

# Name-Value Pair and ECB Owner Names Guidance

---

Josh Wisniewski  
z/TPF Development

# 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.

# Purpose of this Education Session

## Purpose of this Education Session

- Two years ago I gave the education session describing our vision of how NVPs, owner names, and etc. would be used by applications to implement our vision for NVPC:  
[http://www.ibm.com/software/hpf/tpf/tpfug/tgf16/TPFUG\\_2016\\_EDUC\\_ADI.pdf](http://www.ibm.com/software/hpf/tpf/tpfug/tgf16/TPFUG_2016_EDUC_ADI.pdf)
- Much has been implemented and the ideas have evolved.
- And so this session will:
  - Provide updated guidance for how to use all NVP APIs and Owner names APIs.
  - Provide guidance for your NVP conventions.
  - Walk through setting NVPs and owner names for various programming models in light of existing and upcoming APIs.
  - This material to become source for formalized z/TPF documentation.  
A sort of white paper on all aspects of using NVPs and Owner names.

# Benefits of NVPs and ECB Owner Names

## Benefits of NVPs

- NVPs can be used for name-value pair collection (NVPC) to capture system resource usage metrics at the application and message level.
- NVPC provides insights at a lower level than CDC, DC/DR, ZMOWN and etc. with more granularity at the application and message level.
- You contribute the names and values for a truly multi-dimensional analysis customized to your business.
- NVPC can be used to:
  - Save time diagnosing problems, planning application and configuration changes, and refactoring applications.
  - Provide in depth data to justify decision making and prioritization when refactoring applications.
  - Provide data driven business insights.
- NVPs can also be used to pass data (such as credentials) without changing the application linkage.

## NVPC Value Statements

- 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.
- A capacity planner can use name-value pair collection to determine the additional physical assets required for the expected message rate growth of a specific service given new resource usage metrics.
- An application architect can use name-value pair collection to identify inefficient code packages that can be refactored to improve system performance.

## Benefits of ECB Owner Names

- ECB owner names appear in NVPC as vertical NVPs. As such, you can use ECB owner names for delineating code packages or phases of processing for additional insights NVPC.
- ECB owner names can also be leveraged by ZMOWN collection and various Z commands such as ZDECB and ZSTAT to help understand the current state of the system.



# Guiding Principles

## Guiding principles in this presentation

- Some of this is future work and we'll clearly denote that in this presentation.
- In the analysis and interpreting the results of NVPC, ECB owner names are treated as Vertical NVPs. As such, at times I may reference NVPC results as NVPs and this includes ECB Owner names.

## Guiding principles for the z/TPF NVP Strategy

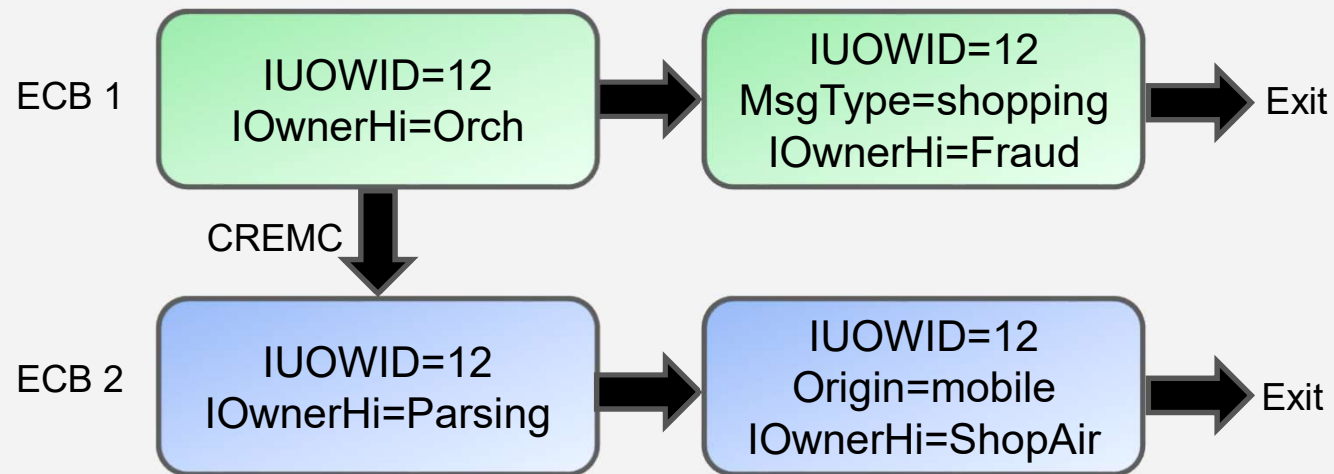
- Easier:
  - Minimize application changes whenever possible:
    - ECB owner names can be set by configuration.
    - Future: Where possible, we'll enable setting NVPs by configuration.
    - Future: z/TPF handles passing NVPs on your behalf whenever possible (such as ECB create, REST, Java, business events, etc.).
    - Set your horizontal NVPs as early as possible, preferably in the middleware so you can avoid making application changes.

## Guiding principles for the z/TPF NVP Strategy

- Easier:
  - Minimize application changes whenever possible:
    - Horizontal NVPs are treated as “set and forget, true for all ECBs for all time”. As such, TPFRTMC will inspect all ECB exit records that share an IUOWID to create the list of horizontal NVPs. These consolidated horizontal NVPs are then applied to the owner name change records. This behavior accommodates for when you can not set the NVPs as early as possible.

## Guiding principles for the z/TPF NVP Strategy

- ECB 1 sets MsgType=shopping while ECB 2 sets Origin=mobile, the horizontal NVPs for this UOW are MsgType=shopping and Origin=mobile.



## Guiding principles for the z/TPF NVP Strategy

- The following table illustrates the results of the TPFRTMC analysis (metrics are omitted).

Row Type	Horizontal NVPs	Vertical NVPs
Horizontal (exit)	MsgType=shopping Origin=mobile	none
Vertical (owner name change)	MsgType=shopping Origin=mobile	IOwnerHi=Orch
Vertical (owner name change)	MsgType=shopping Origin=mobile	IOwnerHi=Parsing
Vertical (owner name change)	MsgType=shopping Origin=mobile	IOwnerHi=Fraud
Vertical (owner name change)	MsgType=shopping Origin=mobile	IOwnerHi=ShopAir

## Guiding principles for the z/TPF NVP Strategy

- More Accurate:
  - How do we help you make your NVPC results more reliable for various application programming models and situations? We will provide various APIs to allow you to augment your applications to facilitate collection for more accurate results.
    - Future: Get and set list NVP APIs.
    - Future: Begin and end UOW NVP APIs.
    - Etc.
- More Value:
  - How do you make the most of your investment of setting NVPs?
    - Future: User defined NVPC metrics such as proprietary APIs, business metrics, or etc.
    - Future: NVPC in real time.
    - Future: Capture diagnostic information based upon specific NVPs in an ECB.

# Where do I begin?



## Where do I begin?

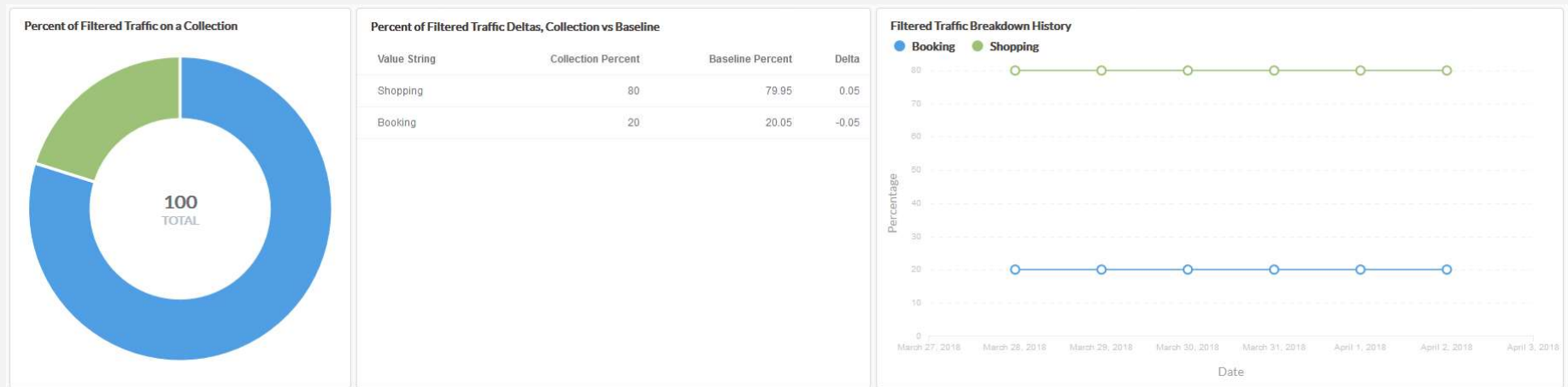
- Start small. Start with a single horizontal NVP such as MsgType and set the ECB Owner Hi by configuration.
- These two dimensions alone have tremendous value as the following slides will show.
- Try NVPC with these two dimensions. Digest the results.
- Begin to ask what else would be helpful.
- When you've got a good understanding, then move on to setting your NVP conventions.
- Don't be afraid to go back and refactor your NVPs afterwards. Be Agile! Try, fail fast and small, learn and grow, repeat.

## The Value of Using NVPs

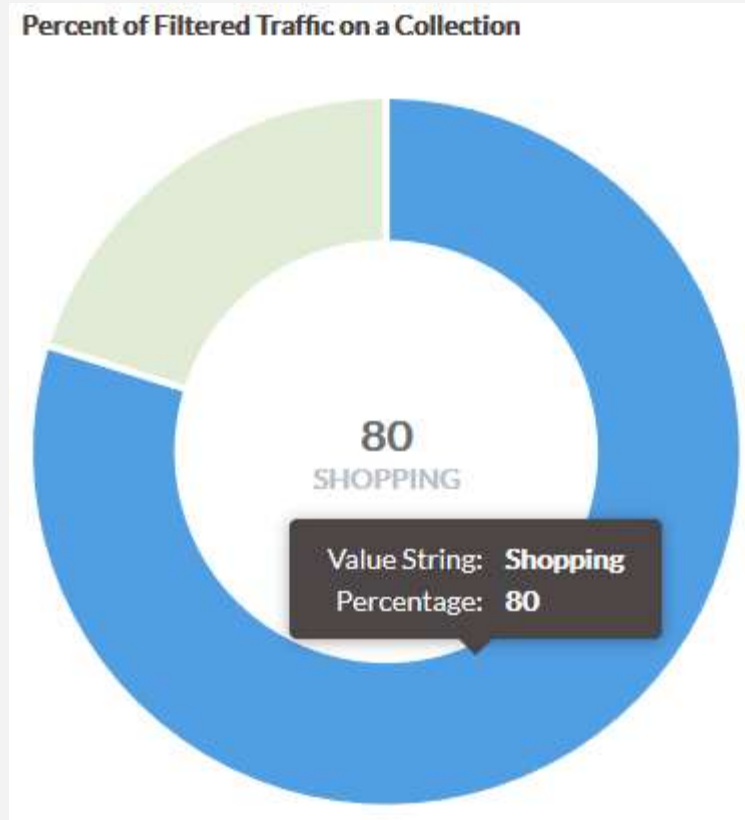
- With a minimal investment, you can realize a tremendous amount of value from NVPs.
- In the following charts we will:
  - demonstrate the value of setting a single NVP.
  - demonstrate the value of setting a single level of ECB Owner name.
  - demonstrate the additional value of setting a single NVP and a single level of ECB Owner name together.
- The following charts show the open source package Metabase as our example implementing the wire frames we showed last year. However, you can use your preferred SQL package.

# The Value of Using NVPs

- Suppose we modify our application base to have one name-value pair “MsgType” that describes the type of message or utility that is executing.
- It can have the possible values: Shopping, Booking, Ticketing, NightlyFileMaintenance, etc..
- NVPC will be able to show you:



# Average message mix for a collection.



Average message mix for a collection compared to baseline.

**Percent of Filtered Traffic Deltas, Collection vs Baseline**

Value String	Collection Percent	Baseline Percent	Delta
Shopping	80	79.95	0.05
Booking	20	20.05	-0.05

# Average message mix historical Trends.

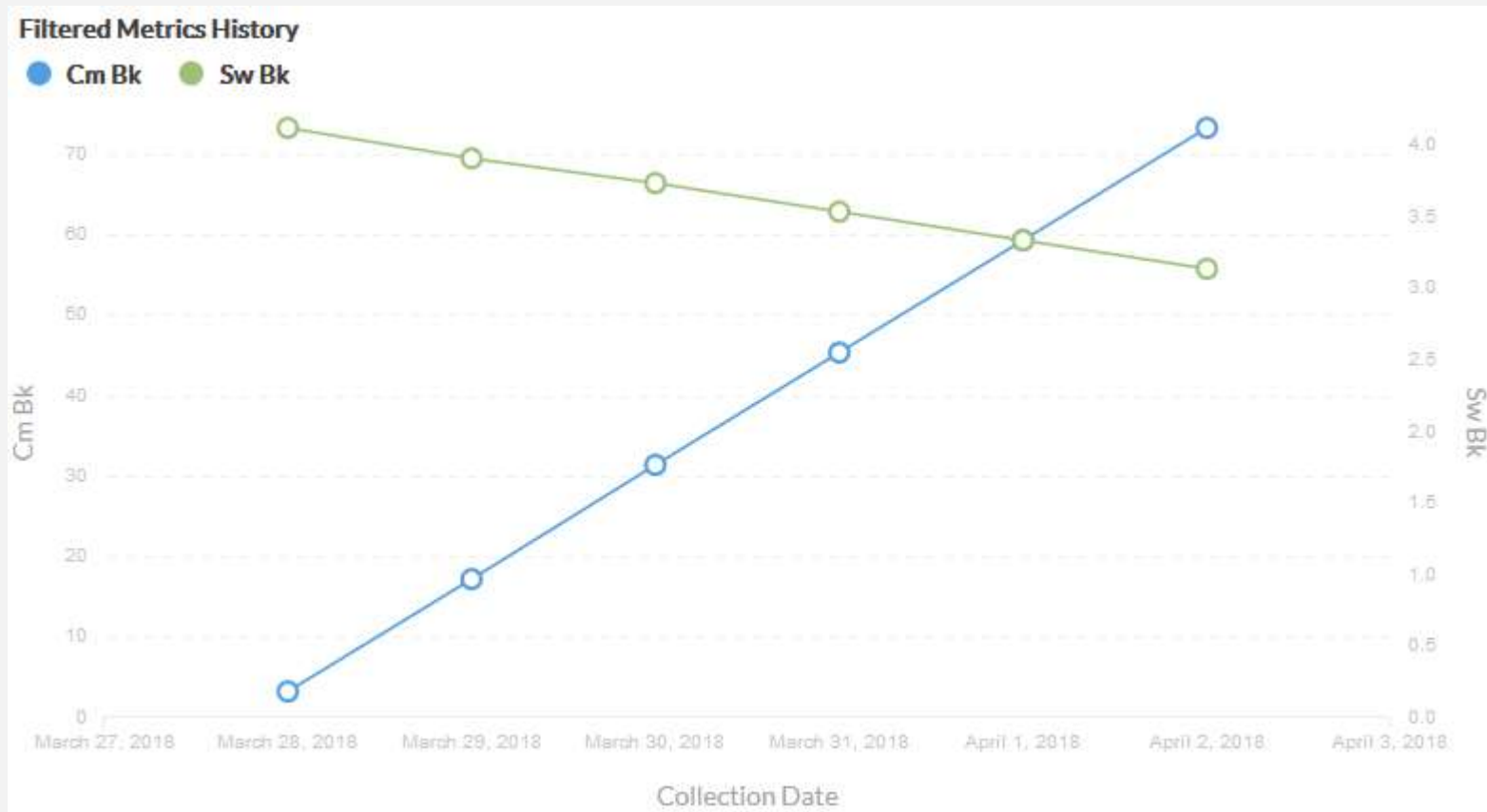


Average resource usage per message compared to baseline.

Filtered Metrics Deltas			
Metric	Current Value	Baseline Value	Delta
cmbk	73.3	31.32	41.98
swbk	3.14	3.72	-0.58

Metrics are the same as ZMOWN and ECB Resource monitor such as: Existence Time, 1MB Frames, COW, IOs, z/TPFDF, and etc..

# Average resource usage per message history.





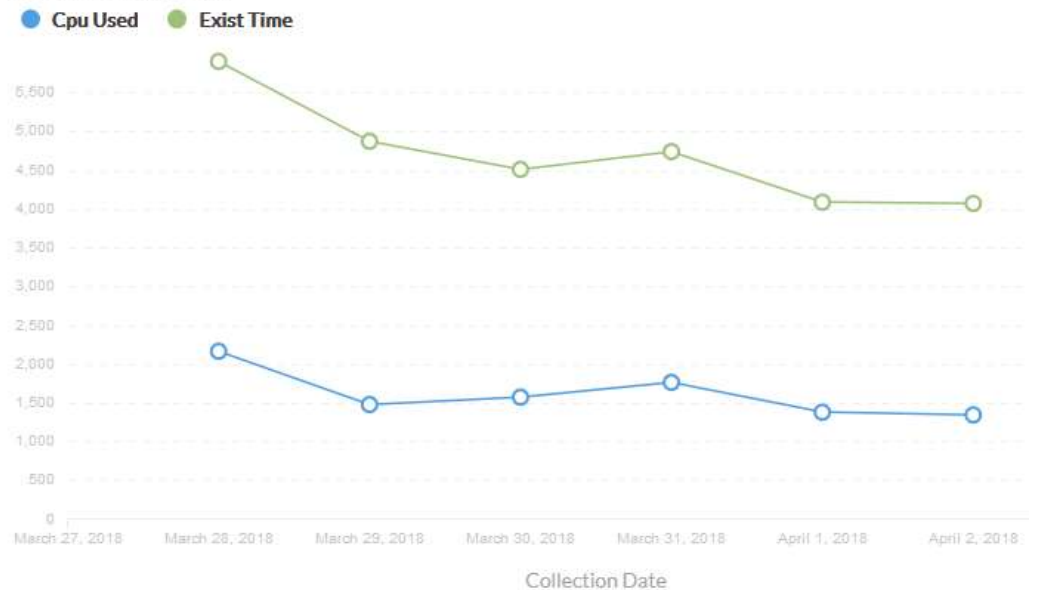
# Average resource usage for a single type of message compared to baseline with history.

Horizontal NVP Filter  
msgtype=booking ×

## Filtered Metrics Deltas

Metric	Current Value	Baseline Value	Delta
cpu_used	1,341.97	1,678.89	-336.92
exist_time	4,070.51	4,824.19	-753.68

## Filtered Metrics History



## The Value of Using NVPs

**All of those data points from  
just one NVP!**

## The Value of Using NVPs

- Suppose we set ECB Owner Name Hi for the code package by configuration (NO code change). It can have the possible values: COMMS, PARSE, FRAUD, SHOP, BOOK, FINALIZE, etc..
- Just like the MsgType NVP, NVPC will be able to show you:
  - Average percent of messages that use this code package compared to baseline. And the historical trends.
  - Average resource usage by a particular code package. And the historical trends.

## The Value of Using NVPs

- Lastly, you can use the NVP MsgType and ECB Owner Name Hi for the code package together.
- Just like the MsgType NVP, NVPC will be able to show you:
  - Average percent of MsgType messages that use a particular code package. And the historical trends.
  - Average percent of MsgType messages that use each code package ever called by that message type. And the historical trends.
  - Average resource usage by a particular code package by a specified MsgType. And the historical trends.

## The Value of Using NVPs

**All of those data points from just one NVP and ECB Owner names set by configuration!**

## The Value of Using NVPs

- Start small, with a single NVP and ECB Owner Name Hi (set by configuration) and explore the value of the data.
- Experiment with the creating the views and analysis you desire for those two dimensions. Go beyond what we've described.
- For example, for the specified breakdown (MsgType or OwnerHi) show the different NVP values and their existence time in descending order with history.

## The Value of Using NVPs

- And then build up. Add in additional horizontal NVPs (values do not change through the life of the processing), ECB Owner Mid, and vertical NVPs such as ECBPurpose or other NVPs that change through the life of processing at collection boundaries (ECB Exit and Owner Name Change).
- Experiment with other NVPs such as horizontal NVPs that categorize message input parameters. For example the name SearchRadius could have the values: Small, Medium, Large, and Extreme.
- When you have a better feel for how you want to use NVPC, standardize your practices across your code base.
- Consider setting NVPs in your middle ware packages such that you do not have change your applications down stream.

