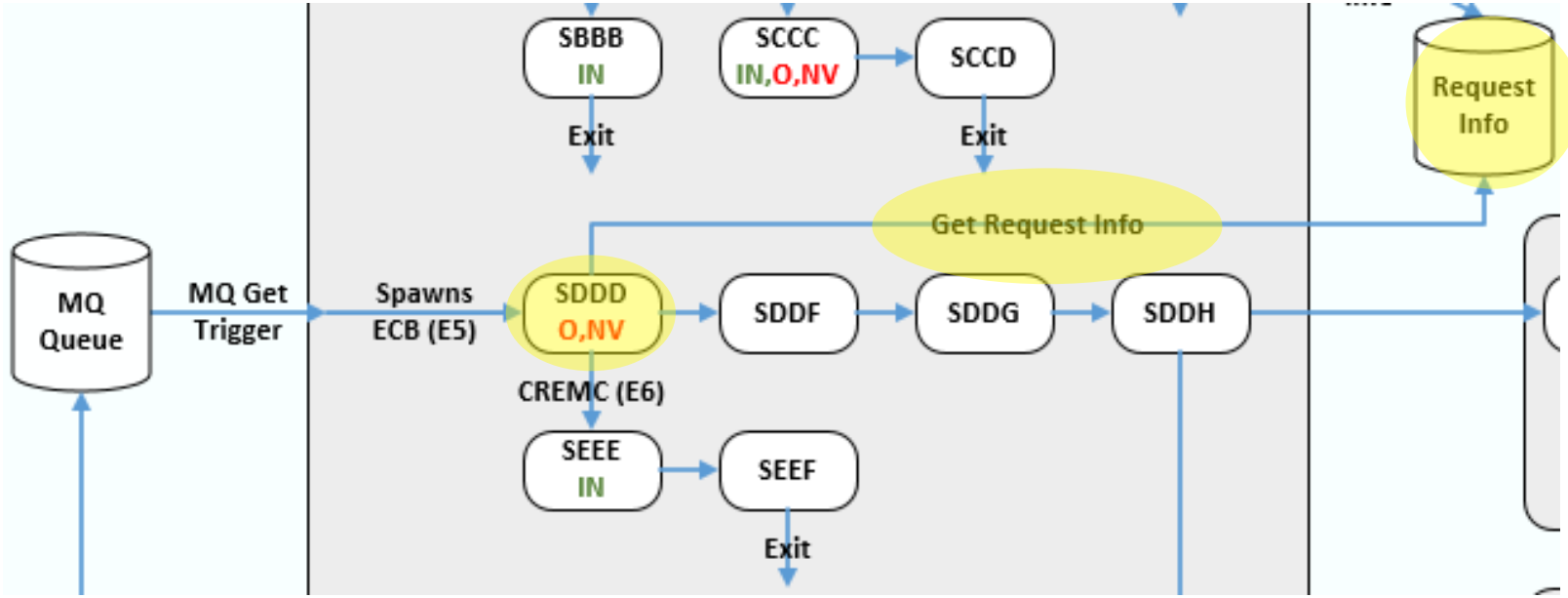
Real World Example

- SDDD is an interface point to the shopping package and as such the owner name is set. Since this is the first time the owner name is set on the ECB, the owner name counts are not written out.
  - Owner name: shop.air.domestic



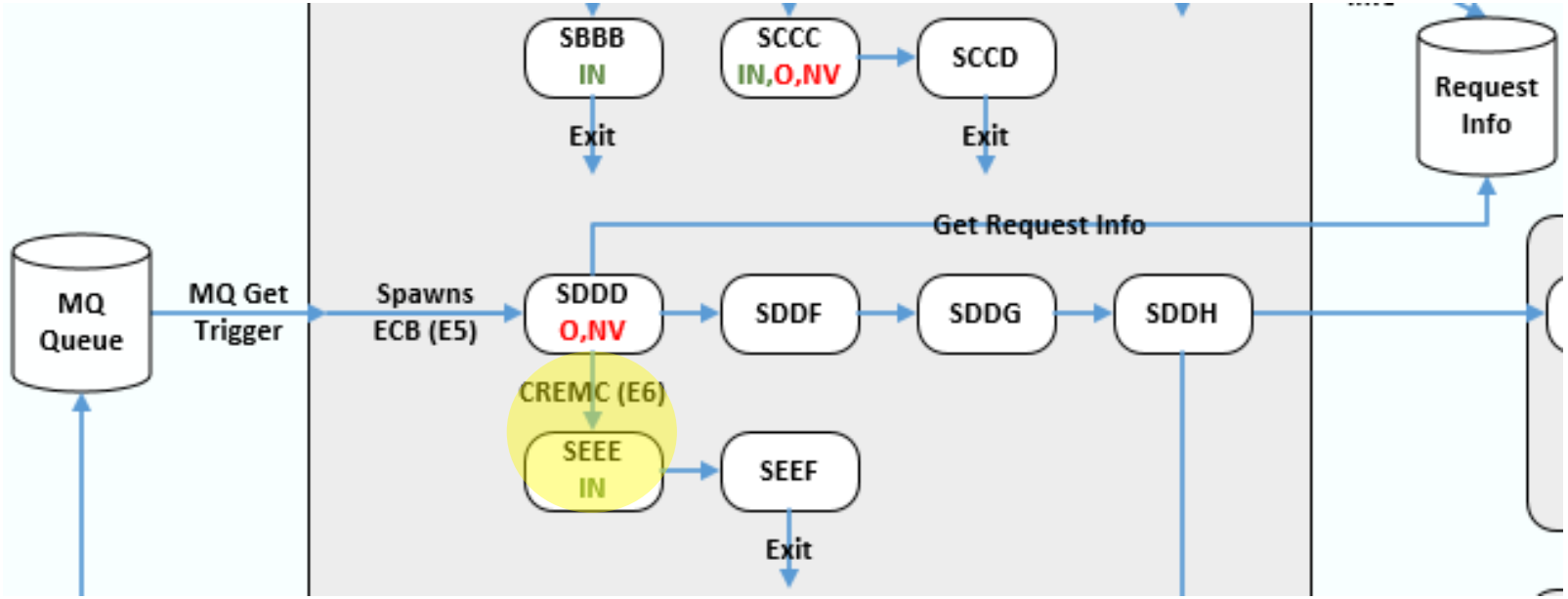
Real World Example

- SDDD retrieves critical request information and name-value pairs from storage. From this, the request protocol and etc, SDDD sets the name-value pairs using the tpf_NameValueLocalRestore() API.
  - Name-value pairs: msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, UserId-user1...



Real World Example

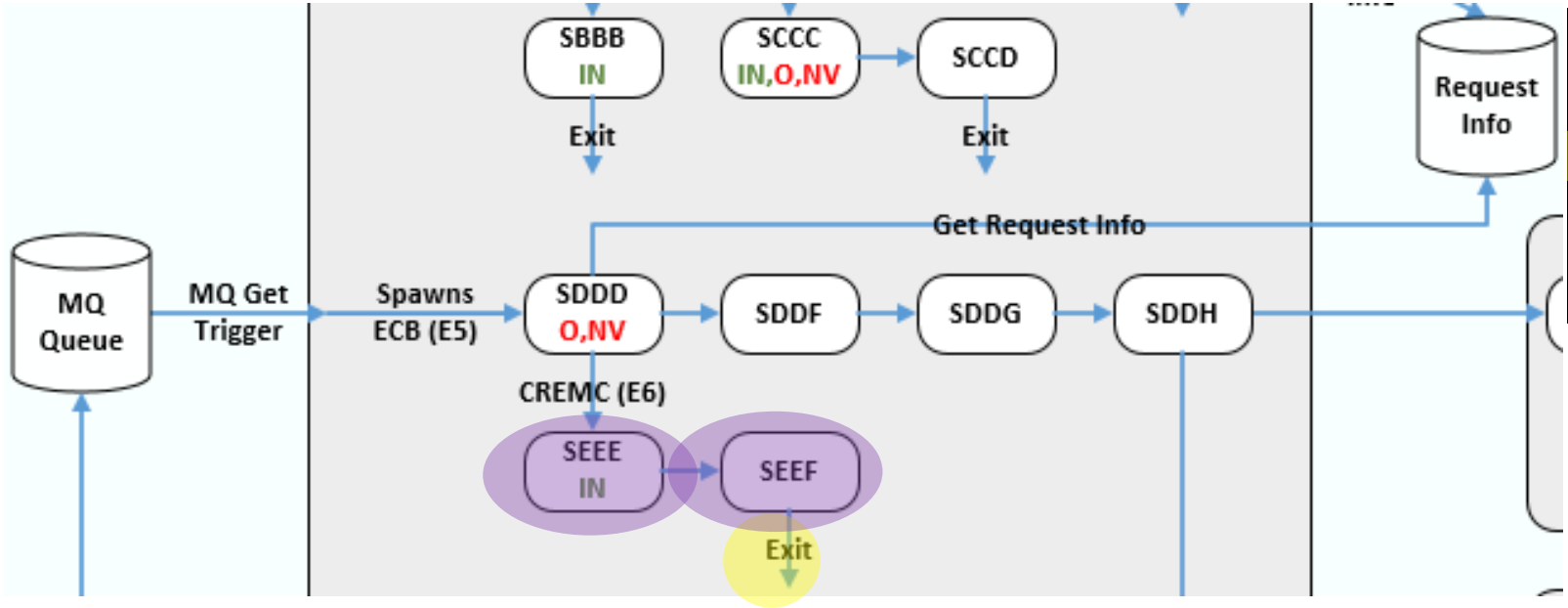
- SDDD creates ECB (E6) executing program SEEE.
- ECB (E6) inherits the owner name and name-value pair settings.
  - name-value pairs: msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, UserId-user1...
  - Owner: shop.air.domestic



Real World Example

- ECB (E6) exit triggers collection.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E4 (External SOOA)	Exit	7	shop.air.coOrd	msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA
E6 (CREMC SEEE)	Exit	2	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...

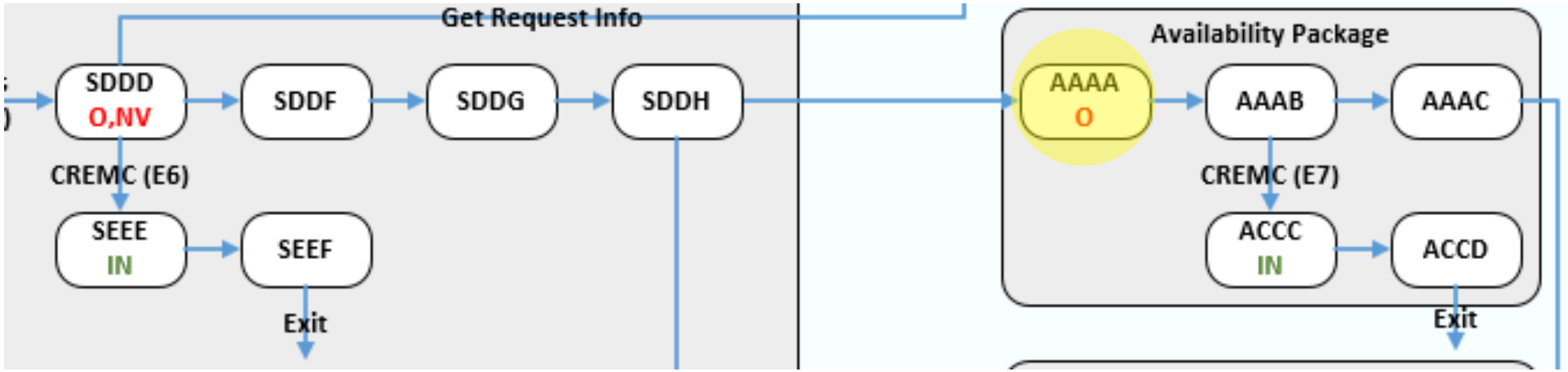


Complex AAA Owner Name Data	
Package	**IOs**
shop	13
avail	0
pricing	0
Total	**13**