## The Value of Using NVPs

- You may be surprised by what you learn about your applications. We were!



What do I do  
next?

## What do I do next?

- You've started small. You've experimented with NVPC. And now you're ready to dig in a bit deeper.
- What do you do next? Ask/answer a variety of questions to help you define your NVPs names and values. Such as:
  - What kinds of problems do you want to solve with NVPs? i.e.
    - Growing file chains by message type.
    - Issues caused by input parameters by customer.
    - Resource usage and frequency of mobile traffic.
  - What kind of problems do you see on your system regularly? What NVPs would help you solve them?
  - What kind of problems do you see on your system that are really hard to figure out? What NVPs would help?

## What do I do next?

- What do you want to see in a report or query? That is, the results of analyzing the results in the database produced by TPFRTMC.
- This can be a very beneficial approach instead of adding every NVP you can think of that may ultimately not be useful.
- For Example:
  - Display values for a specified name (i.e. MsgType, IOwnerHi code package, etc.) in a table in decreasing order by existence time with deltas from the baseline.

MsgType	Exist_time	Baseline	Delta
Booking	20,000	20,000	0
Shopping	15,000	14,000	7%
Ticketing	10,000	10,500	-5%

## What do I do next?

- NVPC is designed to show average usage for a given system resource for a NVP.
- It is built for sampling to minimize impacts to a production system.
- It is not intended for investigating a single message. (You potentially could, but your message and owner skip factors will probably need to be zero! This is more of a limited function/unit test usage, not production system usage. Be careful that you do not draw incorrect conclusions if you look at sampled production data.)
- Lastly, NVPs should reduce to a finite set of values (Booking, Shopping, etc.). We advise against storing ECB SVM address, metrics and etc. in NVPs.

# ECB Owner Names

## What are ECB Owner Names? APIs? Etc?

- This was covered in my 2016 education session (pages 16-25):  
[http://www.ibm.com/software/hwp/tpf/tpfug/tgf16/TPFUG\\_2016\\_EDUC\\_ADI.pdf](http://www.ibm.com/software/hwp/tpf/tpfug/tgf16/TPFUG_2016_EDUC_ADI.pdf)

## ECB Owner Names Guidance

- Set ECB owner names automatically by configuration. See PJ44680 – Program Configuration File – [https://www.ibm.com/support/knowledgecenter/en/SSB23S\\_1.1.0.15/gtpl2/progcfg.html](https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.15/gtpl2/progcfg.html).
- Use ECB owner names to represent code packages or phases of processing. Think of ECB owner names as vertical NVPs. That is, piece parts that make up the whole.
- Do not change ECB owner names too frequently i.e. at entry to every shared object or function. Set ECB owner names at the primary entry points to a code package or phase of processing. If an owner name is set to the value it already is, it is a no-op.
- For more precise overall results without code changes, turn on automatic ECB owner name restore – PJ44470 – [https://www.ibm.com/support/knowledgecenter/en/SSB23S\\_1.1.0.15/gtpa2/ceownrc.html](https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.15/gtpa2/ceownrc.html).

# Name-Value Pairs



## What are Name-Value Pairs?

- This was covered in my 2016 education session (pages 33-37):  
[http://www.ibm.com/software/hwp/tpf/tpfug/tgf16/TPFUG\\_2016\\_EDUC\\_ADI.pdf](http://www.ibm.com/software/hwp/tpf/tpfug/tgf16/TPFUG_2016_EDUC_ADI.pdf)
- NVPs are in z/TPF Dumps. This may be particularly useful as you will be able to see the various NVPs as annotations as to the purpose, source of the work or etc. depending upon your NVP conventions.

# Name-Value Pairs and ECB Owner Name Guidance

## Name-Value Pairs Guidance

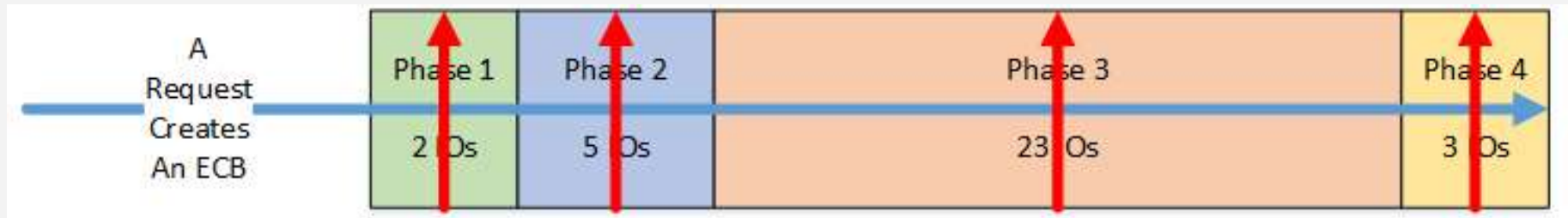
- This section will provide guidance for setting and using NVPs for the best NVPC results.

## Name-Value Pairs Guidance – Policy vs Conventions

- NVPC Policy – Tells NVPC which NVPs to collect and whether they are horizontal vs vertical.  
[https://www.ibm.com/support/knowledgecenter/en/SSB23S\\_1.1.0.14/gtps3/s3nvpcpf.html](https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.14/gtps3/s3nvpcpf.html)
- NVP Conventions – These are decisions you make and enforce in your application base. They are subjective and at your discretion.

## Name-Value Pairs Guidance – Horizontal vs Vertical

- Horizontal NVPs
  - Are represented as the blue line going left to right.



## Name-Value Pairs Guidance – Horizontal vs Vertical

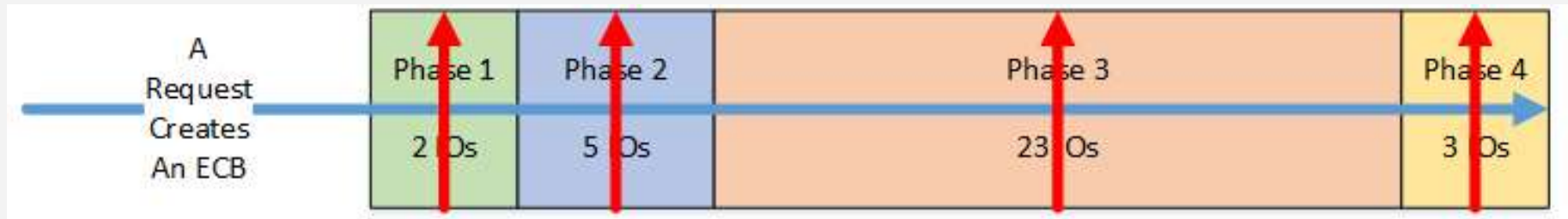
- Horizontal NVPs
  - Set and forget. They do not change. Always true.
  - They are valid for the parent, children, grandchildren, “foster children” on other complexes, and etc. ECBs that share the IUOWID.
  - They represent facts about the whole processing.
  - TPFRTMC will inspect all ECB exit records that share an IUOWID to create the list of horizontal NVPs. These consolidated horizontal NVPs are then applied to the owner name change records. This behavior accommodates for when you can not set the NVPs as early as possible.
    - If your ECBs have conflicting horizontal NVPs, you will lose the values you set. They will contain the value IConflictingValue.

## Name-Value Pairs Guidance – Horizontal vs Vertical

- Horizontal NVPs – Examples
  - MsgType - BalanceQuery, Authorization, Booking, Shopping, etc..
  - MsgOrigin - Mobile, Web, Terminal
  - MsgQualifier - Domestic, International, Internal
  - Merchant - HotelChainM, AirLineA, GroceryA&Z
  - Geography - NorthAmerica, Europe, AsiaPacific
  - SourceType - Merchant, Bank, BusinessPartner, InHouse
  - SourceProtocol - REST, SOA, EFTPOS, ISO8583, InHouse
  - InputParameters - S,M,L,XL

## Name-Value Pairs Guidance – Horizontal vs Vertical

- Vertical NVPs
  - Are represented as the red lines going bottom to top.





## Name-Value Pairs Guidance – Horizontal vs Vertical

- Vertical NVPs
  - Change.
  - If you might change a value during the processing of a message, then it should be a vertical.
  - If it might have a different value for a different ECB with the same IUOWID, then it should be a vertical NVP.
  - They represent the piece parts that make up the whole.
  - NVPs are collected at:
    - Owner change: Use ECB owner names for phases of processing or code package.
    - Exit: You could use the set API to have an ECB's Purpose.
    - Future – Begin/End UOW APIs: You can use these collection points to for verticals NVPs such as ServiceType.

## Name-Value Pairs Guidance – NVP Conventions

- Standardize your ECB Owner names, NVP names and NVP values.
  - Develop and implement a unified strategy across all of your applications for using the owner names and name-value pairs.
  - Standardize both names and values.
  - You want your NVP results to reduce. 10,000 sampled shopping messages should result in a handful of varieties in the NVPC results. Remember, NVPC results are intended to show you what's going on, on average. Sampling dictates this needs to be the case.
  - ECB Owner Names, NVP Names and NVP Values must be consistent in order to reduce. Be careful of mixed case, different abbreviations and etc. Use consistent strings.

## Name-Value Pairs Guidance – NVP Conventions

- Standardize your ECB Owner names, NVP names and NVP values.
  - **Don't use values that are not reducible.**
    - Timestamps
    - ECB address
    - File address
    - PNR number
    - Unique ids (set to IUOWID)
    - Passwords, user names (set these as NDSP so they can not be collected)
    - Metrics (such as existence time)
      - Understand the metrics IBM provides.
      - Future - We plan to add support for user defined metrics.
      - You can use numeric or even binary values. But ensure these are characteristics of the processing that will reduce. Or consider defining groups of values: S, M, L, XL.

## Name-Value Pairs Guidance – NVP Conventions

- Standardize your ECB Owner names, NVP names and NVP values.
  - Use NVP names and values that are not too small but not too large. Make them readable, not cryptic. But don't use sentences either. All name-value pairs fit in a single 4k block.
  - Consider how an owner name or NVP might be broken down by another. Owner hi broken down by owner mid. A good vertical example is ECBPurpose broken down by phase of processing broken down by code package. A good horizontal example is MsgType broken down by Origin.
  - Consider overwriting the IUOWID if you have an enterprise wide unique token.
  - Experiment with other NVPs such as horizontal NVPs that categorize message input parameters. But make sure your results will reduce (S, M, L, XL).

## Name-Value Pairs Guidance – NVP Conventions

- Define when and where NVPs and ECB owner names will be set.
  - Horizontal NVPs
    - Set as early as you can, before ECB create macros are issued. Set and forget, do not change.
  - ECB Owner Names
    - Use for phases of processing or code package.
    - Set at main interface points. Do not change owner names too frequently.
  - Vertical NVPs
    - Set prior collection point (exit, owner name change, Future – Begin/End UOW APIs) occurs (or as early as possible).
- Define how NVPs will be managed for various conditions:
  - Sending NVPs across communications protocols to remote systems.
  - Saving NVPs to storage to be set across AOR types situations.

## Name-Value Pairs Guidance – Running NVPC

- See 2018 “Value of NVPs” TPFUG presentation from the “Operations and Coverage Subcommittee” for general guidance on running NVPC.
- As a general guideline, ensure that you have around 1000 samples for each of your most common messages to have a reasonably tight 90% confidence interval in the majority of your results.
- Suppose bookings are 1% of our message mix and we want to ensure we have 1000 booking samples. This means we want to have about 100,000 samples total. Given a rate of 2000 messages per second with the message skip factor set to 20, we would need to run collection for about 17 minutes.

# Programming Models

## Programming Models

- The following section will walk through various programming models and will demonstrate how the various NVP APIs will be used together to generate accurate results for NVPC.
- The following models are considered:
  - Single Request – Single ECB – Introduces set/get APIs and owner change
  - Multiple Requests – Single ECB (Read Loop) – Introduces begin/end APIs.
  - Single Request – ECB and children – shows NVPs passed across creation
  - Single Request – AOR and Remote – Introduces set/get list APIs.
  - Single Request – Single ECB – Orchestration – begin/end APIs
  - Single Request – Multiple ECB – Orchestration
  - Single Request – REST
  - Single Request – Java
  - Single Request – Business Events



## Programming Model - ECB Owner Names

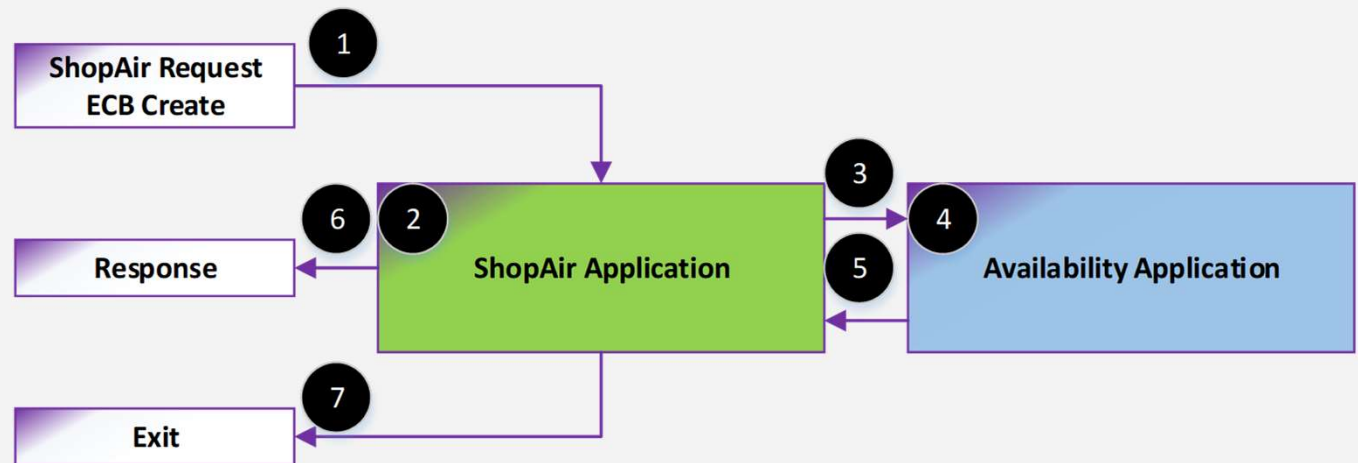
For most accurate results for ALL programming models with no code changes:

- Automatically set owner name using the Program Configuration File.
- Use automatic ECB Owner name restore.

# Model: Single Request – Single ECB

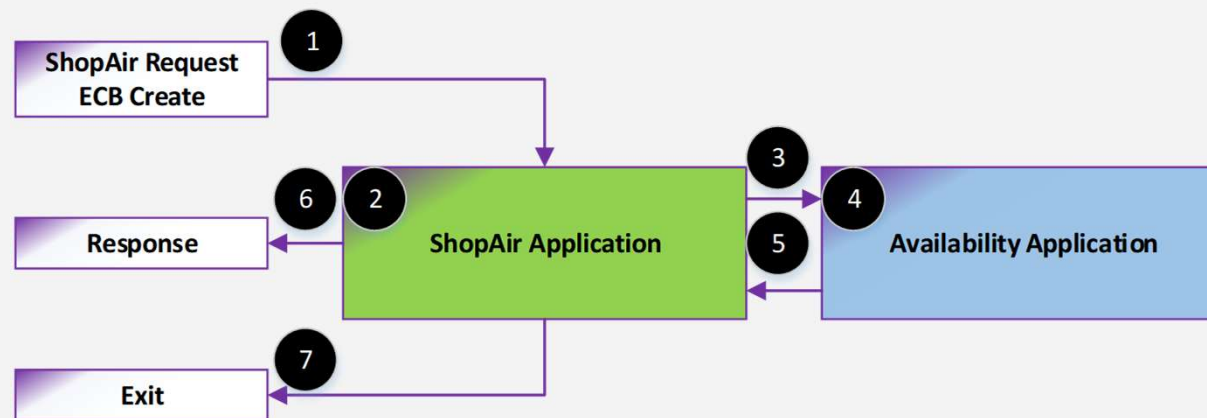
# What is this programming model? Single Request – Single ECB

- **What:** A single ECB processes a single request.
- **Problem:** How and when do you set NVPs? How do you retrieve NVP values? How and when do you set ECB Owner Names?



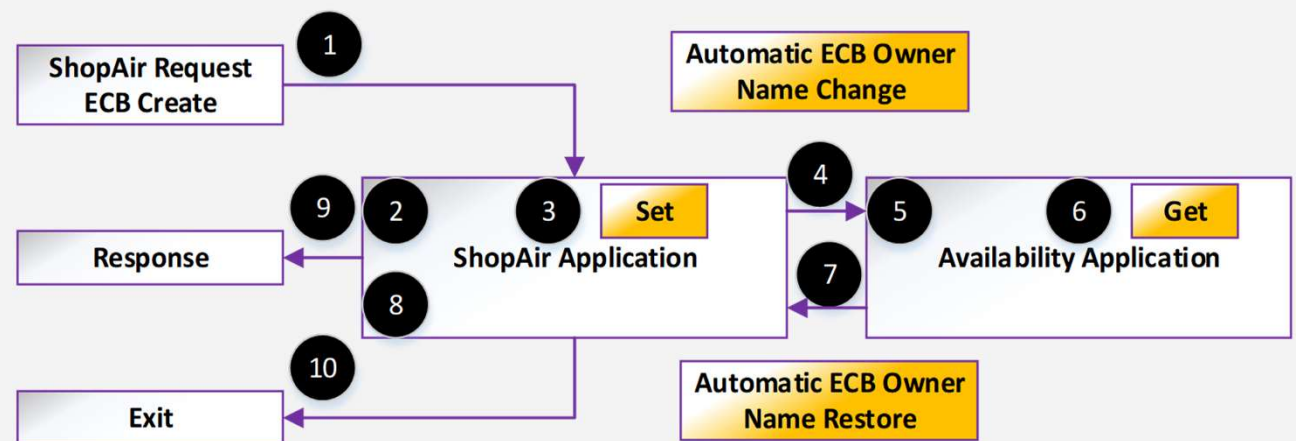
# What is this programming model? Single Request – Single ECB

- **Desired Results:**
  - Shopping UOW - 6 IOs
    - 4 IOs - ShopAir Package
    - 2 IOs - Avail Package



# What is this programming model? Single Request – Single ECB

- Solution:** Use the NVP set API after parsing the message. Use the NVP get API. Leverage the program configuration file to automatically ECB Owner Names. Leverage ECB owner name restore.



## Name-Value Pairs APIs: Get, Set, and Remove

- Purpose: These APIs provide a means for you to store a NVP, retrieve a value for a specified name, or remove a NVP.
- `tpf_nameValueLocalSet`
  - Specify a unique string for the name. Like a key for a hash map. These names are case sensitive.
  - All IBM created names will be with the letter “I”.
  - Specify the value and it’s length.
    - This can be a binary value, so it could be a printable string or data. TPFRTMC stores binary values in the database in the string form of “0xFF00”.
    - From the perspective of NVPC and the resulting analysis, values are case sensitive.

## Name-Value Pairs APIs: Get, Set, and Remove

- `tpf_nameValueLocalSet`
  - Specify a characteristic such as non-displayable.
    - Use this non-displayable characteristic to hide values in dumps, Z commands, the debugger and etc. For example, a NVP that may warrant being non-displayable is “Password=ABCD1234”.
    - NVPs that are marked as non-displayable will not be captured by NVPC.

## Name-Value Pairs APIs: Get, Set, and Remove

- `tpf_nameValueLocalSet` - IUOWID
  - At the time an ECB is created (having no parent ECB), a unique id is created. This id is saved in the NVP IUOWID (Unit of Work ID).
  - The IUOWID consists of the processor ID, complex name, and timestamp.
  - Like all other NVPs, the IUOWID is passed on create to any child ECB.
  - The idea of the Unit of Work terminology is that the Parent, Children, Grandchildren and etc. ECBs typically constitute a set of processes achieving a single task and as such share the single IUOWID so that their resource usage can be coalesced in the results of the TPFRTMC analysis. i.e. booking uses a total of 140 IOs, but was accomplished with 5 ECBs.
  - You can use the set API with the name IUOWID to a 64 byte or less value (ensure your value is unique across the complex).
    - You may want to do this if a child ECB is spawned to accomplish an unrelated task.
    - Or maybe you have a unique transaction id that comes in with the request.