Complex ZZZ Owner Name Data	
Package	**IOs**
shop	7
Total	**7**

Real World Example

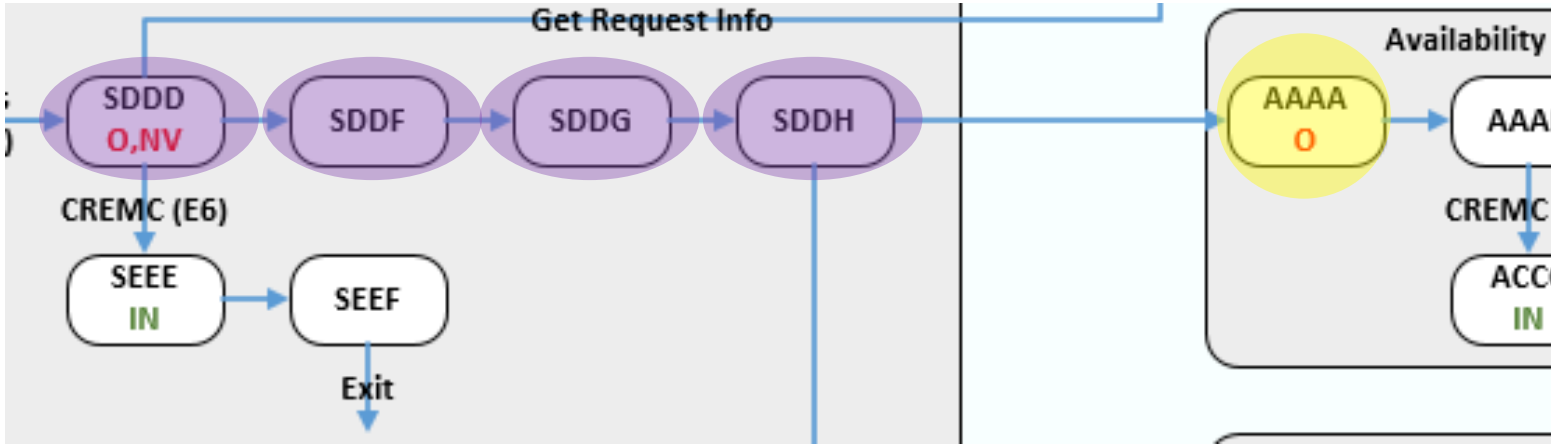
- SDDH calls the Availability package.
- AAAA is an interface point to the availability package and as such the owner name is set.
  - Owner name: avail.air.domestic



Real World Example

- Owner name change triggers collection.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E4 (External SOOA)	Exit	7	shop.air.coOrd	msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA
E6 (CREMC SEEE)	Exit	2	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	4	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...

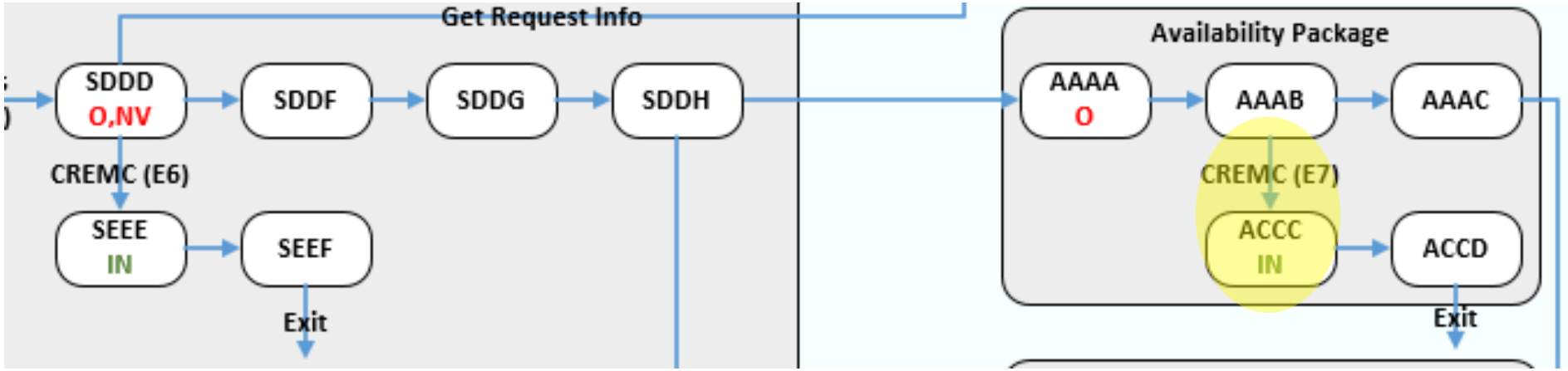


Complex AAA Owner Name Data	
Package	**IOs**
shop	17
avail	0
pricing	0
Total	**17**

Complex ZZZ Owner Name Data	
Package	**IOs**
shop	7
Total	**7**

Real World Example

- AAAB creates ECB (E7) executing program ACCC.
- ECB (E7) inherits the owner name and name-value pair settings.
  - name-value pairs: msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, UserId-user1...
  - Owner: avail.air.domestic



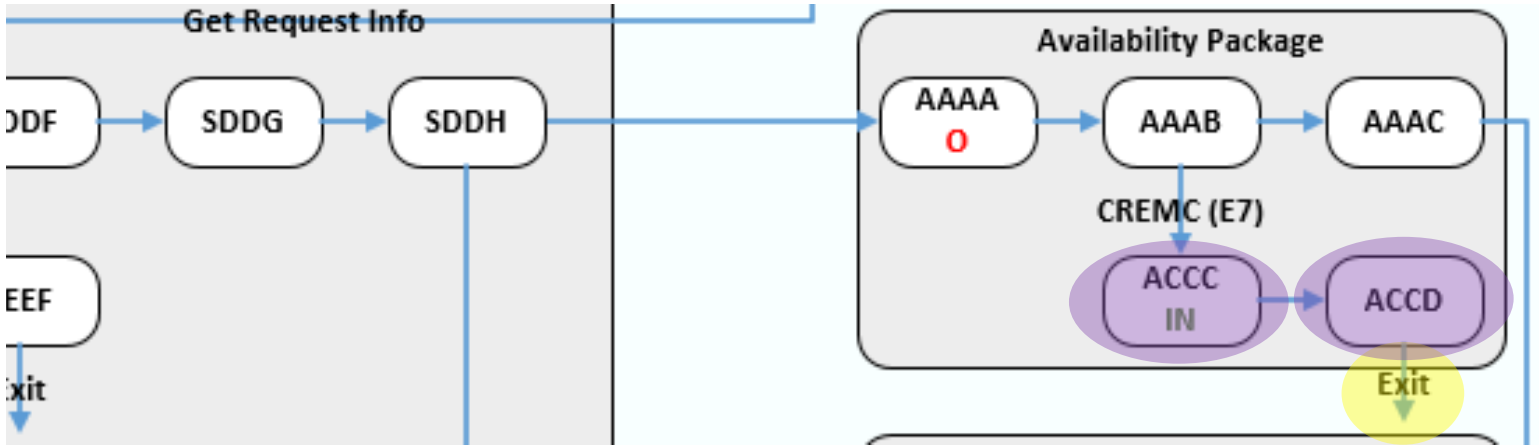
Real World Example

- ECB (E7) exit triggers collection.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E4 (External SOOA)	Exit	7	shop.air.coOrd	msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA
E6 (CREMC SEEE)	Exit	2	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	4	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E7 (CREMC ACCC)	Exit	2	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...

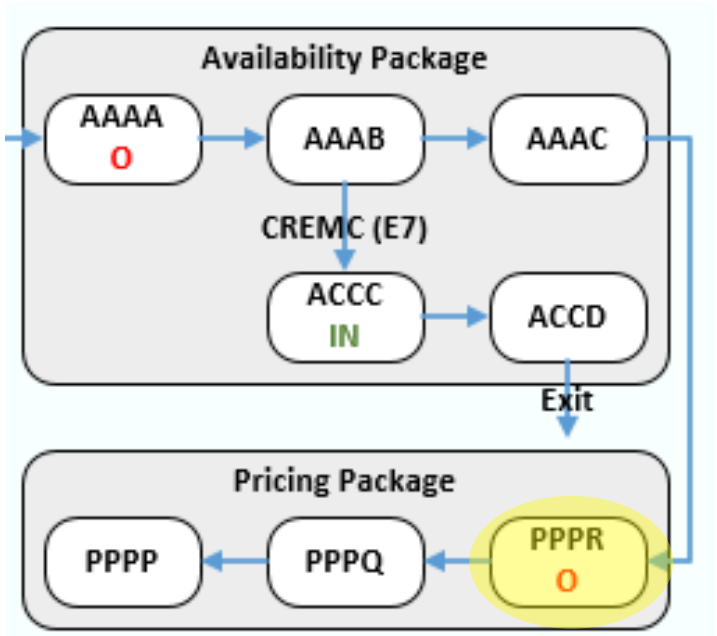
Complex AAA Owner Name Data	
Package	**IOs**
shop	17
avail	2
pricing	0
Total	**19**

Complex ZZZ Owner Name Data	
Package	**IOs**
shop	7
Total	**7**



Real World Example

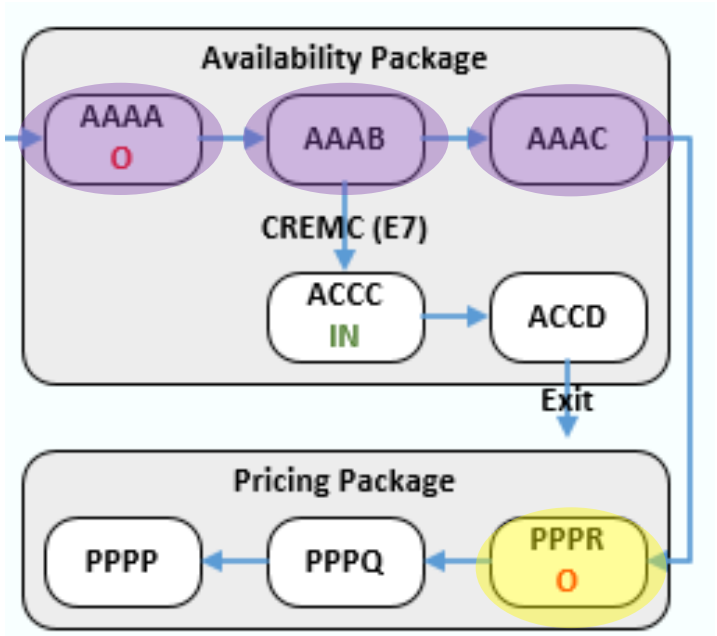
- AAAC calls the Pricing package.
- PPPR is an interface point to the pricing package and as such the owner name is set.
  - Owner name: pricing.air.domestic



Real World Example

- Owner name change triggers collection.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E4 (External SOOA)	Exit	7	shop.air.coOrd	msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA
E6 (CREMC SEEE)	Exit	2	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	4	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E7 (CREMC ACCC)	Exit	2	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	3	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...