## Name-Value Pairs APIs: Get, Set, and Remove

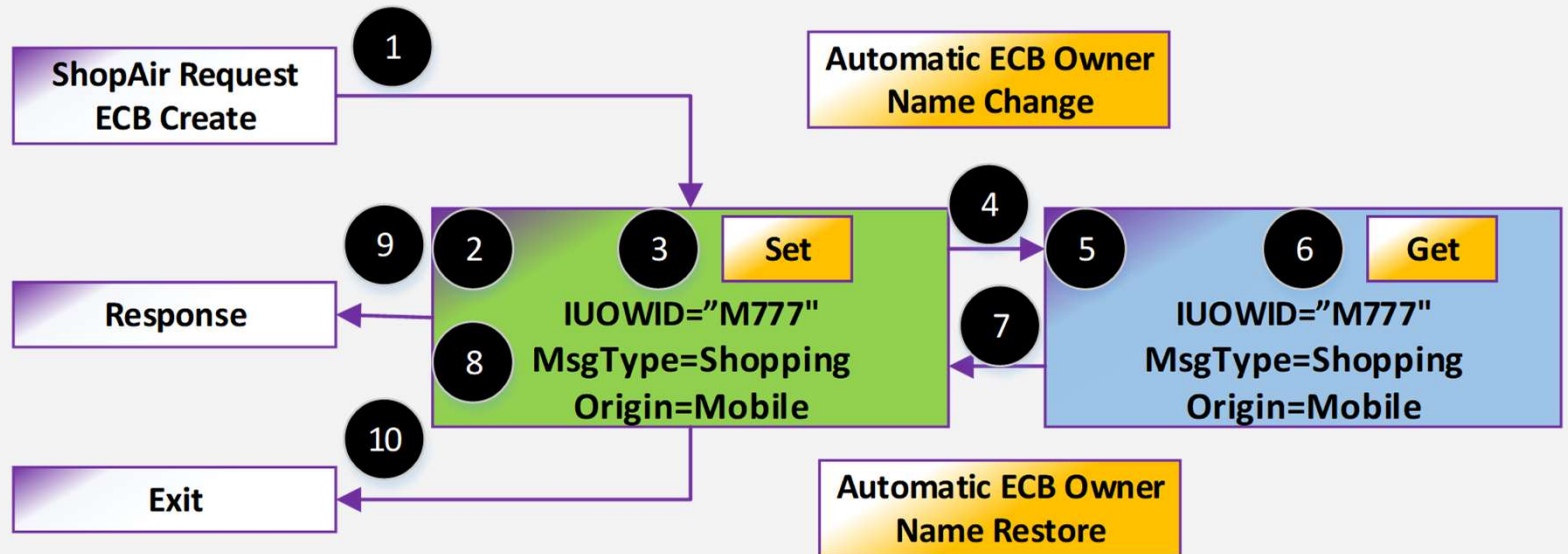
- `tpf_nameValueLocalSet` – Horizontal or Vertical?
  - You do not identify if a NVP is horizontal or vertical at set time. This is specified in the NVPC policy file when you run NVPC.

## Name-Value Pairs APIs: Get, Set, and Remove

- `tpf_nameValueLocalGet`
  - Specify the name to retrieve the corresponding value.
- `tpf_nameValueLocalRemove`
  - Specify the name to remove the NVP.
  - Alternatively, you can specify `TPF_NAMEVALUE_REMOVE_ALL` to remove all NVPs.
  - The IUOWID NVP can not be removed.

# Name-value Pairs – Single Request – Single ECB

- **Solution:** Here is an example of the desired NVPs and the UOW Id.



# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

- Orange is logic and resource for Shopping UOW.
- Green is the ShopAir package. Blue is Avail Package.

# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalG Return to ShopAir pa	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**ECB Created**  
 IUOWID 23 Generated by  
 system and set

# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalG Return to ShopAir par	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile



# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile



# Name-value pairs API walkthrough - Single Request – Single ECB

- Optional, set the IUOWID.

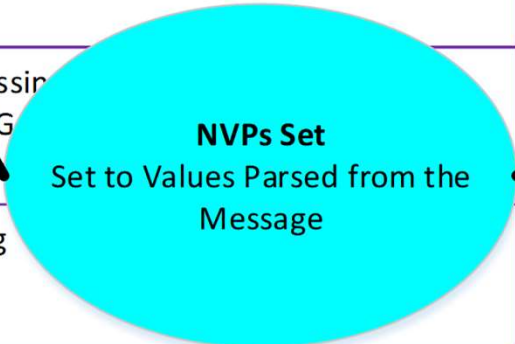
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**IUOWID – M777**  
Set to Transaction Id Parsed  
From the Message



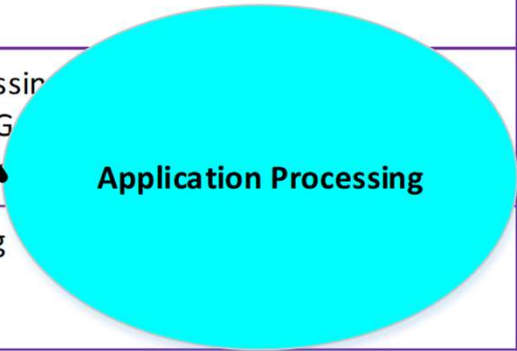
# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalSet(Avail) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile



# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalG... Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile



# Name-value pairs API walkthrough - Single Request – Single ECB

- No application code change.

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalG... Return to ShopAir pa...	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile



# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**ECB Owner Name Change**  
 NVPC collection occurs  
 Vertical (Owner Change) –  
 3 IOs  
 IUOWID=M777  
 IOwnerHi=ShopAir

# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(Origin) tpf_nameValueLocalSet(ShopAir) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**Application Processing**

# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(Origin) tpf_nameValueLocalSet(ShopAir) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**Get NVP API**  
 Use the get API to retrieve the value for a specified name.



# Name-value pairs API walkthrough - Single Request – Single ECB

- No application code change.

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(Origin) tpf_nameValueLocalSet(ShopAir) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**Automatic ECB Owner Name Restore**

# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(I tpf_nameValueLocalSet(S Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**ECB Owner Name Change**  
 NVPC collection occurs  
 Vertical (Owner Change) –  
 2 IO  
 IUOWID=M777  
 IOwnerHi=Avail

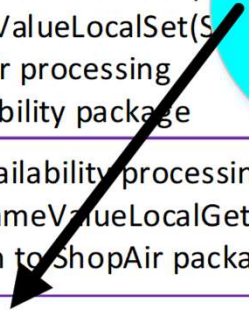




# Name-value pairs API walkthrough - Single Request – Single ECB

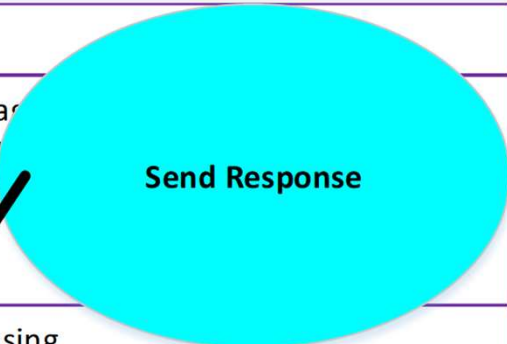
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(Origin) tpf_nameValueLocalSet(ShopAir) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**Application Processing**



# Name-value pairs API walkthrough - Single Request – Single ECB

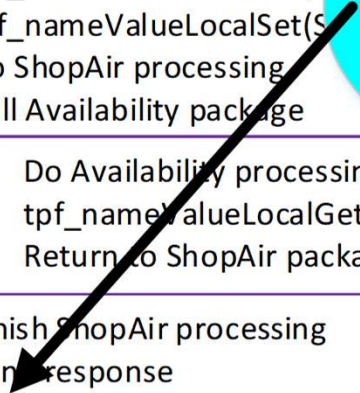
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(I... tpf_nameValueLocalSet(S... Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile



# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID=M777) tpf_nameValueLocalSet(S) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**Exit**  
 NVPC collection occurs  
 Horizontal (Exit) - 6 IO  
 IUOWID=M777  
 MsgType=Shopping  
 Origin=Mobile



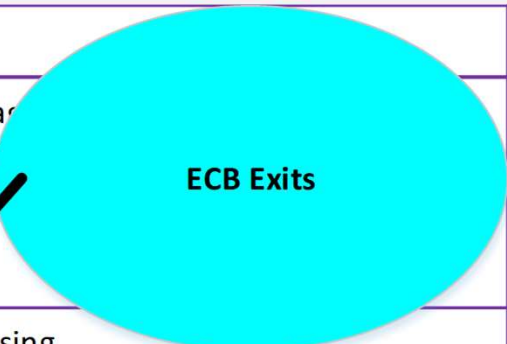
# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(Origin) tpf_nameValueLocalSet(ShopAir) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile

**Exit**  
 NVPC collection occurs  
 Vertical (Owner Change) –  
 1 IO  
 IUOWID=M777  
 IOwnerHi=ShopAir

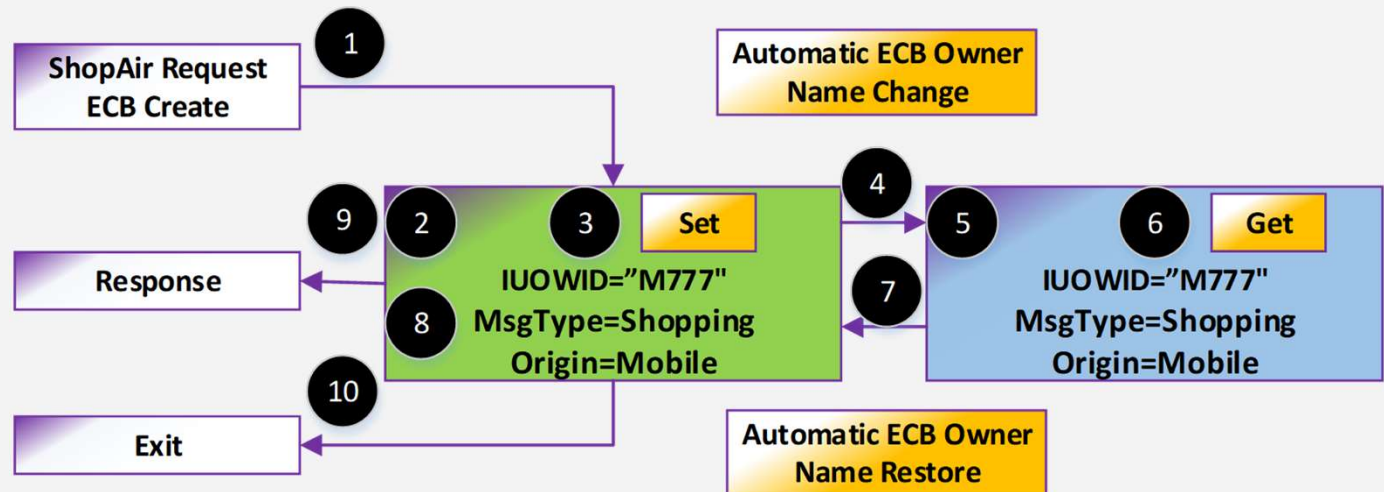
# Name-value pairs API walkthrough - Single Request – Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(Origin) tpf_nameValueLocalSet(Site) Do ShopAir processing Call Availability package	ShopAir	3	23 M777	MsgType=Shopping Origin=Mobile
Do Availability processing tpf_nameValueLocalGet(Origin) Return to ShopAir package	Avail	2	M777	MsgType=Shopping Origin=Mobile
Finish ShopAir processing Send response exit	ShopAir	1	M777	MsgType=Shopping Origin=Mobile



# Name-value pairs API walkthrough - Single Request – Single ECB

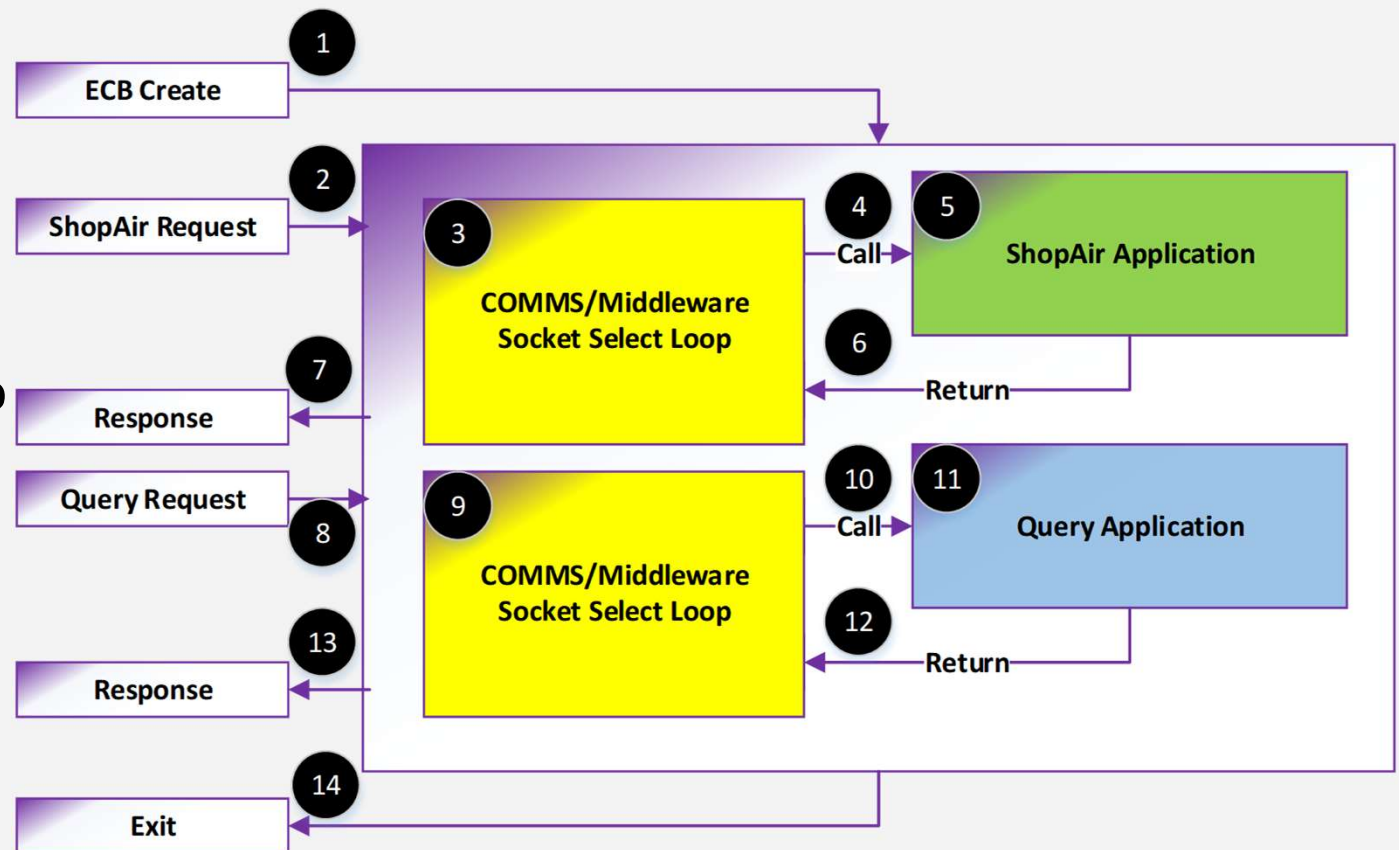
Row Type	Horizontal NVPs	Vertical NVPs	IOs
Horizontal (exit)	MsgType=shopping Origin=mobile	none	6
Vertical (owner name change)	MsgType=shopping Origin=mobile	IOwnerHi=ShopAir	4
Vertical (owner name change)	MsgType=shopping Origin=mobile	IOwnerHi=Avail	2



# Model: Multiple Requests – Single ECB (Read Loop)

# What is this programming model? Multiple Requests – Single ECB

- **What:** A single ECB processes multiple requests.
- **Problem:** All resources used to process the various requests are attributed to a single UOW at ECB exit.





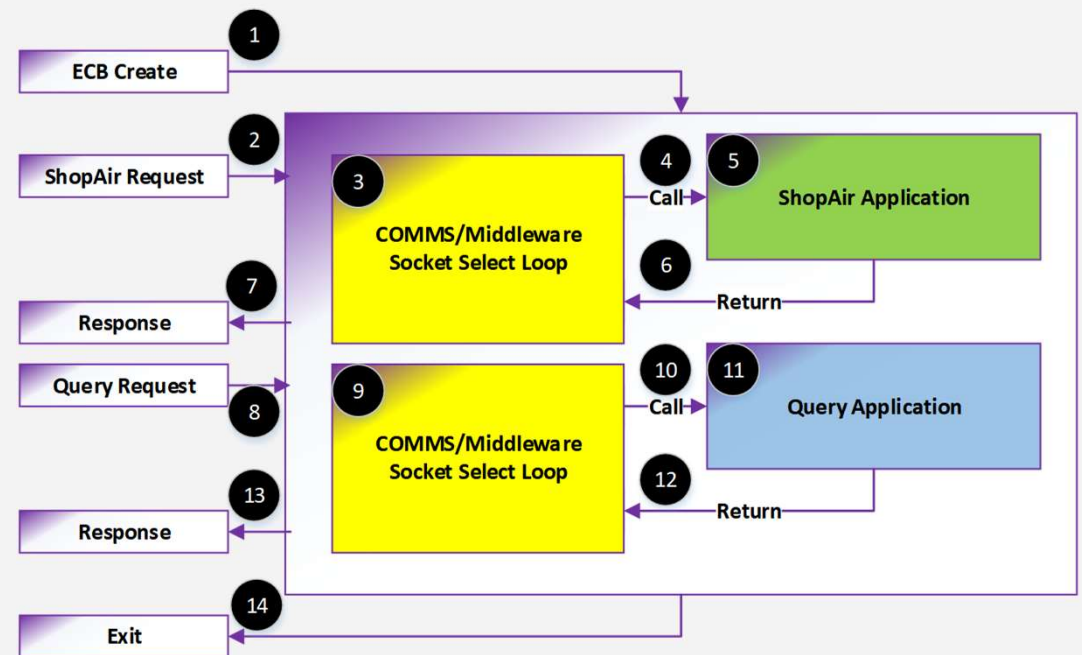
# What is this programming model? Multiple Requests – Single ECB

- **As-Is Results:**

- Query UOW - 14 IOs.
  - 6 IOs - COMMS Package
  - 4 IOs - Query Package
  - 4 IOs - ShopAir package
- Why is Query UOW calling ShopAir? It's not!