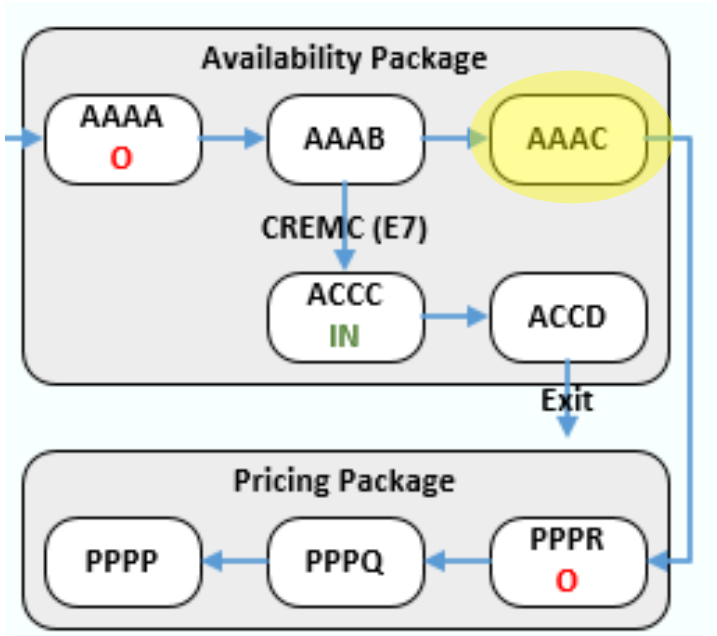


Complex AAA Owner Name Data	
Package	**IOs**
shop	17
avail	5
pricing	0
Total	**22**

Complex ZZZ Owner Name Data	
Package	**IOs**
shop	7
Total	**7**

Real World Example

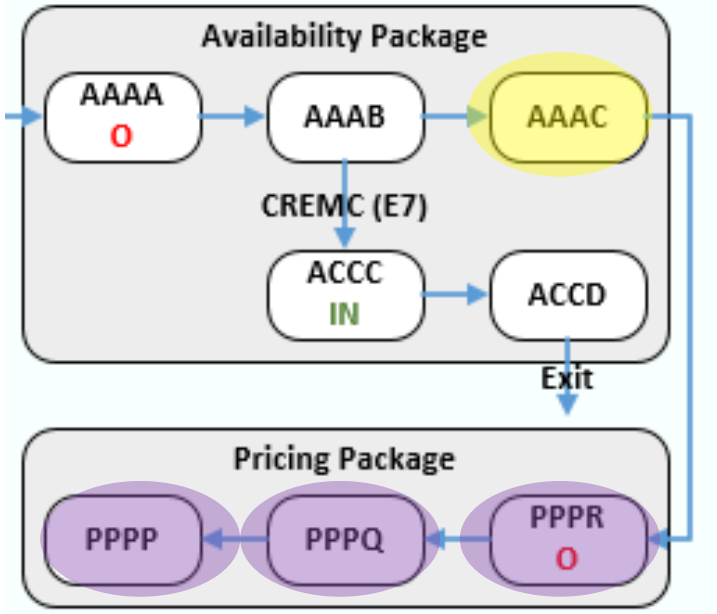
- Pricing package processing completes
- Upon return to the availability package, the system automatically restores the owner name.
  - Owner name: avail.air.domestic



Real World Example

- Owner name restore triggers collection.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E4 (External SOOA)	Exit	7	shop.air.coOrd	msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA
E6 (CREMC SEEE)	Exit	2	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	4	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E7 (CREMC ACCC)	Exit	2	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	3	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Reset	3	pricing.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...

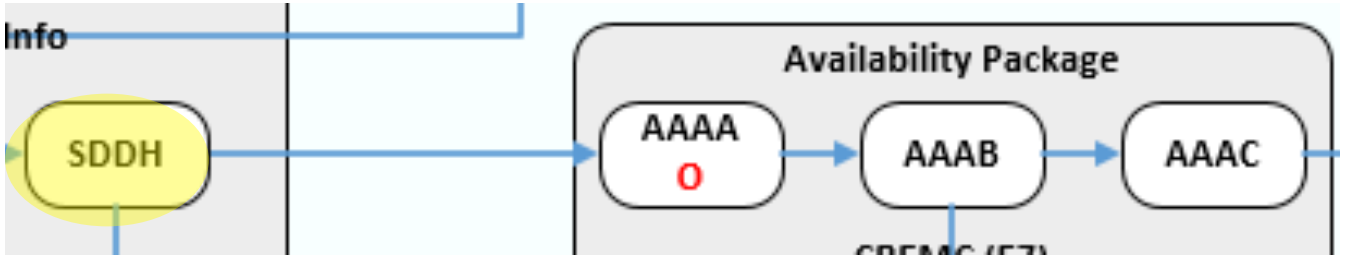


Complex AAA Owner Name Data	
Package	**IOs**
shop	17
avail	5
pricing	3
Total	**25**

Complex ZZZ Owner Name Data	
Package	**IOs**
shop	7
Total	**7**

Real World Example

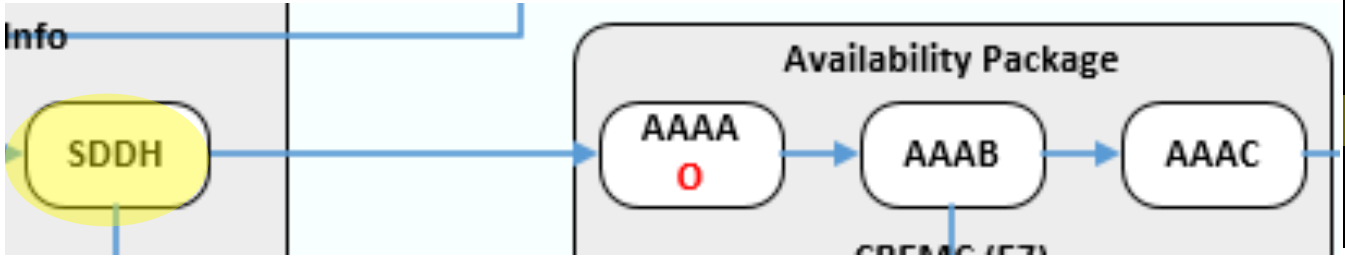
- Availability package processing completes
- Upon return to the shopping package, the system automatically resets the owner name.
  - Owner name: shop.air.domestic



Real World Example

- owner name restore triggers collection. In our example, no additional IOs are recorded. However, if they were, they would be written to the avail owner name bucket.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E4 (External SOOA)	Exit	7	shop.air.coOrd	msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA
E6 (CREMC SEEE)	Exit	2	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	4	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E7 (CREMC ACCC)	Exit	2	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	3	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Reset	3	pricing.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Reset	0	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...