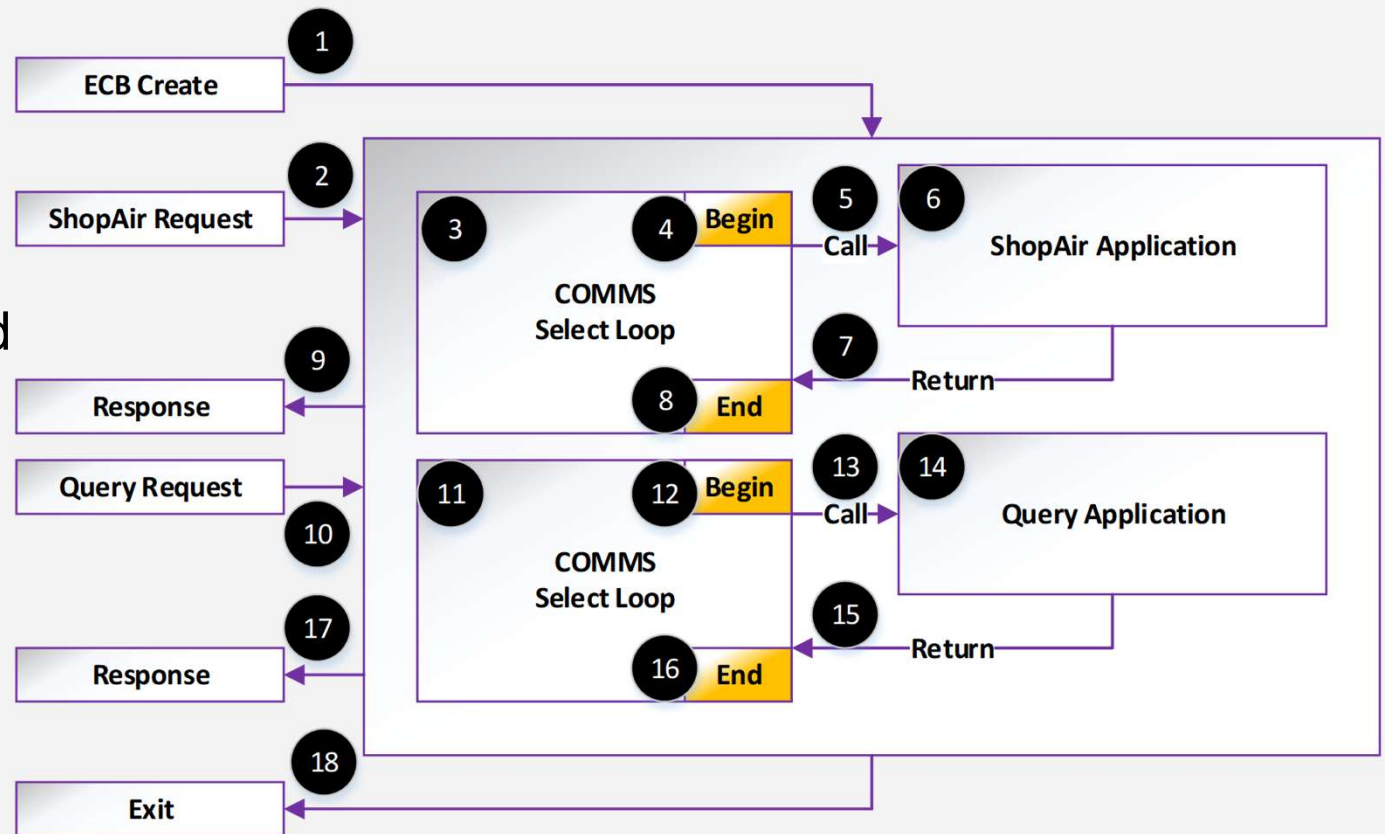
- **Desired Results:**

- COMMS UOW - 2 IOs
  - 2 IOs - COMMS Package
- Shopping UOW - 6 IOs
  - 2 IOs - COMMS Package
  - 4 IOs - ShopAir Package
- Query UOW - 6 IOs
  - 2 IOs - COMMS Package
  - 4 IOs - Query Package



# What is this programming model? Multiple Requests – Single ECB

- **Solution:** Begin and end UOW NVP APIs allow you to delineate the start and end of processing a unique message.



## Name-Value Pairs APIs: Begin and End UOW - Future

- Purpose: These APIs are used to denote the start and end of a piece of work. From the view of the NVPC results, it will appear as if a new ECB used the resources between the issuing of the begin and end APIs. In this way, the NVPC results can be made more accurate for various application architectures.
- Second use case demonstrated later in this presentation in the Single ECB Orchestration Model.
- More details to come as designs are firmed up.

## Name-Value Pairs APIs: Begin and End UOW - Future

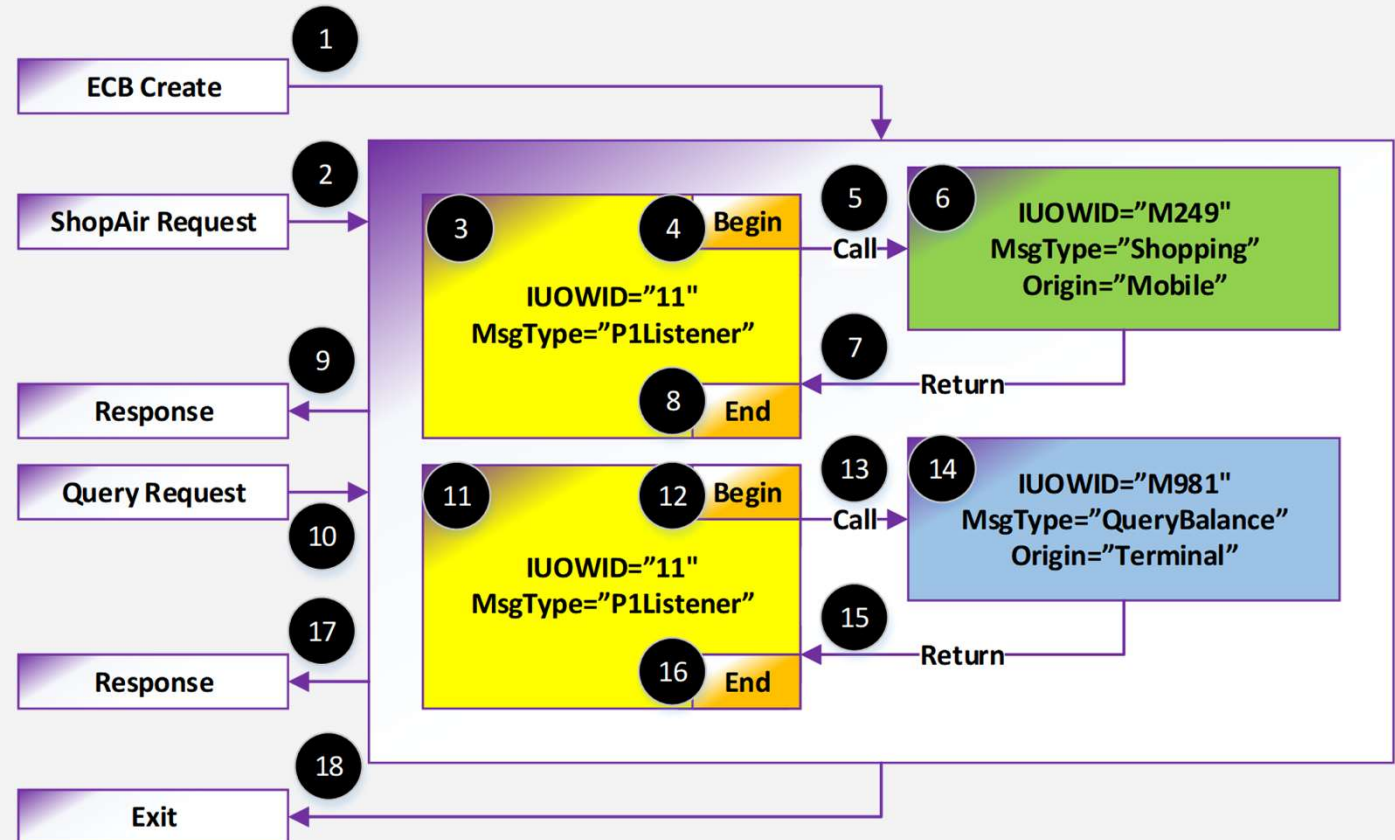
- tpf\_nameValueBeginUOW
  - Implicitly calls the end UOW API.
  - Push the NVPs into a NVP stack. They will be restored to their current values when the end UOW API is called.
  - NVPs can be managed in two ways:
    - **Clear NVPs** – This begin option indicates a new independent piece of work is starting. The NVPs are cleared and a new IUOWID is generated.
    - **Keep NVPs** – This begin option indicates an orchestration scenario is in play. Work is continuing. The NVPs remain unchanged including the IUOWID. You can think of this usage as setting an artificial boundary for Vertical NVPs such as ServiceType.
    - Each usage is demonstrated below in this presentation.

## Name-Value Pairs APIs: Begin and End UOW - Future

- `tpf_nameValueEndUOW`
  - Call an Accounting User Exit.
  - If active, perform NVPC or ZMOWN collection.
  - Reset ECB counters to 0 as if it's a new ECB.
  - Pop the NVPs from the NVP stack.
  - Kind of like a simulated ECB Exit event from the perspective of NVPC.

# Name-value Pairs – Multiple Requests – Single ECB

- Solution:** Here is an example of the desired NVPs and the UOW Id. Notice, the different IUOWIDs in the different colored blocks.



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

- Orange is logic and resource for Shopping UOW. White is COMMS UOW. Query UOW omitted but similar to the ShopAir request.
- Yellow COMMS package. Green is ShopAir Package

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**ECB Created**  
IUOWID 11 Generated by  
system and set



# Name-value pairs API walkthrough – Multiple Requests Single ECB

- No application code change.

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**ECB Owner Name**  
Set in linkage by  
configuration

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**NVP Begin**  
 NVPC collection occurs  
 Horizontal (Exit) - 1 IO  
 IUOWID=11  
 MsgType=P1Listener

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**NVP Begin**  
 NVPC collection occurs  
 Vertical (Owner Change) –  
 1 IO  
 IUOWID=11  
 IOwnerHi=COMMS

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**NVP Begin**  
Zero IO Counter as if it's a new ECB

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW( <b>TPF_NVP_BEGIN_CLEAR_NVPS</b> ) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	<b>(NVPs Cleared)</b> MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**NVP Begin**  
Clear all NVPs as if it's a new ECB.

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener



# Name-value pairs API walkthrough – Multiple Requests Single ECB

- Begin and end UOW APIs have no effect on ECB owner name.

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**ECB Owner Name is Unchanged**

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**IUOWID – M249**  
Set to Transaction Id Parsed  
From the Message

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

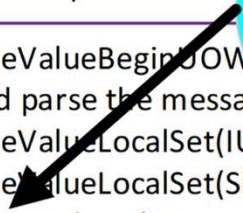
**NVPs Set**  
Set to Values Parsed From  
the Message

# Name-value pairs API walkthrough – Multiple Requests Single ECB

- No application code change.

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW() Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**ECB Owner Name  
Changed by Linkage by  
Configuration**



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW() Read and parse the message tpf_nameValueLocalSet(IUOWID, IOwnerHi=COMMS) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**ECB Owner Name Change**  
 NVPC collection occurs  
 Vertical (Owner Change) –  
 1 IO  
 IUOWID=M249  
 IOwnerHi=COMMS



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW() Read and parse the message tpf_nameValueLocalSet(IUOWID, ...) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**Application Processing**  
 Since begin/end and NVP setting is in the middleware, minimal changes are required to the application.

# Name-value pairs API walkthrough – Multiple Requests Single ECB

- No application code change.

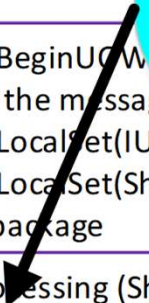
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**Automatic ECB Owner Name Restore**

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW() Read and parse the message tpf_nameValueLocalSet(IUOWID, IOwnerHi=ShopAir) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

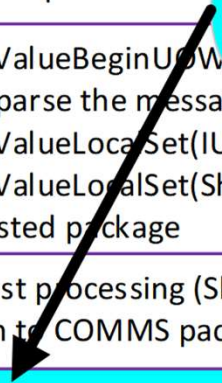
**ECB Owner Name Change**  
 NVPC collection occurs  
 Vertical (Owner Change) –  
 4 IO  
 IUOWID=M249  
 IOwnerHi=ShopAir



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**NVP Begin**  
Call Accounting User Exit



# Name-value pairs API walkthrough – Multiple Requests Single ECB

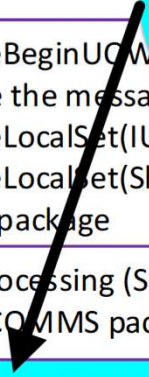
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
<b>tpf_nameValueEndUOW()</b>	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**NVP End**  
 NVPC collection occurs  
 Horizontal (Exit) - 6 IO  
 IUOWID=M249  
 MsgType=Shopping  
 Origin=Mobile

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
<b>tpf_nameValueEndUOW()</b>	<b>COMMS</b>	<b>1</b>	<b>M249</b>	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

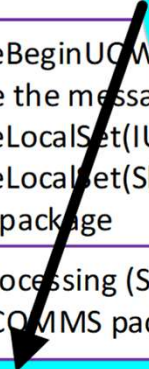
**NVP End**  
 NVPC collection occurs  
 Vertical (Owner Change) –  
 1 IO  
 IUOWID=M249  
 IOwnerHi=COMMS



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW() Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
<b>tpf_nameValueEndUOW()</b>	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

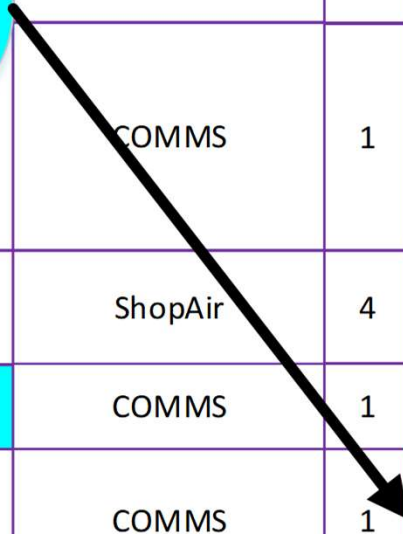
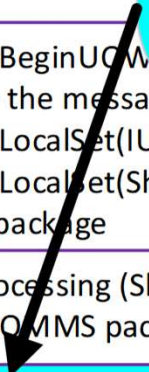
**NVP End**  
Zero IO Counter as if it's a new ECB



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
<b>tpf_nameValueEndUOW()</b>	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

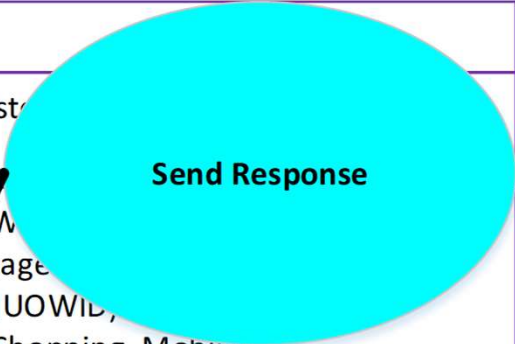
**NVP End**  
Pop NVPs from NVP Stack





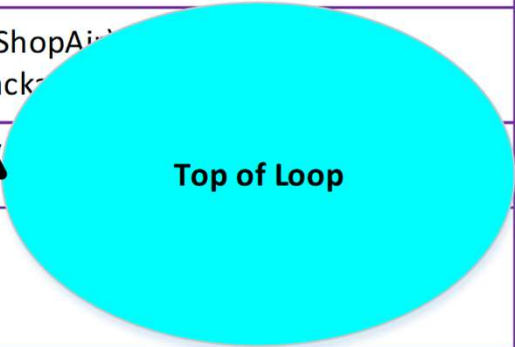
# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW() Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener



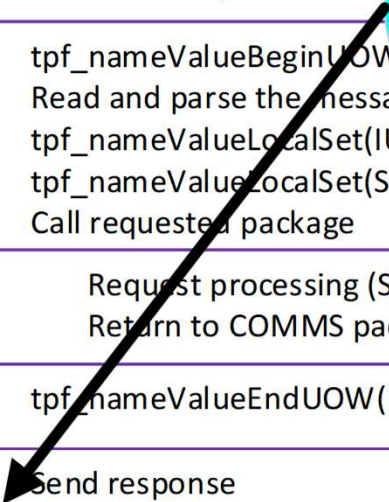
# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW(TPF_NVP_BEGIN_CLEAR_NVPS) Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	12 M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS pack	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW( <b>Process Query Message in the same manner</b>	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

# Name-value pairs API walkthrough – Multiple Requests Single ECB

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW() Read and parse the message tpf_nameValueLocalSet(IUOWID, ...) tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

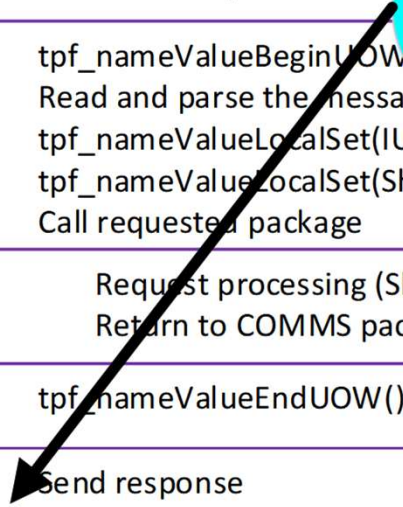
**Exit**  
 NVPC collection occurs  
 Horizontal (Exit) - 1 IO  
 IUOWID=11  
 MsgType=P1Listener



# Name-value pairs API walkthrough – Multiple Requests Single ECB

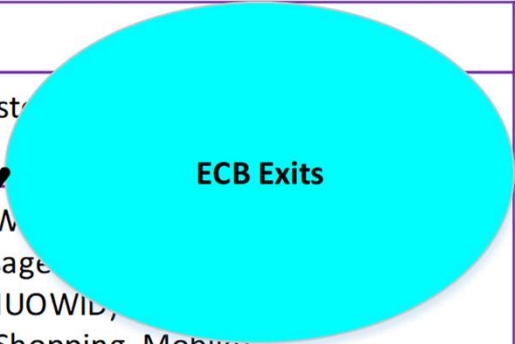
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW() Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener

**Exit**  
 NVPC collection occurs  
 Vertical (Owner Change) –  
 1 IO  
 IUOWID=11  
 IOwnerHi=COMMS



# Name-value pairs API walkthrough – Multiple Requests Single ECB

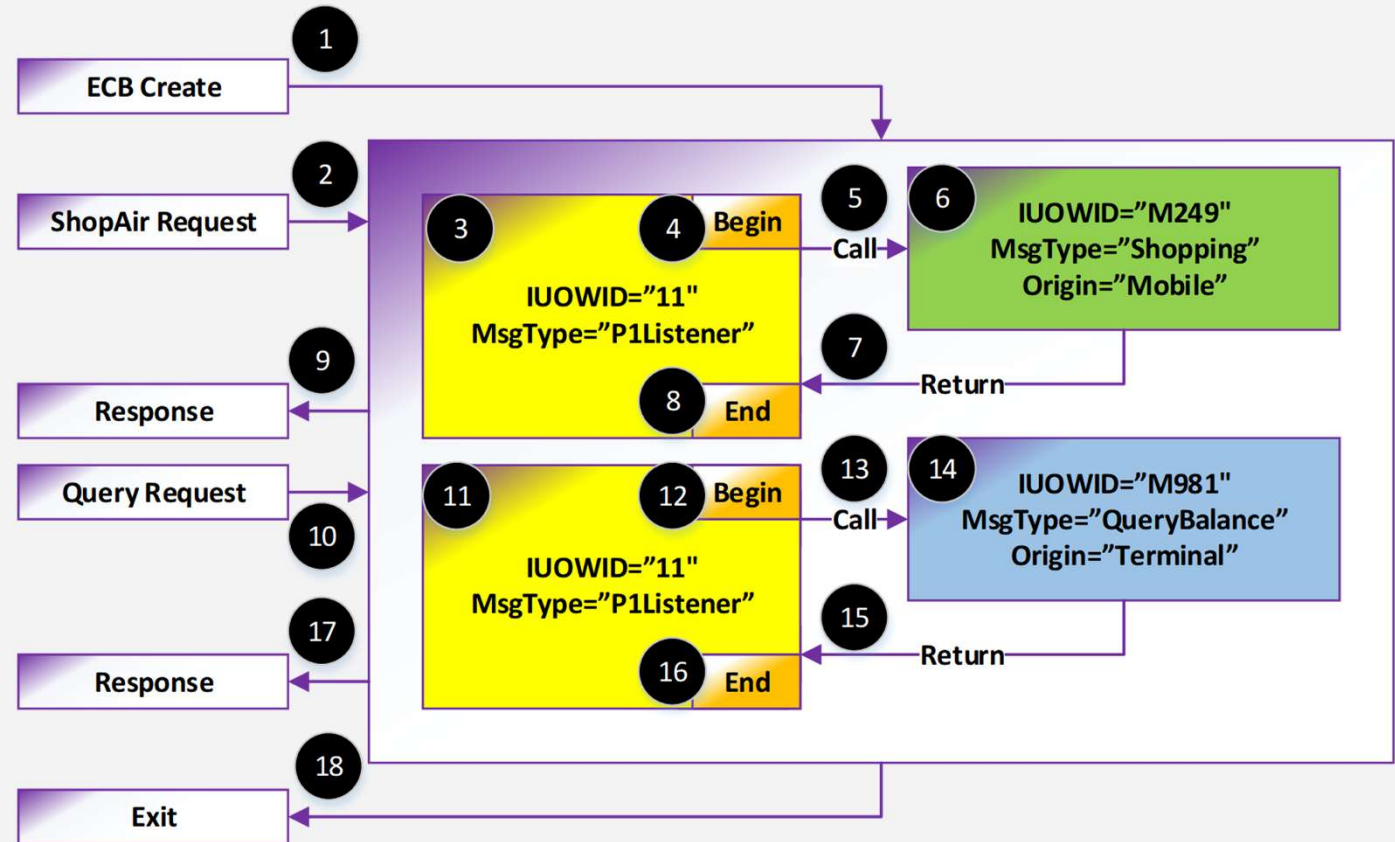
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
tpf_nameValueLocalSet(P1Listener) Socket Select Loop	COMMS	1	11	MsgType=P1Listener
tpf_nameValueBeginUOW() Read and parse the message tpf_nameValueLocalSet(IUOWID, tpf_nameValueLocalSet(Shopping, Mobile) Call requested package	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Request processing (ShopAir) Return to COMMS package	ShopAir	4	M249	MsgType=Shopping Origin=Mobile
tpf_nameValueEndUOW()	COMMS	1	M249	MsgType=Shopping Origin=Mobile
Send response exit	COMMS	1	11	MsgType=P1Listener



# Name-value pairs API walkthrough – Multiple Requests Single ECB

Row Type	Horizontal NVPs	Vertical NVPs	IOs
Horizontal (exit)	MsgType=P1Listener	none	2
Vertical (owner name change)	MsgType=P1Listener	IOwnerHi=COMMS	2
Horizontal (exit)	MsgType=shopping Origin=mobile	none	6
Vertical (owner name change)	MsgType=shopping Origin=mobile	IOwnerHi=COMMS	2
Vertical (owner name change)	MsgType=shopping Origin=mobile	IOwnerHi=ShopAir	4
Horizontal (exit)	MsgType=QueryBalance Origin=Terminal	none	6
Vertical (owner name change)	MsgType=QueryBalance Origin=Terminal	IOwnerHi=COMMS	2
Vertical (owner name change)	MsgType=QueryBalance Origin=Terminal	IOwnerHi=Query	4

# Name-value Pairs – Multiple Requests – Single ECB

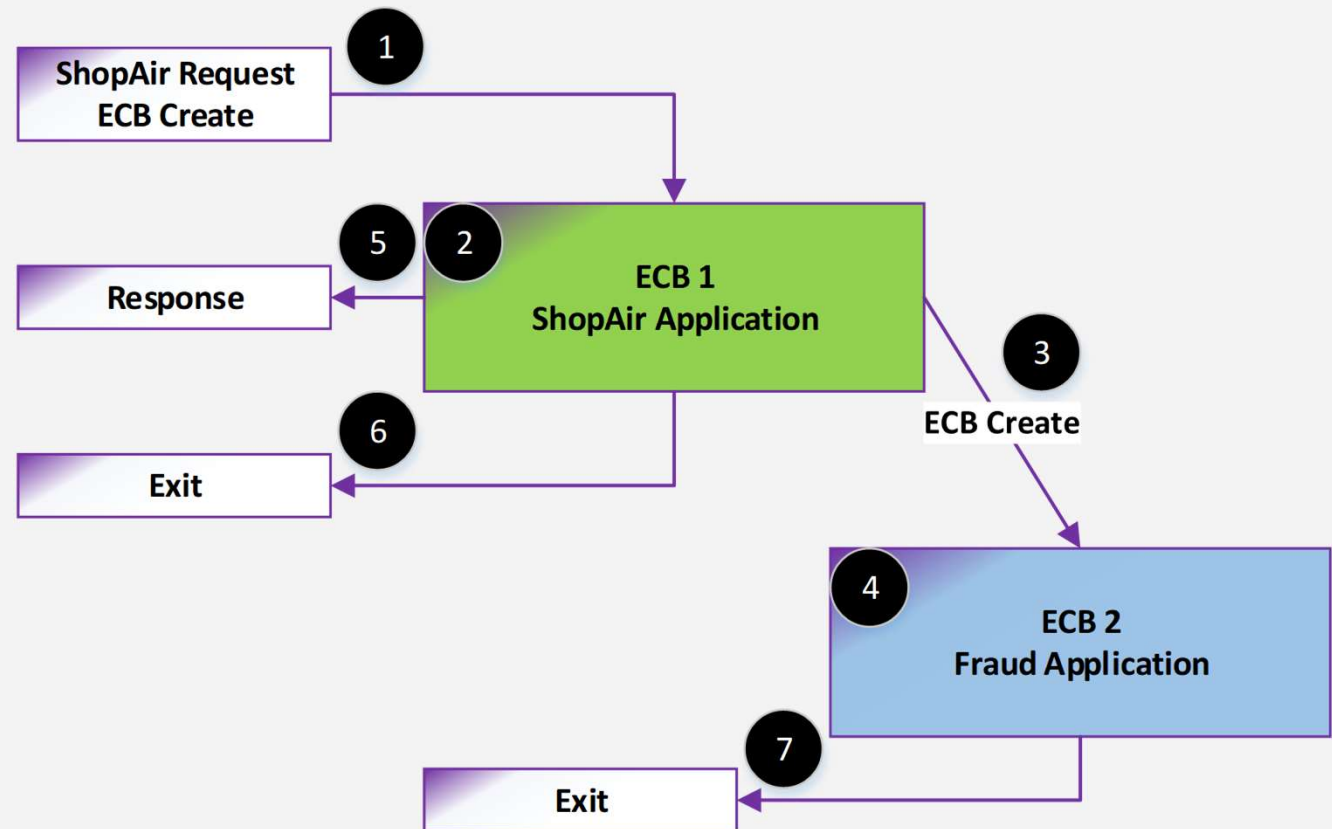




# Model: Single Request – ECB and children

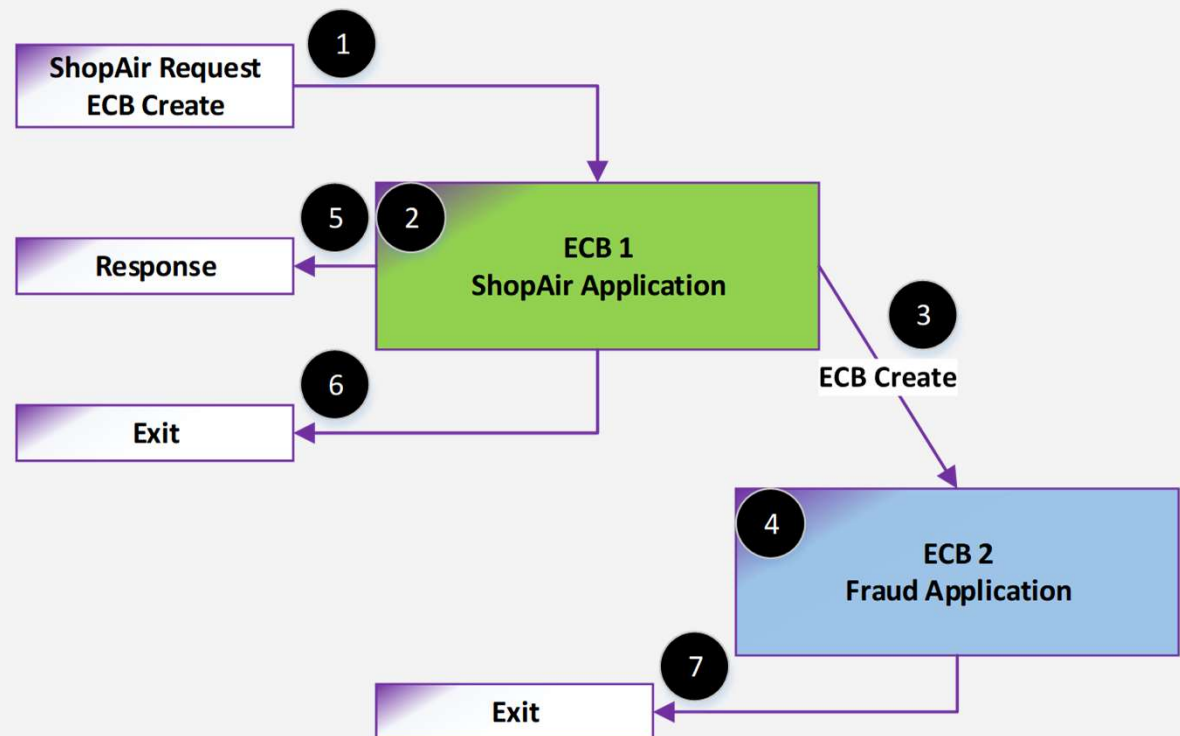
# What is this programming model? ECB and Children

- **What:** An ECB processing a single request creates a second ECB.
- **Problem:** How do you see the total Resources used? How do break the results down by each ECB?



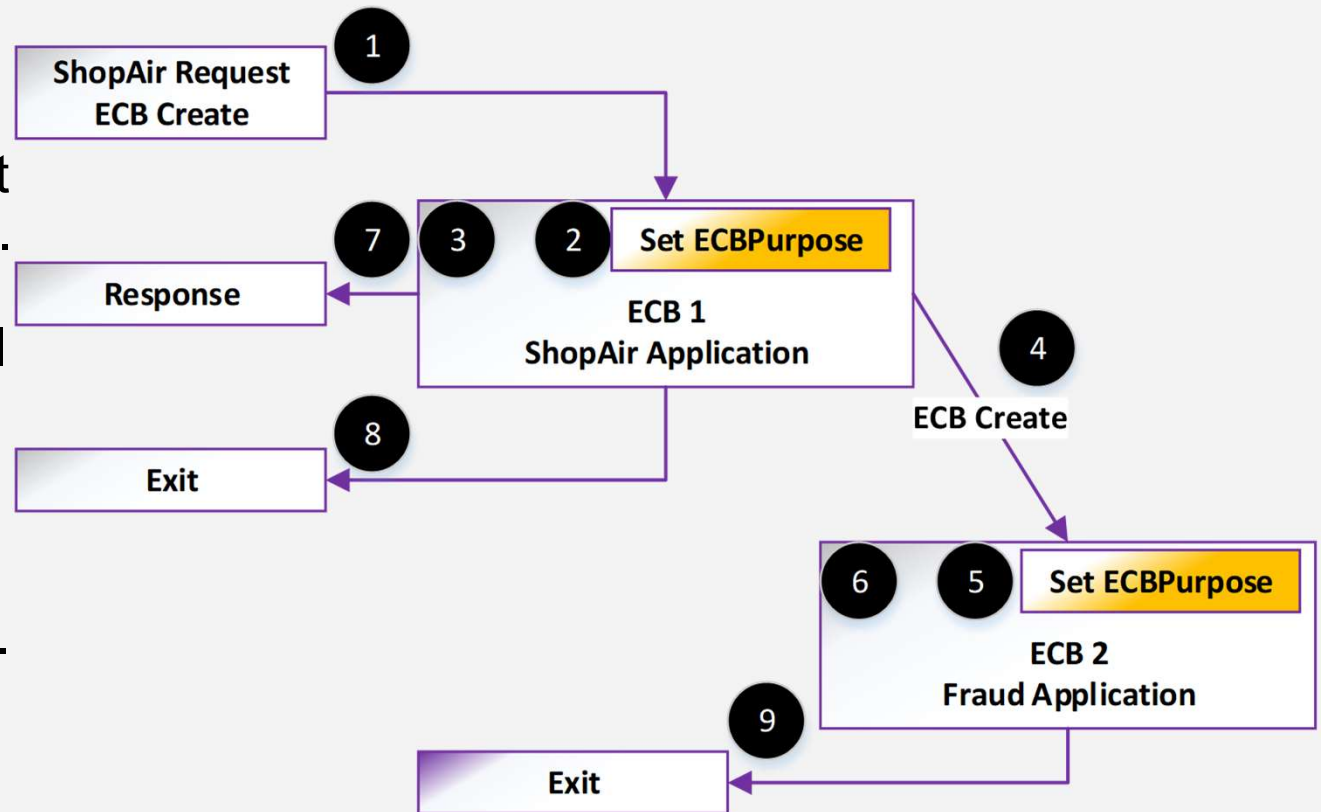
# What is this programming model? ECB and Children

- **Desired Results:**
  - Shopping UOW - 10 IOs
    - 4 IOs – ShopAir ECB
    - 6 IOs – Fraud ECB



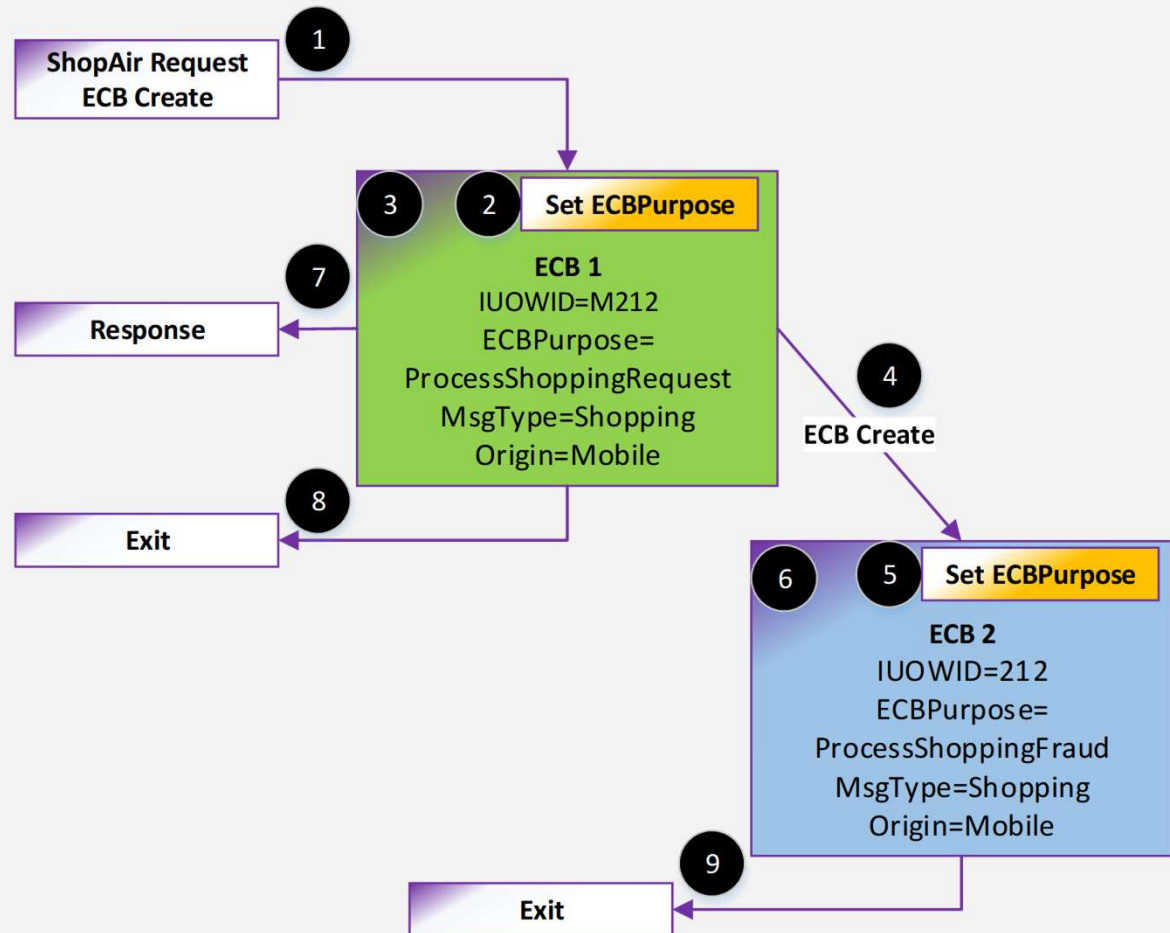
# What is this programming model? ECB and Children

- Solution:** Created ECBs inherit NVPs automatically (except CRETC and the like). Create an ECBPurpose Vertical NVP to understand resources used for each ECB. Owner names work as previously described.



# Name-value Pairs – ECB and Children

- **Solution:** Here is an example of the desired NVPs and the UOW Id.



# Name-value pairs API walkthrough – ECB and Children

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing CREMC Fraud ECB Send response exit	ShopAir	4	23 M212	ECBPurpose=ProcessShoppingRequest MsgType=Shopping Origin=Mobile
tpf_nameValueLocalSet(ECBPurpose) Do Fraud processing exit	Fraud	6	M212	ECBPurpose=ProcessShoppingFraud MsgType=Shopping Origin=Mobile

- Green AND Blue is the Shopping UOW.
- Green is the ProcessShoppingRequest ECBPurpose.
- Blue is the ProcessShoppingFraud ECBPurpose .
- ECB owner names, set and other mechanisms work the same and will not be discussed here.

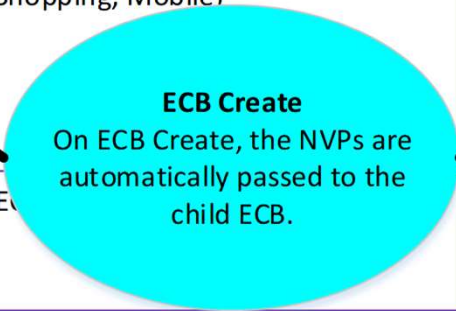
# Name-value pairs API walkthrough – ECB and Children

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing CREMC Fraud ECB Send response exit	ShopAir	4	23 M212	ECBPurpose=ProcessShoppingRequest MsgType=Shopping Origin=Mobile
tpf_nameValueLocalSet(E... Do Fraud processing exit	Fraud	6	M212	ECBPurpose=ProcessShoppingFraud MsgType=Shopping Origin=Mobile

**NVP Set**  
 When setting other NVPs in a multi ECB application, set an ECBPurpose NVP. In the NVPC policy, include ECBPurpose as a vertical NVP

# Name-value pairs API walkthrough – ECB and Children

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(Shopping, Mobile) Do ShopAir processing CREMC Fraud ECB Send response exit	ShopAir	4	23 M212	ECBPurpose=ProcessShoppingRequest MsgType=Shopping Origin=Mobile
tpf_nameValueLocalSet(E Do Fraud processing exit	Fraud	6	M212	ECBPurpose=ProcessShoppingFraud MsgType=Shopping Origin=Mobile

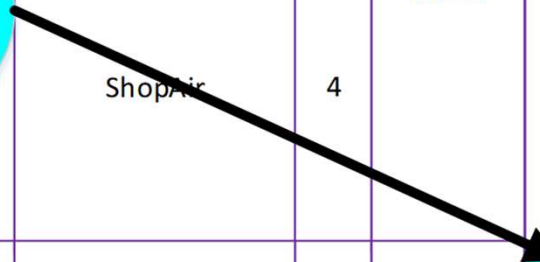
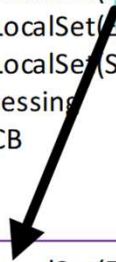




# Name-value pairs API walkthrough – ECB and Children

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(ShopAir) tpf_nameValueLocalSet(ShopAir) tpf_nameValueLocalSet(ShopAir) Do ShopAir processing CREMC Fraud ECB Send response exit	ShopAir	4	23 M212	ECBPurpose=ProcessShoppingRequest MsgType=Shopping Origin=Mobile
tpf_nameValueLocalSet(ECBPurpose) Do Fraud processing exit	Fraud	6	M212	ECBPurpose=ProcessShoppingFraud MsgType=Shopping Origin=Mobile

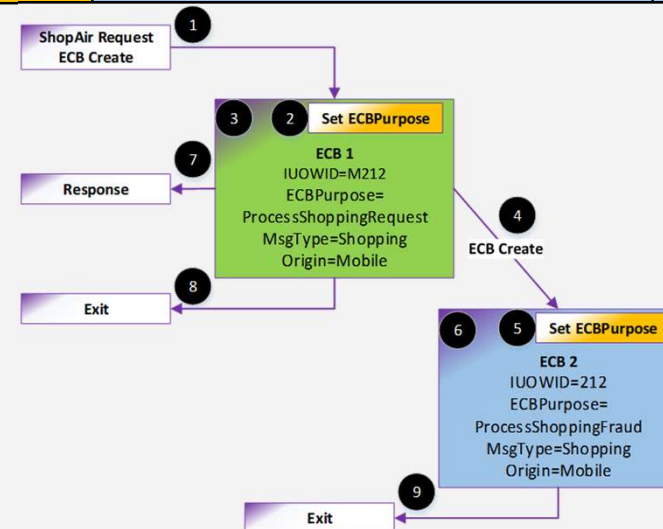
**NVP Set**  
 When setting other NVPs in a multi ECB application, set an ECBPurpose NVP. In the NVPC policy, include ECBPurpose as a vertical NVP