Complex AAA Owner Name Data	
Package	**IOs**
shop	17
avail	5
pricing	3
Total	**25**

Complex ZZZ Owner Name Data	
Package	**IOs**
shop	7
Total	**7**

Real World Example

- The shopping request processing completes.
- A response is sent to the requester.

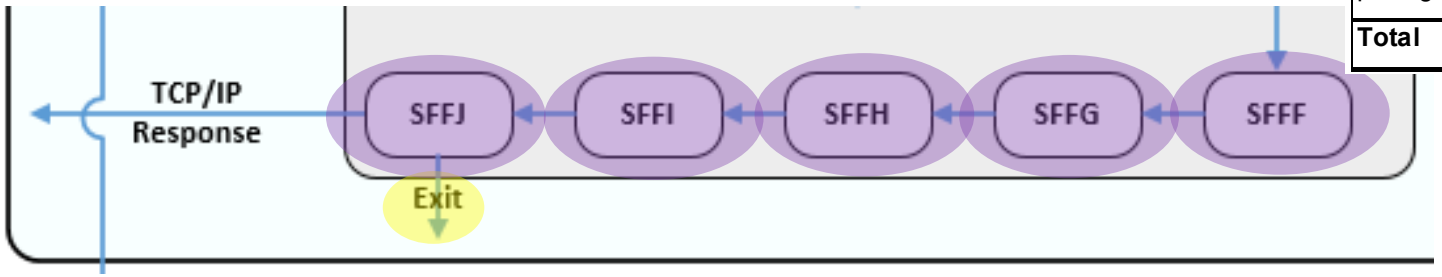
Real World Example

- ECB (E5) exit triggers collection.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq,
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E4 (External SOOA)	Exit	7	shop.air.coOrd	msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA
E6 (CREMC SEEE)	Exit	2	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	4	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E7 (CREMC ACCC)	Exit	2	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	3	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Reset	3	pricing.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Reset	0	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Exit	5	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...

Complex AAA Owner Name Data	
Package	**IOs**
shop	22
avail	5
pricing	3
Total	**30**

Complex ZZZ Owner Name Data	
Package	**IOs**
shop	7
Total	**7**



Real World Example

- Here is the full list of name-value pair collection entries organized by ECB.

Name value pair collection entry (as seen offline, includes data from all complexes)				
ECB	**Events**	**IOs**	**Owner Name**	**Name-Value Pairs**
E1 (RRRR)	Exit	8	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E2 (CREMC SBBB)	Exit	1	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Owner Change	0	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E3 (CREMC SCCC)	Exit	2	shop.policing.	msgId-AAA.B.9527, msgType-policing, msgSub-shop, origin-shopAirReq, ...
E4 (External SOOA)	Exit	7	shop.air.coOrd	msgId-AAA.B.F021, msgType-shop, msgSub-airCoOrd, origin-CompAAA ...
E5 (SDDD)	Owner Change	4	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Change	3	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Reset	3	pricing.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Owner Reset	0	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E5 (SDDD)	Exit	5	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E6 (CREMC SEEE)	Exit	2	shop.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
E7 (CREMC ACCC)	Exit	2	avail.air.domestic	msgId-AAA.B.F021, msgType-shop, msgSub-air, origin-web, ...
Total		**37**		

Complex AAA Owner Name Data	
Package	**IOs**
shop	22
avail	5
pricing	3
Total	**30**

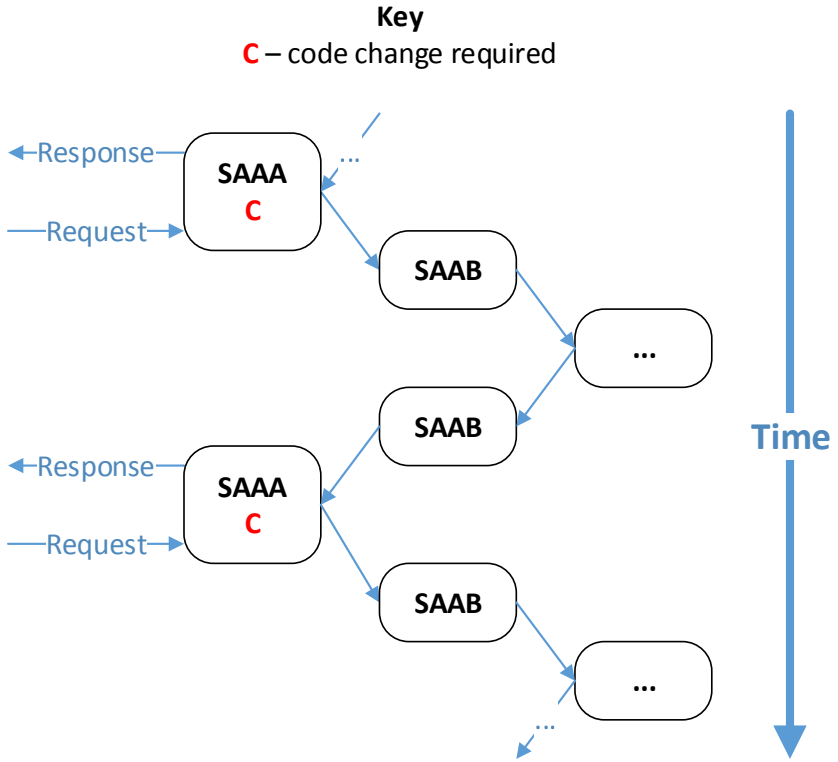
Complex ZZZ Owner Name Data	
Package	**IOs**
shop	7
Total	**7**

Real World Example

- The following table shows the IO count breakdown by collection mechanism.
  - Notice that the owner name totals across the two complexes are reflected in totality in the name value pair collection results.
  - Notice that the policing ECB resources are obscured in the shop owner name results on complex AAA but are separated out in the name value pair collection results.