## Name-value pairs API walkthrough – ECB and Children

Row Type	Horizontal NVPs	Vertical NVPs	IOs
Horizontal (exit)	MsgType=shopping Origin=mobile	none	10
Vertical (owner name change)	MsgType=shopping Origin=mobile	ECBPurpose= ProcessShoppingRequest	4
Vertical (owner name change)	MsgType=shopping Origin=mobile	ECBPurpose= ProcessShoppingFraud	6

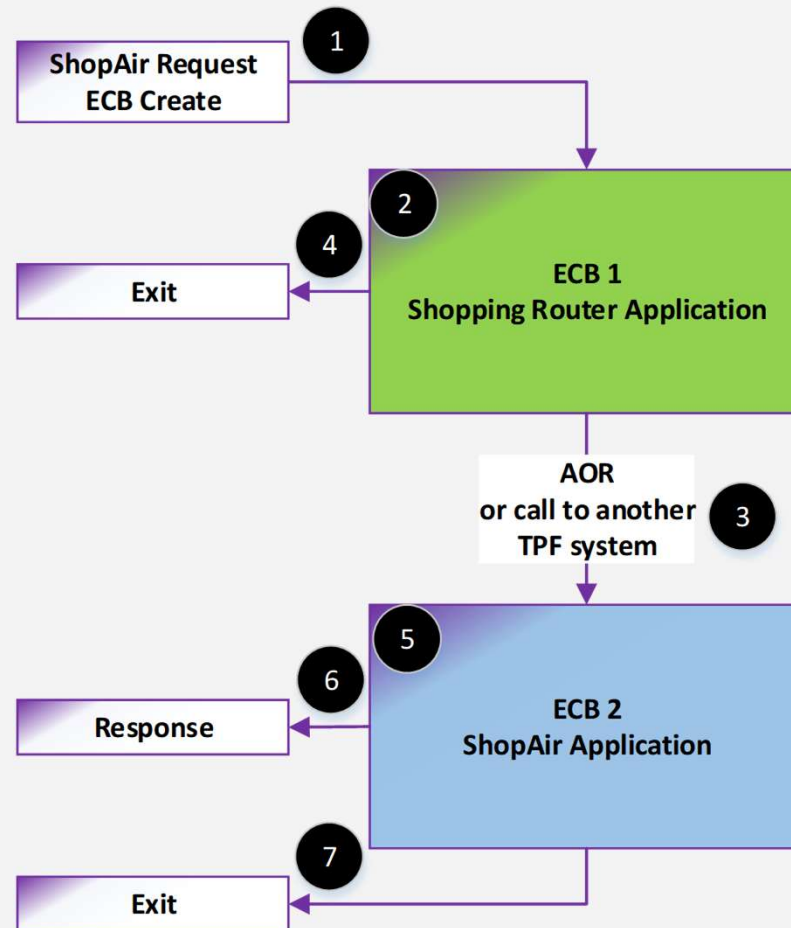
- Additional verticals would be included for ECB owner names.



# Model: Single Request – AOR and Remote

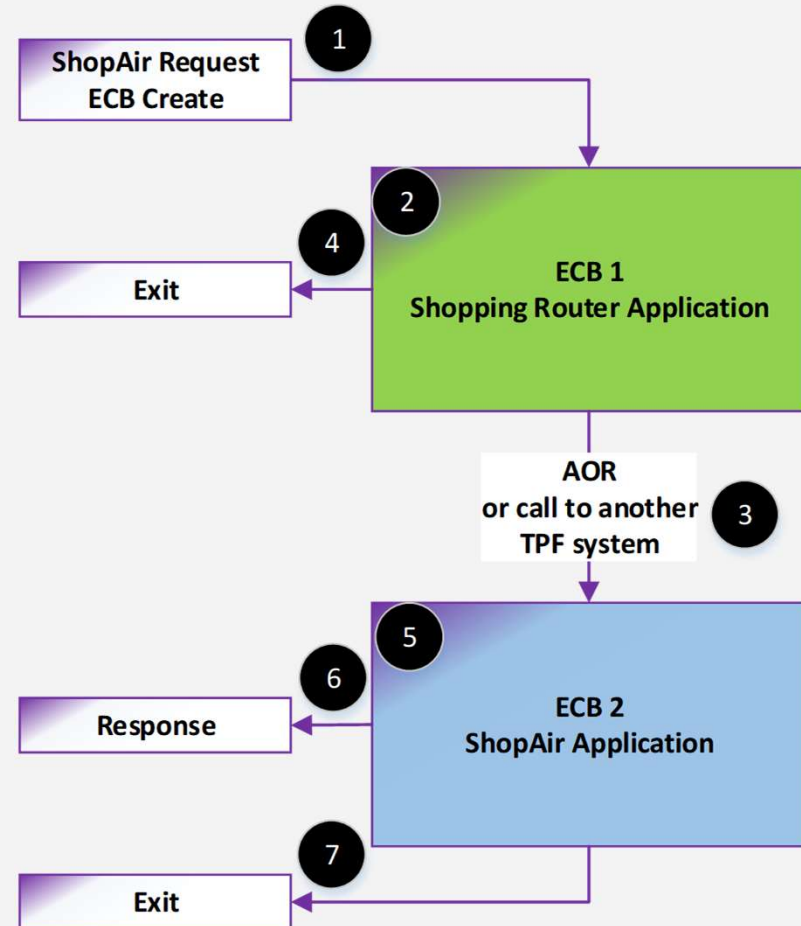
# What is this programming model? AOR and Remote

- **What:** An ECB processing a single request continues processing, through AOR or by sending a message to another z/TPF system, with a second ECB.
- **Problem:** How do you collect NVPs to pass to the second ECB? How do you see the total resources used? How do you break the results down by each ECB?



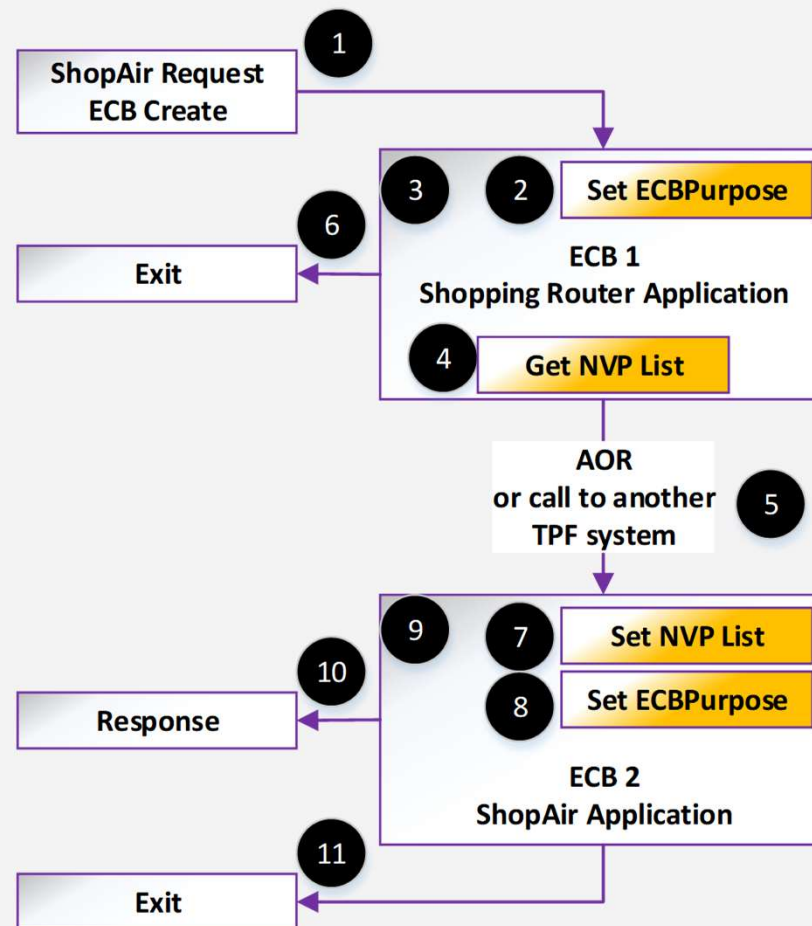
# What is this programming model? AOR and Remote

- **Desired Results:**
  - Shopping UOW - 8 IOs
    - 1 IOs – Shopping Router ECB
    - 7 IOs – ShopAir ECB



# What is this programming model? AOR and Remote

- Solution:** Use the NVP get/set list APIs to retrieve the NVPs. Save or send the get list block with your message. Create an ECBPurpose Vertical NVP to understand resources used for each ECB. Owner names work as previously described.



## Name-Value Pairs APIs: Get and Set List - Future

- Purpose: These APIs are used to save and restore all of the NVPs from or to an ECB. They allow you to pass the list of NVPs to another ECB on a remote system, on an AOR, or etc.
- `tpf_nameValueLocalGetSize`:
  - Get the size of the storage that is needed to save the name-value pair list to a memory location by using the `tpf_nameValueLocalGetList` function.
- `tpf_nameValueLocalGetList`:
  - Copies the name-value pair list from an ECB to a memory location.
  - This includes the IUOWID.
  - If you specify an encryption key, the list of NVPs will be encrypted. If you do not specify an encryption key, any non-displayable NVPs will be omitted from the list.

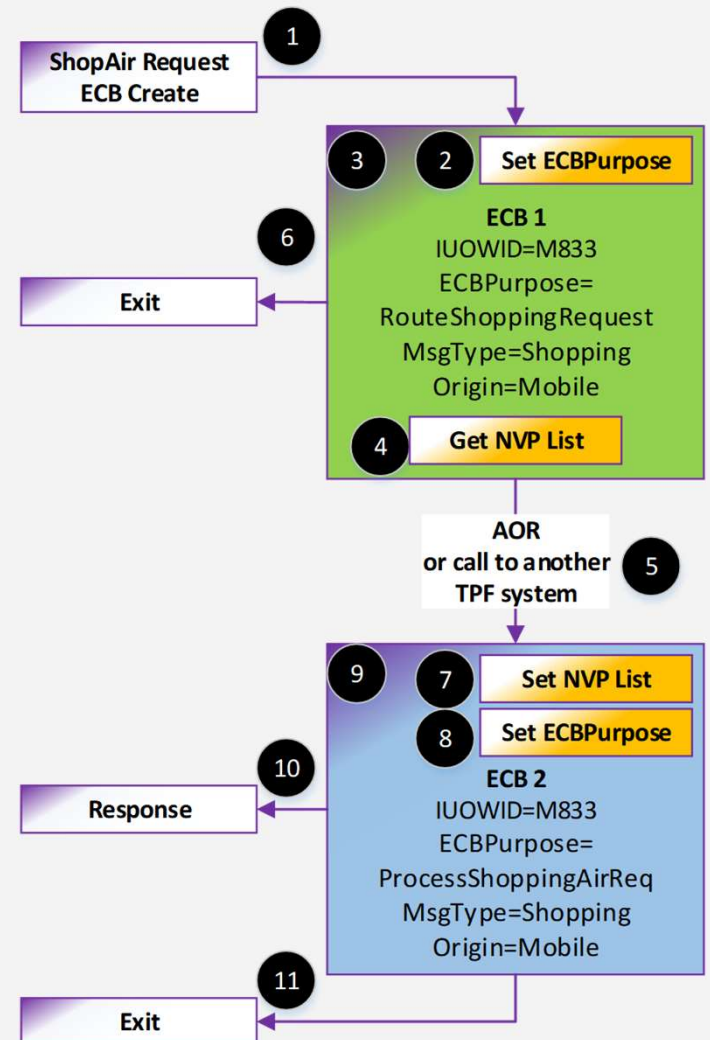
## Name-Value Pairs APIs: Get and Set List - Future

- `tpf_nameValueLocalSetList`:
  - Copy the name-value pair list from a memory location to an ECB.
  - This includes the IUOWID.
  - This API provides a replace or append option for how the NVPs are set from the list to the ECB. However, the IUOWID is always replaced.



# Name-value Pairs - AOR and Remote

- **Solution:** Here is an example of the desired NVPs and the UOW Id.



## Name-value pairs API walkthrough – AOR and Remote

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(Shopping, Mobile) Do Shopping Router processing tpf_nameValueLocalGetSize() Allocate Message Buffer or Save Area tpf_nameValueLocalGetList() Send Message or Write the Save Area exit	ShopRter	1	62 M833	ECBPurpose=RouteShoppingRequest MsgType=Shopping Origin=Mobile
Read and parse the Message or Save Area tpf_nameValueLocalSetList() tpf_nameValueLocalSet(ECBPurpose) Do ShopAir processing Send response exit	ShopAir	7	49 M833	ECBPurpose=ProcessShoppingAirReq MsgType=Shopping Origin=Mobile

- Green AND Blue is the Shopping UOW.
- Green is the RouteShoppingRequest ECBPurpose.
- Blue is the ProcessShoppingAirReq ECBPurpose .
- ECB owner names, set and other mechanisms work the same and will not be discussed here.

# Name-value pairs API walkthrough – AOR and Remote

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(Shopping, Mobile) Do Shopping Router processing tpf_nameValueLocalGetSize() Allocate Message Buffer or Save Area tpf_nameValueLocalGetList() Send Message or Write to the Save Area exit	ShopRter	1	62 M833	ECBPurpose=RouteShoppingRequest MsgType=Shopping Origin=Mobile
Read and parse the Message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) Do ShopAir processing Send response exit	ShopAir	7	49 M833	ECBPurpose=ProcessShoppingAirReq MsgType=Shopping Origin=Mobile

**NVP Set**  
 When setting other NVPs in a multi ECB application, set an ECBPurpose NVP. In the NVPC policy, include ECBPurpose as a vertical NVP

# Name-value pairs API walkthrough – AOR and Remote

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(Shopping, Mobile) Do Shopping Router processing tpf_nameValueLocalGetSize() Allocate Message Buffer or Save Area tpf_nameValueLocalGetList() Send Message or Write the Save Area exit	ShopRter	1	62 M833	ECBPurpose=RouteShoppingRequest MsgType=Shopping Origin=Mobile
Read and parse the Message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) Do ShopAir processing Send response exit	ShopAir	7	49 M833	ECBPurpose=ProcessShoppingAirReq MsgType=Shopping Origin=Mobile

**Prepare to save or send message**

# Name-value pairs API walkthrough – AOR and Remote

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(Shopping, Mobile) Do Shopping Router processing tpf_nameValueLocalGetSize() Allocate Message Buffer or Save Area tpf_nameValueLocalGetList() Send Message or Write the Save Area exit	ShopRter	1	62 M833	ECBPurpose=RouteShoppingRequest MsgType=Shopping Origin=Mobile
Read and parse the Message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) Do ShopAir processing Send response exit	ShopAir	7	49 M833	ECBPurpose=ProcessShoppingAirReq MsgType=Shopping Origin=Mobile

**GetSize API**  
 Get the size of the area  
 required to store the NVPs  
 and their control structure.

# Name-value pairs API walkthrough – AOR and Remote

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(Shopping, Mobile) Do Shopping Router processing tpf_nameValueLocalGetSize() Allocate Message Buffer or Save Area tpf_nameValueLocalGetList() Send Message or Write the Save Area exit	ShopRter	1	62 M833	ECBPurpose=RouteShoppingRequest MsgType=Shopping Origin=Mobile
Read and parse the Message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) Do ShopAir processing Send response exit	ShopAir	7	49 M833	ECBPurpose=ProcessShoppingAirReq MsgType=Shopping Origin=Mobile

**Allocate the message buffer  
or save area (system heap,  
file, etc)**

# Name-value pairs API walkthrough – AOR and Remote

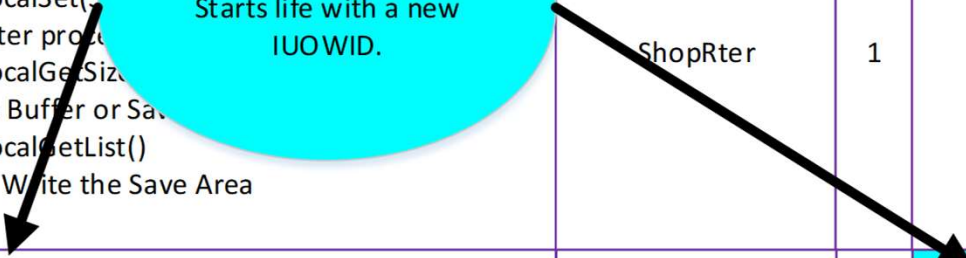
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(Shopping, Mobile) Do Shopping Router processing tpf_nameValueLocalGetSize() Allocate Message Buffer or Save Area tpf_nameValueLocalGetList() Send Message or Write the Save Area exit	ShopRter	1	62 M833	ECBPurpose=RouteShoppingRequest MsgType=Shopping Origin=Mobile
Read and parse the Message tpf_nameValueLocalSet(L... tpf_nameValueLocalSet(L... Do ShopAir processing Send response exit	ShopAir	7	49 M833	ECBPurpose=ProcessShoppingAirReq MsgType=Shopping Origin=Mobile

**Get List API**  
 Use the get list API to retrieve the NVPs to the allocated memory including the IUOWID. If non-displayable NVPs are included specify an encryption key.

# Name-value pairs API walkthrough – AOR and Remote

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ECBPurpose) tpf_nameValueLocalSet(MsgType) Do Shopping Router processing tpf_nameValueLocalGetSize() Allocate Message Buffer or Save Area tpf_nameValueLocalSetList() Send Message or Write the Save Area exit	ShopRter	1	62 M833	ECBPurpose=RouteShoppingRequest MsgType=Shopping Origin=Mobile
Read and parse the Message or Save Area tpf_nameValueLocalSetList() tpf_nameValueLocalSet(ECBPurpose) Do ShopAir processing Send response exit	ShopAir	7	49 M833	ECBPurpose=ProcessShoppingAirReq MsgType=Shopping Origin=Mobile

**New ECB**  
Starts life with a new IUOWID.

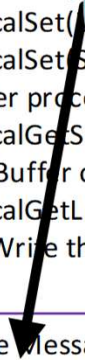




# Name-value pairs API walkthrough – AOR and Remote

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(ShopRter) Do Shopping Router processing tpf_nameValueLocalGetSize() Allocate Message Buffer or Save Area tpf_nameValueLocalGetList() Send Message or Write the Save Area exit	ShopRter	1	62 M833	ECBPurpose=RouteShoppingRequest MsgType=Shopping Origin=Mobile
Read and parse the Message or Save Area tpf_nameValueLocalSetList() tpf_nameValueLocalSet(ECBPurpose) Do ShopAir processing Send response exit	ShopAir	7	49 M833	ECBPurpose=ProcessShoppingAirReq MsgType=Shopping Origin=Mobile

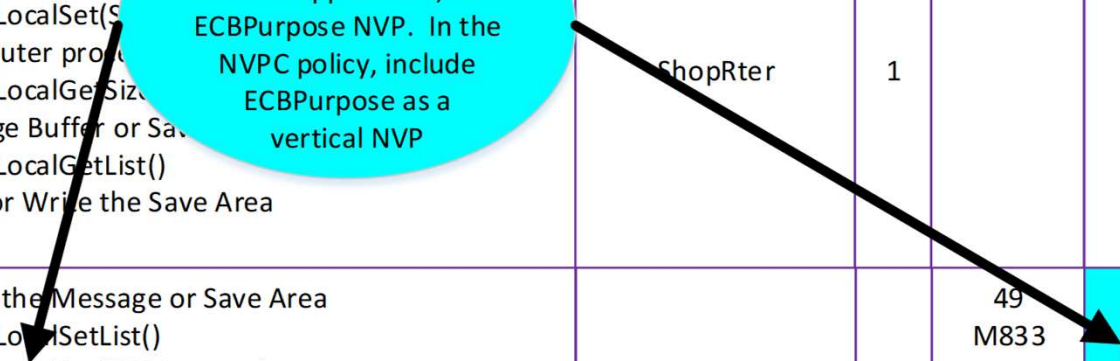
**Set List API**  
 The set list API sets the NVPs from the memory block into the ECB including the IUOWID.



# Name-value pairs API walkthrough – AOR and Remote

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(Origin) tpf_nameValueLocalSet(MsgType) Do Shopping Router processing tpf_nameValueLocalGetSize() Allocate Message Buffer or Save Area tpf_nameValueLocalGetList() Send Message or Write the Save Area exit	ShopRter	1	62 M833	ECBPurpose=RouteShoppingRequest MsgType=Shopping Origin=Mobile
Read and parse the Message or Save Area tpf_nameValueLocalGetList() tpf_nameValueLocalSet(ECBPurpose) Do ShopAir processing Send response exit	ShopAir	7	49 M833	ECBPurpose=ProcessShoppingAirReq MsgType=Shopping Origin=Mobile

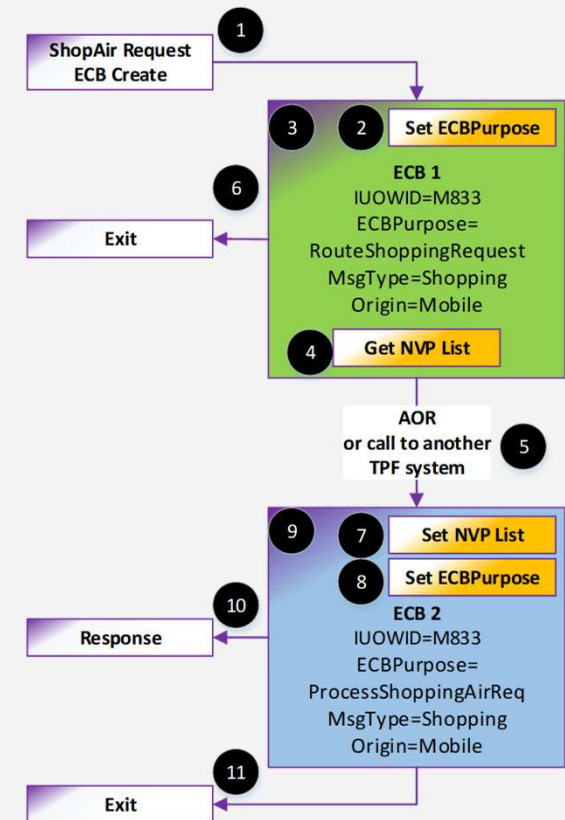
**NVP Set**  
 When setting other NVPs in a multi ECB application, set an ECBPurpose NVP. In the NVPC policy, include ECBPurpose as a vertical NVP



# Name-value pairs API walkthrough – AOR and Remote

Row Type	Horizontal NVPs	Vertical NVPs	IOs
Horizontal (exit)	MsgType=shopping Origin=mobile	none	8
Vertical (owner name change)	MsgType=shopping Origin=mobile	ECBPurpose= RouteShoppingRequest	1
Vertical (owner name change)	MsgType=shopping Origin=mobile	ECBPurpose= ProcessShoppingAirReq	7

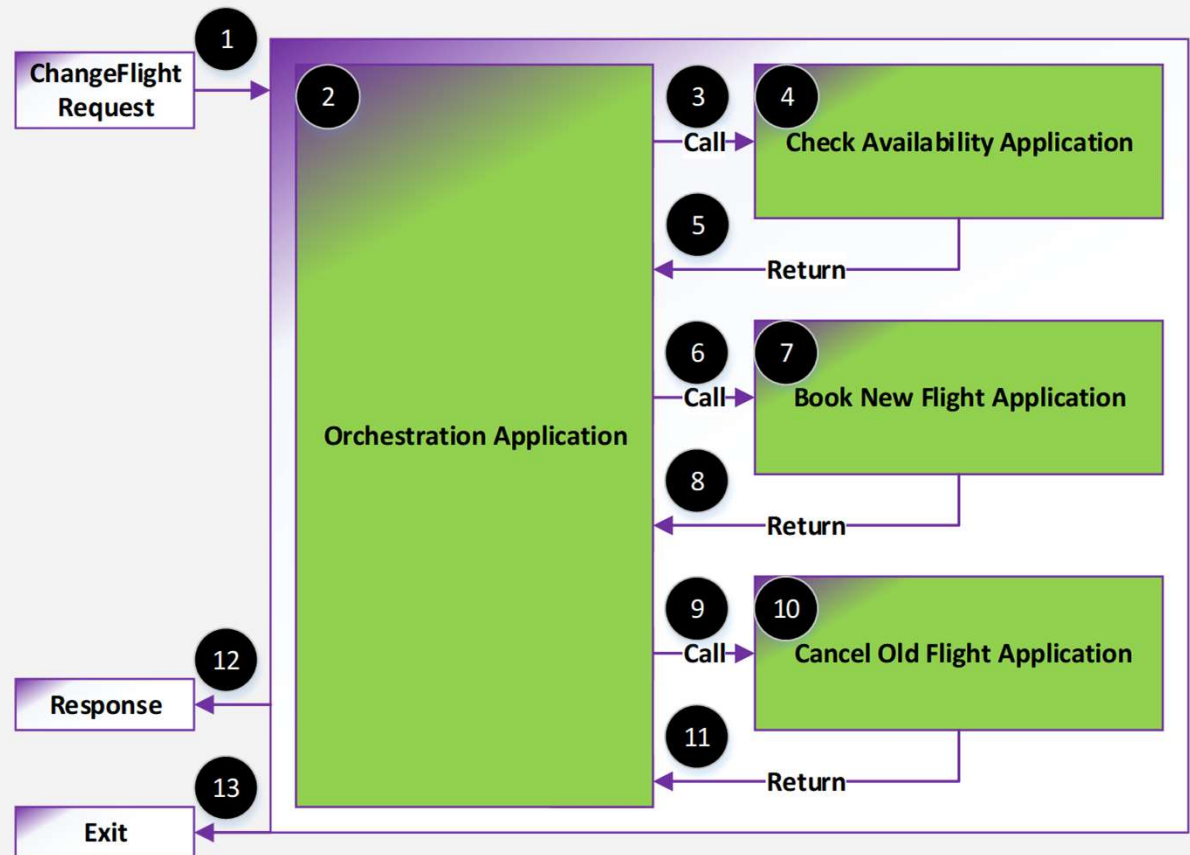
- Additional verticals would be included for ECB owner names.



# Model: Single Request – Single ECB – Orchestration

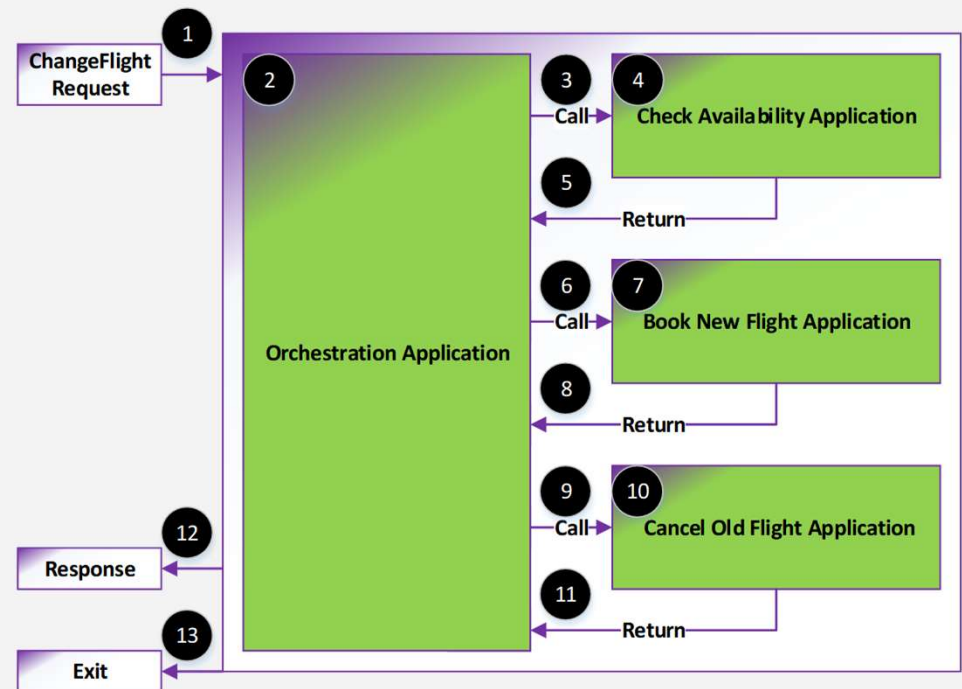
# What is this programming model? Single ECB – Orchestration

- **What:** An ECB processing a single request calls various existing services (not necessarily REST).
- **Problem:** How do you see the total resources used? How do you break the results down by each service?



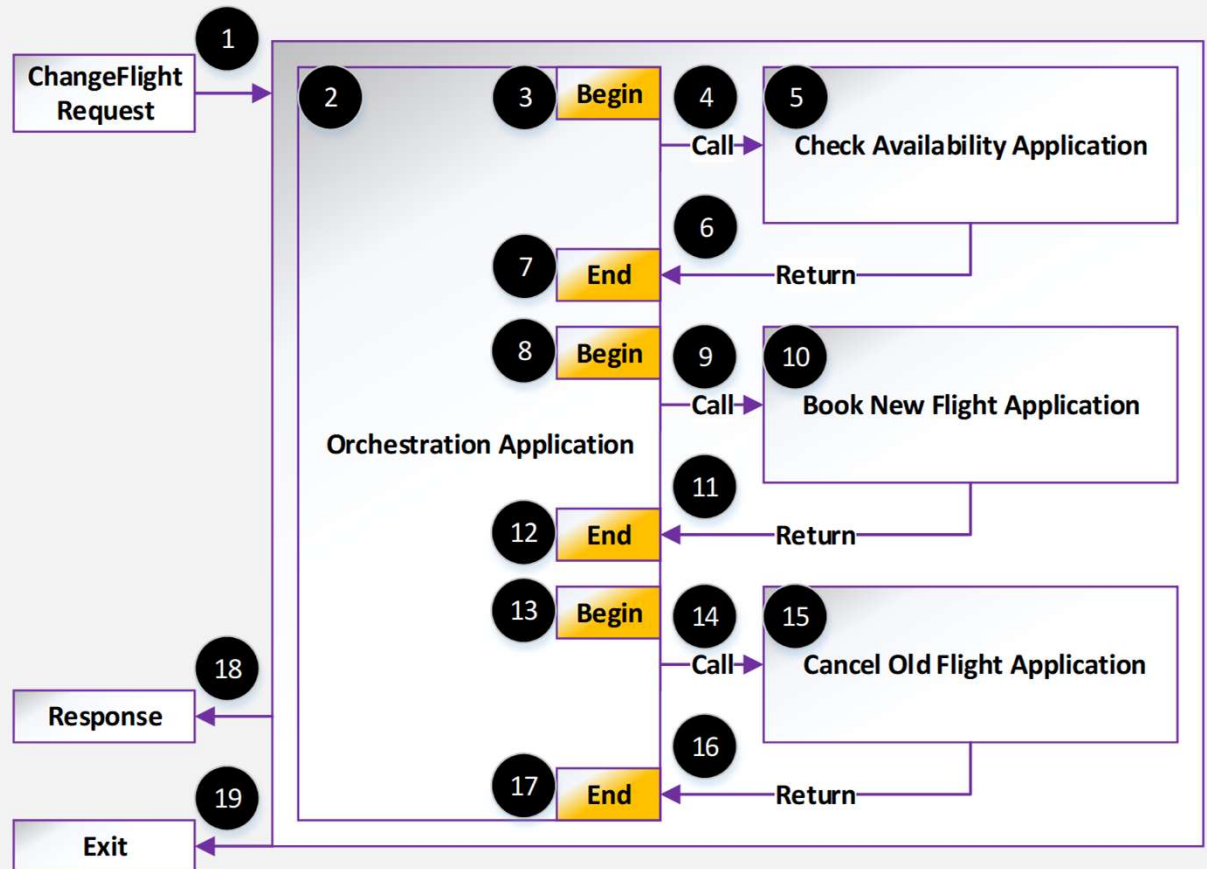
# What is this programming model? Single ECB – Orchestration

- **Desired Results:**
  - ChangeFlight UOW - 21 IOs
    - 4 IOs – ChangeFlightOrch Service
    - 4 IOs – CheckAvailability Service
    - 10 IOs – BookFlight Service
    - 3 IOs – CancelFlight Service



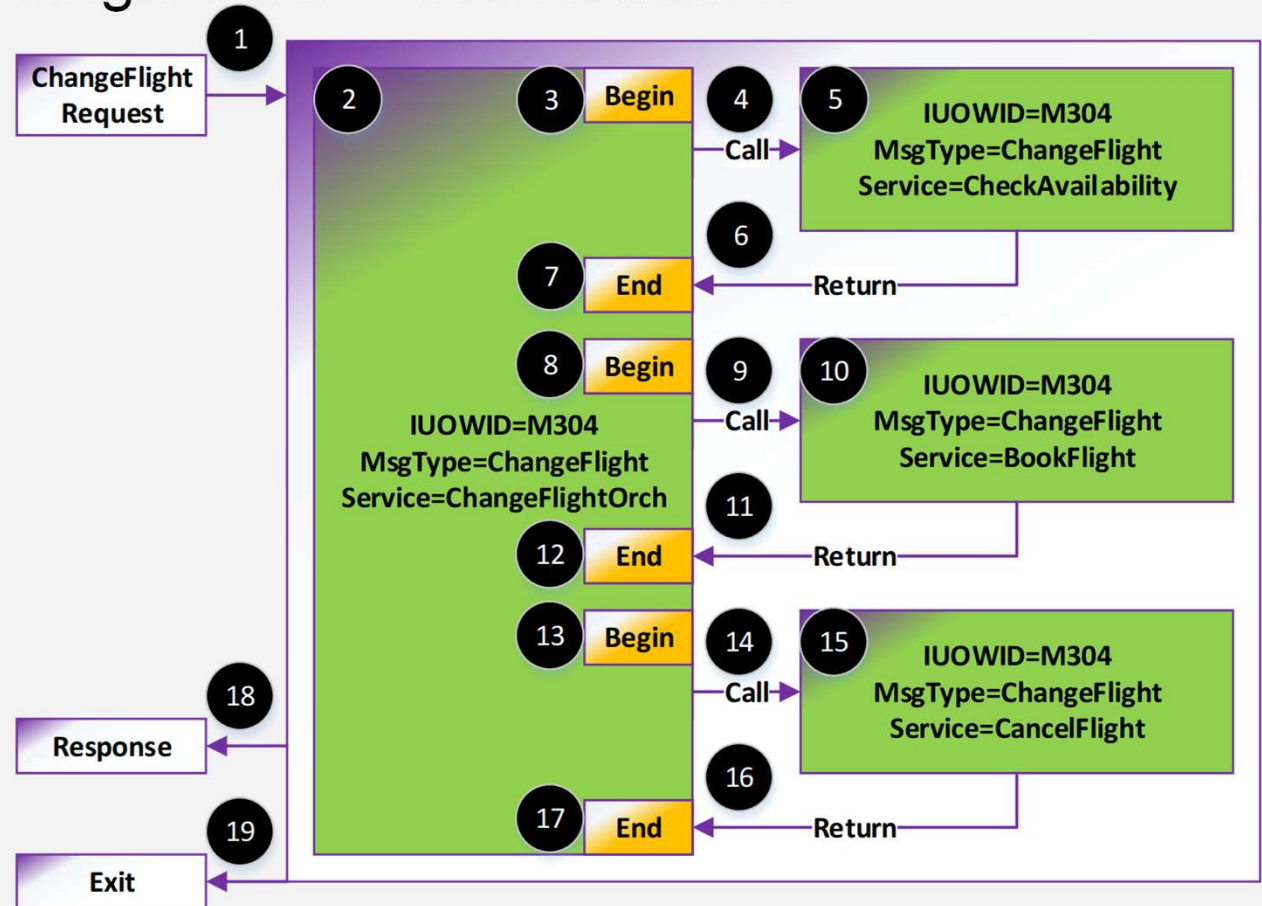
# What is this programming model? Single ECB – Orchestration

- Solution:** Use the NVP begin and end APIs around calls to each service to cause collection to occur. In conjunction, have a vertical NVP Service to breakdown the resources used by each service.



# Name-value Pairs – Single ECB – Orchestration

- Solution:** Here is an example of the desired NVPs and the UOW Id.





# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Book Flight processing Return to ChangeFlight service	Various Owner Names	10	M304	MsgType=ChangeFlight Service=BookFlight
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call Cancel Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Cancel Flight processing Return to ChangeFlight service	Various Owner Names	3	M304	MsgType=ChangeFlight Service=CancelFlight
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call Cancel Flight service Exit	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

- Green is the total ChangeFlight UOW.
- Yellow is the ChangeFlightOrch service.
- Blue is the CheckAvailability service.
- Orange is the BookFlight service.
- Purple is the CancelFlight service.
- ECB owner names, set and other mechanisms work the same and will not be discussed here.

# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**NVP Set**  
 Set a horizontal NVP for the type of message that started our request. This NVP will have some special treatment going forward.

# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**NVP Set**  
Set a Vertical NVP for each individual service.

# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_...) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**NVP Begin**  
 Prior to calling a service, issue begin to cause collection for the work done for the orchestration phase of processing.

# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch



# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**NVP Begin**  
 NVPC collection occurs  
 Horizontal (Exit) - 1 IO  
 IUOWID=M304  
 MsgType=ChangeFlight  
 Service=ChangeFlightOrch

# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**NVP Begin**  
 NVPC collection occurs  
 Vertical (Owner Change) –  
 1 IO  
 IUOWID=M304  
 IOwnerHi=

# Name-value pairs API walkthrough – Single ECB – Orchestration

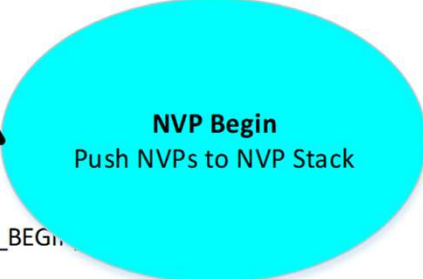
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN...) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**NVP Begin**  
Zero IO Counter as if it's a new ECB



# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch



# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**NVP Begin**  
 Since this is an orchestration case where processing continues, the keep parameter allows the NVPs to remain unchanged. IUOWID doesn't change either.

# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_...) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**ECB Owner Name**  
Changes as normal as you move between packages by configuration.

# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	NVP Set	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEE... Call ChangeFlight service	Here is our special handling of MsgType, we test if it has already been set. If so, we don't change it. It is horizontal...set and do not change.	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service		Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEE... Call Book Flight service		Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

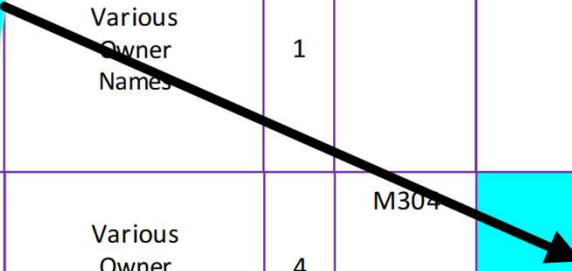
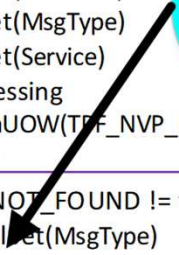
## Name-value pairs API walkthrough – Single ECB – Orchestration

- IBM may modify or provide additional NVP APIs to facilitate the testing and setting of the horizontal NVPs such as MsgType for the orchestration application model. Stay tuned...

# Name-value pairs API walkthrough – Single ECB – Orchestration

Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_...) Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**NVP Set**  
 Here is our special handling of Service, we always set it. It is vertical...it denotes the piece parts.