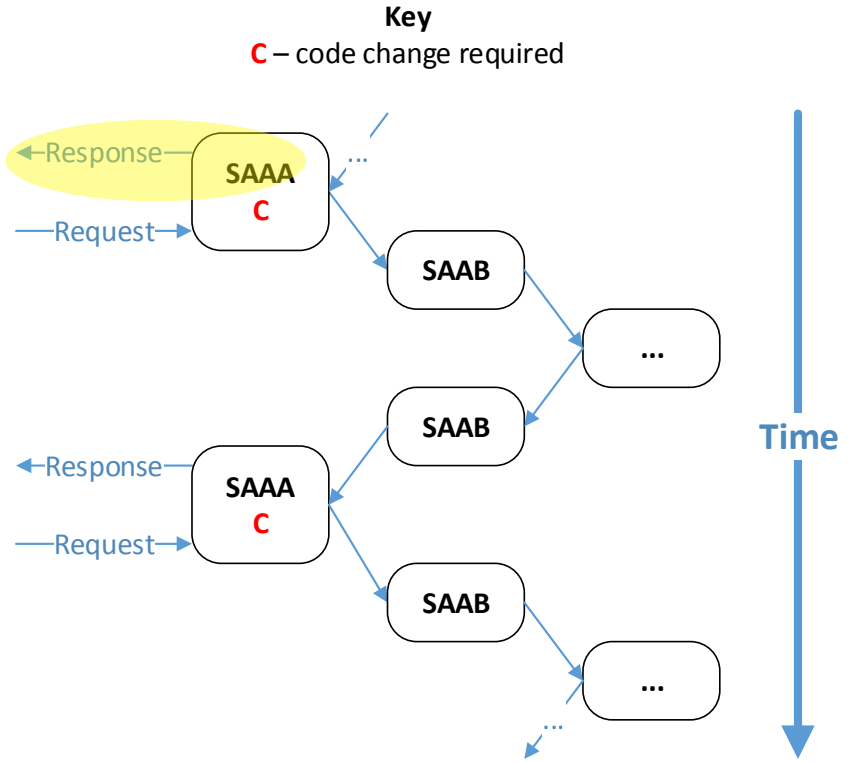
What did each ECB actually do? (not a real report)		Complex AAA Owner Name Data		Complex ZZZ Owner Name Data		Name Value View (cross complex analysis)			
ECB	**IOs**	**Package**	**IOs**	**Package**	**IOs**			**Owner Breakdown**	
E1 (RRRR)	8	shop	22	shop	7	**msgType**	**IOs**	**owner**	**IOs**
E2 (CREMC SBBB)	1	avail	5	**Total**	**7**	shop	35	shop	27
E3 (CREMC SCCC)	2	pricing	3					avail	5
E4 (External SOOA)	7	**Total**	**30**					pricing	3
E5 (SDDD)	15					policing	2	shop	2
E6 (CREMC SEEE)	2					**Total**	**37**		**37**
E7 (CREMC ACCC)	2								
Total	**37**								

One ECB for multiple requests Example



- In this example, a single ECB will be used to process multiple requests.
- Applications that follow this programming model should be updated to utilize the start request API once it is available.

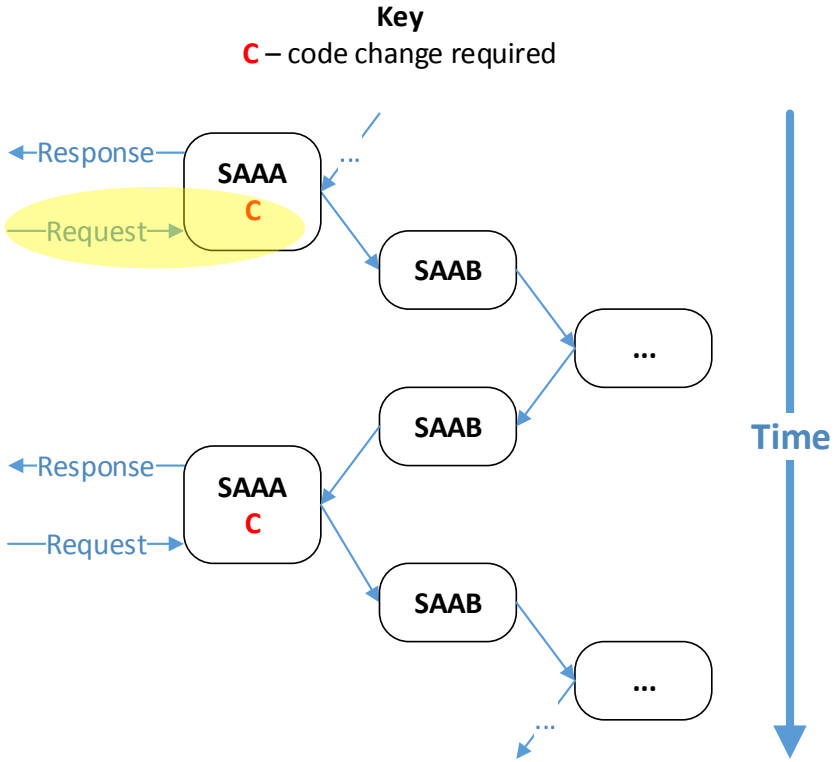
One ECB for multiple requests Example



- After the response is sent and clean up for the previous request is complete, the start request processing API is called to
  - flush out the name limit, owner name and name-value pair counters to collection (as if the ECB exited).
  - set all ECB counters to zero.
  - reset name limit, owner name and name-value pairs to the system defaults.

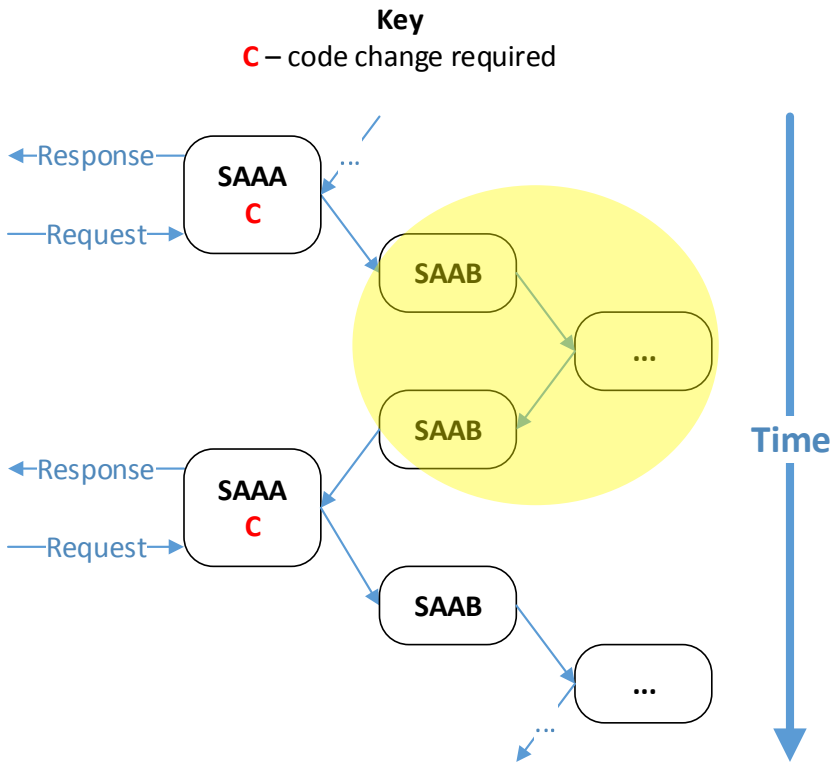
One ECB for multiple requests Example

- When the next request is received the application sets the owner name and name-value pairs that are appropriate.

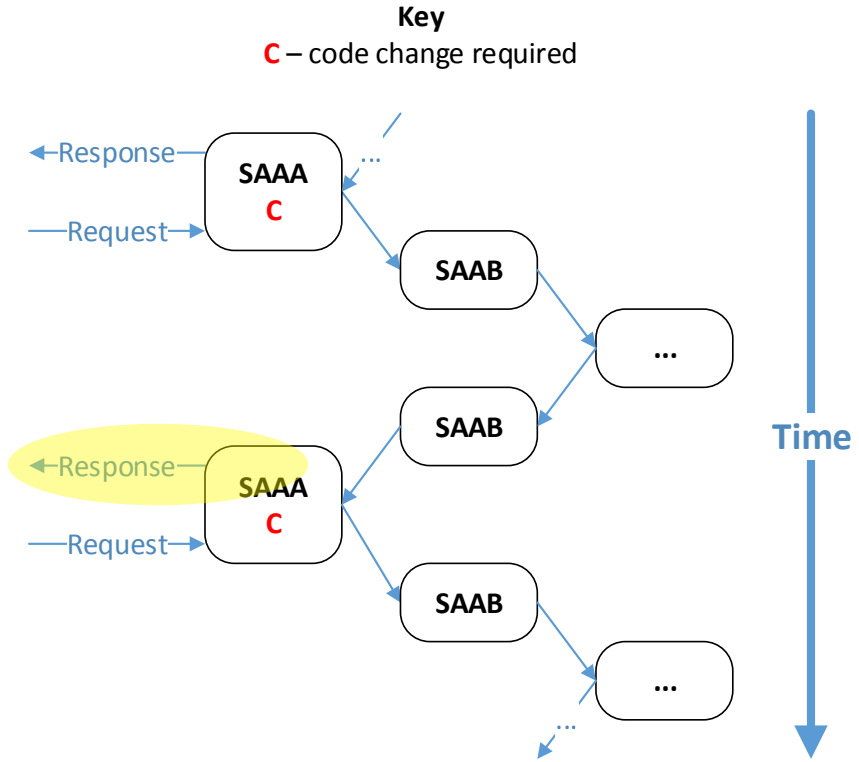


One ECB for multiple requests Example

- The request is processed.



One ECB for multiple requests Example



- After the response is sent and clean up for the previous request is complete, the start request processing API is called.
- And so on...

Business Value of Existing Mechanisms: Today

- Named limit set: Understand what resources request processing is using for the ECB Resource monitor to prevent outages.
- ECB owner names: Understand what resources code packages are using.
- Block owner names: Understand what blocks code packages are using.
- Name-value pairs: Make data available throughout the application (credentials, request source, request details, etc).

Business Value of Existing Mechanisms: To be

- ECB owner names: More accurate understanding of what resources code packages are using. More granularity of resources used in ADI analysis.

- Name-value pairs: Make data available throughout the application. Name-value pair collection in ADI provides greater depth of understanding of resource usage than existing tools.

Business Value of ADI and NVP Collection

- IBM Application Delivery Intelligence (ADI) is an Enterprise continuous integration analytics dashboard and optimization software with focus on application understanding, performance and quality metrics and trends.

- The name-value pair collection feature will be implemented in ADI so that:
  - A coverage programmer can use name-value pair collection to gain new insights into system resource usage and identify the source of problems in as little as 1/20 of the time previously required.

Call to action

- Update your code today
  - to set name-value pairs when a request is received to describe the request to be processed, the origin of the request, and provide unique ids that identify units of work.
  - to set owner names at the primary interface points to code packages.
  - to leverage the immediate benefits of using name-value pairs to pass data, owner name collection (ZMOWN) and etc.
  - to prepare to leverage name-value pair collection once it is available.

More Information

- Future discussions of the 2016 development efforts will continue with Sponsor Users.
- Please let us know if you are interested in being a Sponsor User.

Thank you!

Questions or comments?

Trademarks

- IBM, the IBM logo, ibm.com and Rational are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Notes

- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- 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.