# Name-value pairs API walkthrough – Single ECB – Orchestration

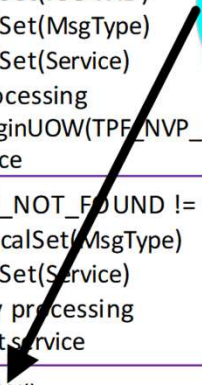
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEE Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability Processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEE Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**Service is complete, return to caller, ECB owner name change occurs by automatic restore and collection occurs as normal.**

# Name-value pairs API walkthrough – Single ECB – Orchestration

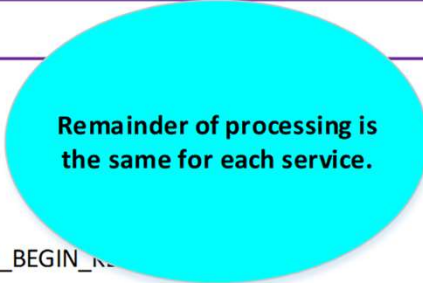
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEE Call ChangeFlight service	Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEE Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

**End NVP API**  
 Occurs as normal: call  
 accounting user exit,  
 horizontal collection, vertical  
 collection, reset counters to  
 0, pop the NVPs.





# Name-value pairs API walkthrough – Single ECB – Orchestration

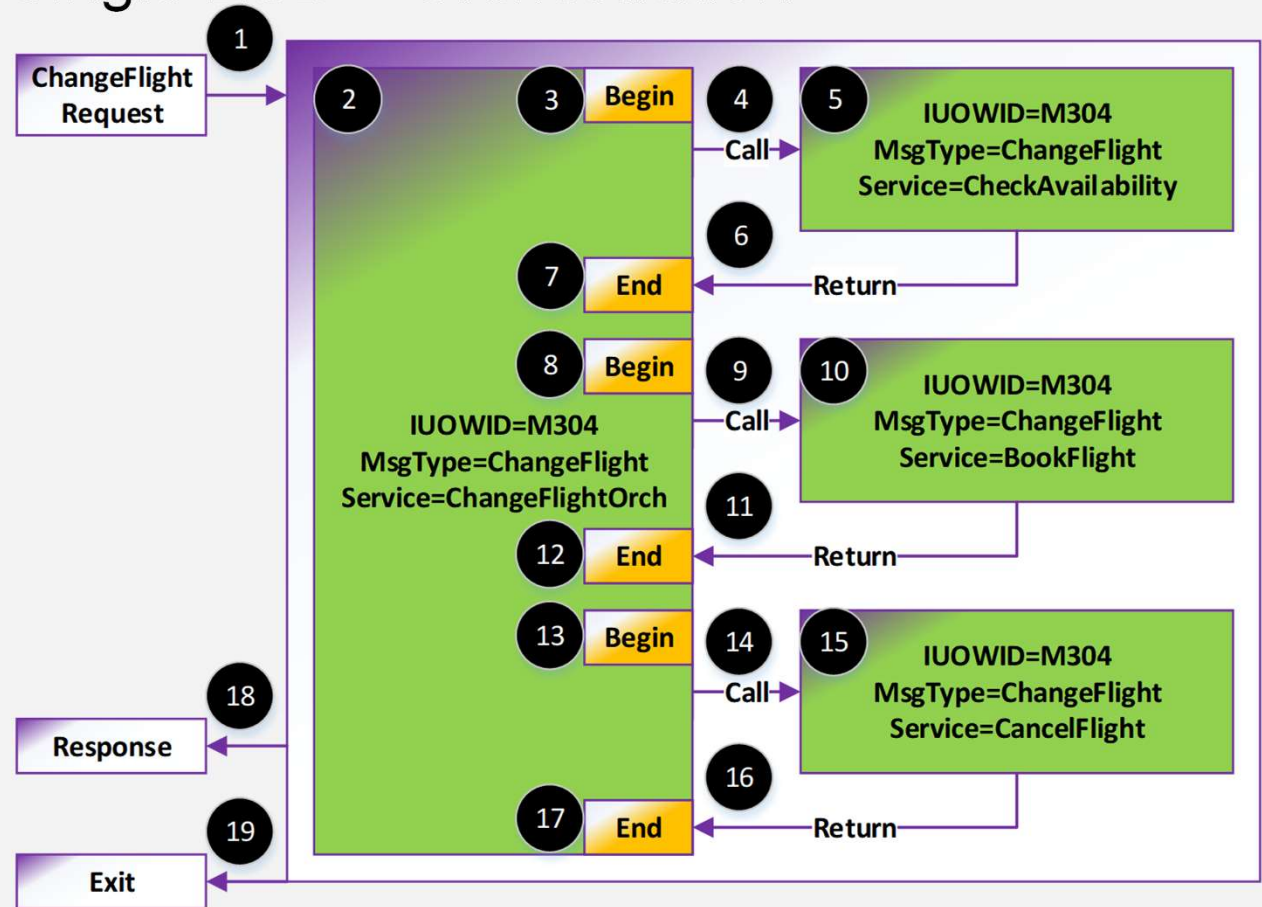
Logic:	ECB Owner Name:	IOs	IUOWID	NVPs
Read and parse the message tpf_nameValueLocalSet(IUOWID) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_...) Call ChangeFlight service	 Various Owner Names	1	89 M304	MsgType=ChangeFlight Service=ChangeFlightOrch
If (TPF_NAMEVALUE_NOT_FOUND != tpf_nameValueLocalGet(MsgType)) tpf_nameValueLocalSet(MsgType) tpf_nameValueLocalSet(Service) Do Check Availability processing Return to ChangeFlight service	Various Owner Names	4	M304	MsgType=ChangeFlight Service=CheckAvailability
tpf_nameValueEndUOW() Do Change Flight processing Call tpf_nameValueBeginUOW(TPF_NVP_BEGIN_KEEP_NVPS) Call Book Flight service	Various Owner Names	1	M304	MsgType=ChangeFlight Service=ChangeFlightOrch

## Name-value pairs API walkthrough – Single ECB – Orchestration

Row Type	Horizontal NVPs	Vertical NVPs	IOs
Horizontal (exit)	MsgType=ChangeFlight	none	21
Vertical (owner name change)	MsgType=ChangeFlight	Service=ChangeFlightOrch	4
Vertical (owner name change)	MsgType=ChangeFlight	Service=CheckAvailability	4
Vertical (owner name change)	MsgType=ChangeFlight	Service=BookFlight	10
Vertical (owner name change)	MsgType=ChangeFlight	Service=CancelFlight	3

- Additional verticals would be included for ECB owner names.

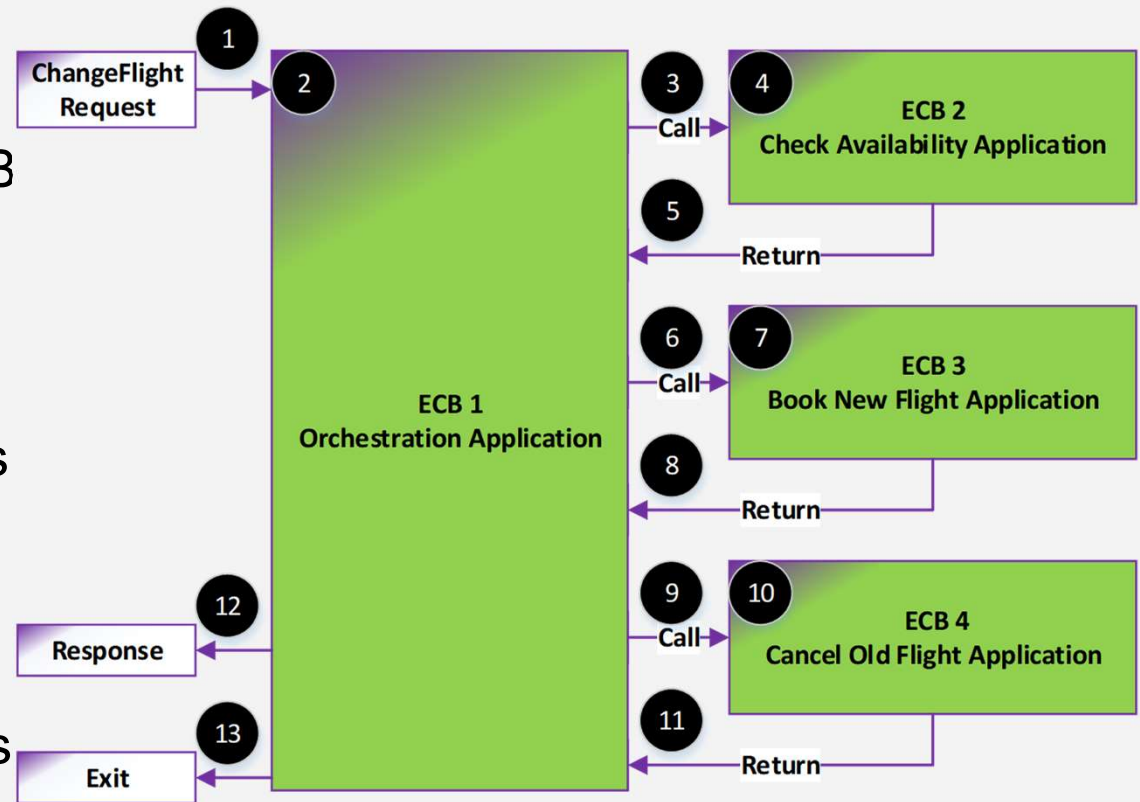
# Name-value Pairs – Single ECB – Orchestration



# Model: Single Request – Multiple ECB – Orchestration

# What is this programming model? Multiple ECB – Orchestration

- **What:** The multiple ECB orchestration case model is very similar to the single ECB orchestration model. An ECB processing a single request calls various existing services (not necessarily REST) where each service is performed by a new ECB.
- **Problem:** How do you see the total resources used? How do you break the results down by each service?

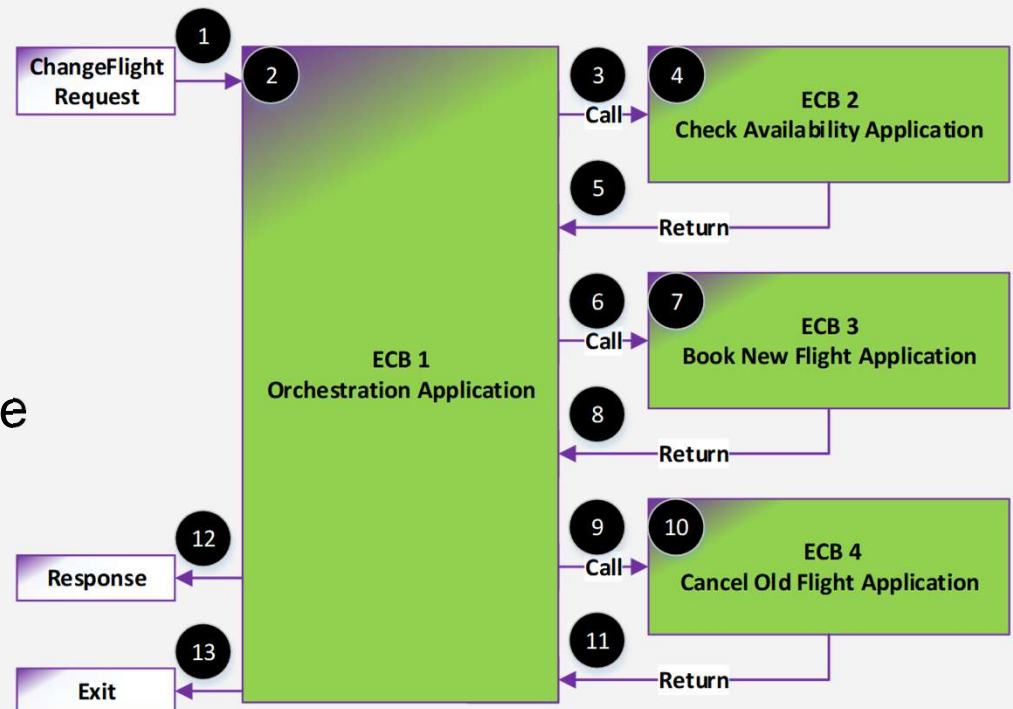


## What is this programming model? Multiple ECB – Orchestration

- **Desired Results:**
  - ChangeFlight UOW - 21 IOs
    - 4 IOs – ChangeFlightOrch Service
    - 4 IOs – CheckAvailability Service
    - 10 IOs – BookFlight Service
    - 3 IOs – CancelFlight Service

# What is this programming model? Multiple ECB – Orchestration

- **Solution:** There is NO need to use the begin and end APIs around the calls to each service because each ECB's exit will functionally perform the same purpose. We use the same MsgType and Service (or ECBPurpose) methodology as the single ECB orchestration model. The NVPs are automatically passed to child ECBs.

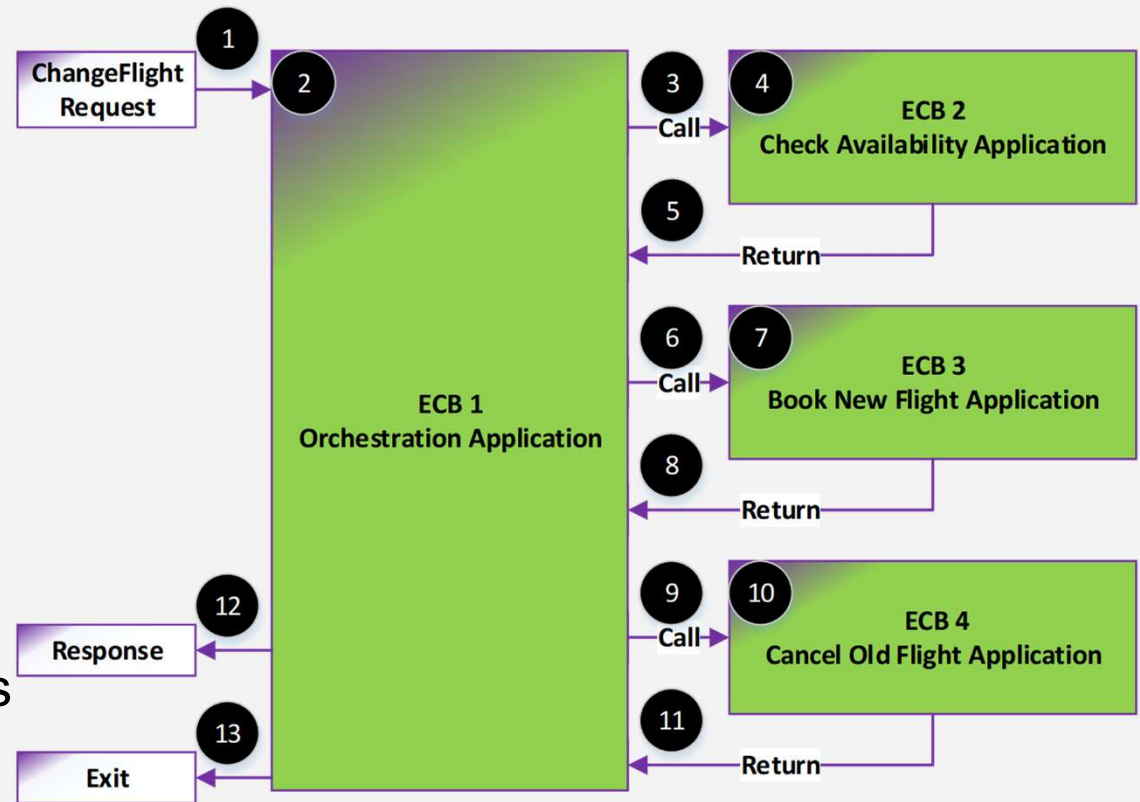


# Model: Single Request - REST



# What is this programming model? REST

- **What:** The REST model is very similar to the multiple ECB orchestration model. An ECB processing a single request calls various existing REST services where each service is performed by a new ECB.
- **Problem:** How do you see the total resources used? How do you break the results down by each service?



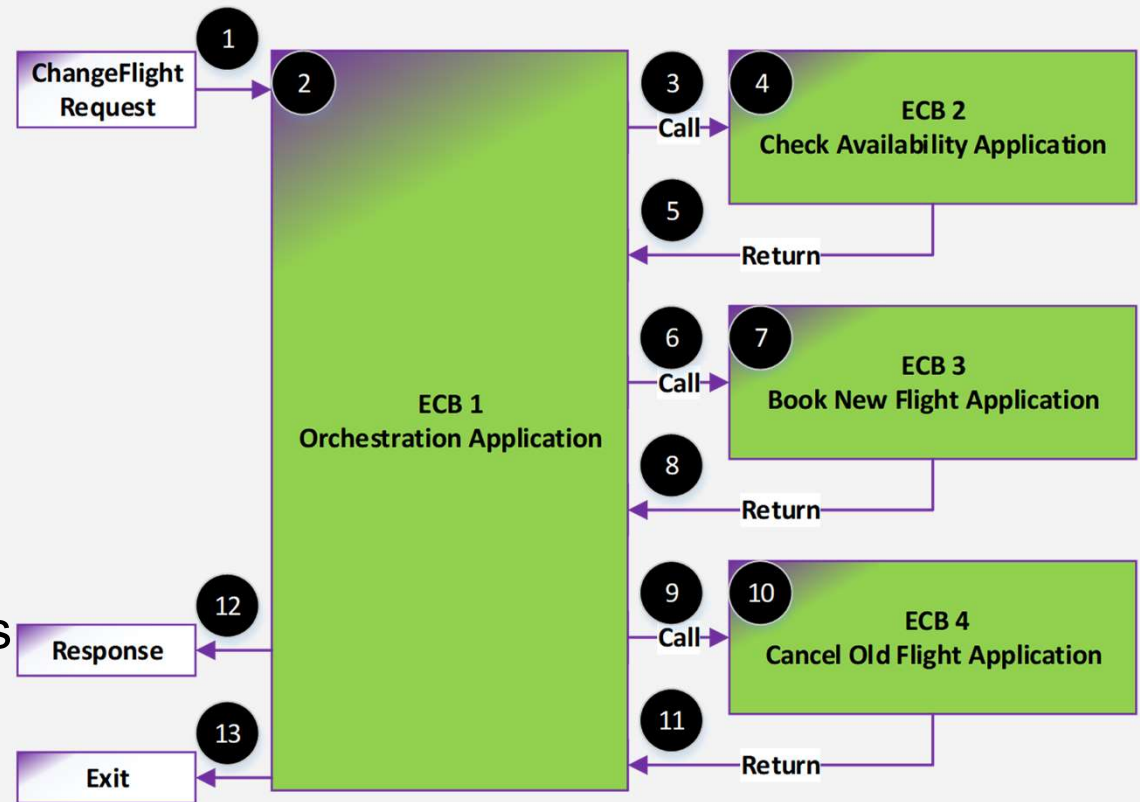
## What is this programming model? REST

- The designs for NVP support for REST are in the early stages. Stay tuned... But here are some initial thoughts...
- If the REST service is called from within the same processor, z/TPF will automatically pass the NVPs.
- The Service Descriptor can be used in conjunction with the DFDL that describes the input structure, to automatically set NVPs. The Service descriptor would provide the name the DFDL xpath. The value would be extracted from the input message.
- Begin/end NVP APIs are not necessary to code since each REST service is a new ECB.
- Set ECB owner names by Program Configuration File like everywhere else.

# Model: Single Request – Java

# What is this programming model? Java

- **What:** The Java model is similar to the multiple ECB orchestration model. A Java application on z/TPF processes a single request possibly calling existing REST services.
- **Problem:** How do you see the total resources used? How do you break the results down by each service?



## What is this programming model? Java

- The designs for NVP support for Java are in the early stages. Stay tuned... But here are some initial thoughts...
- If the Java service is called from within the same processor, z/TPF will automatically pass the NVPs.
- The Service Descriptor can be used in conjunction with the DFDL that describes the input structure, to automatically set NVPs. The Service descriptor would provide the name the DFDL xpath. The value would be extracted from the input message.
- Set/get, begin/end and etc. NVP APIs would probably be implemented in some form. However, JAMs might allow z/TPF to be able to automate a significant portion of these APIs.
- Set owner names by configuration in the JAM Descriptor.

# Model: Single Request – Business Events

## What is this programming model? Business Events

- The designs for NVP support for Business Events is in the early stages. Stay tuned... But here are some initial thoughts...
- z/TPF will automatically pass the NVPs.
- Customer does not code begin/end APIs as these are handled by the IBM Event code.
- Vertical name-value pairs IDispatchAdapter and IEventName are set automatically based upon the event descriptor.
- Set ECB owner names by Program Configuration File like everywhere else.

# Call to Action



## Call to Action

- Start simple. One NVP and ECB Owner Hi set by configuration.
  - Then grow your usage.
- Get involved in our continuing playbacks and provide feedback.

# Thank you

Josh Wisniewski  
z/TPF Development



# 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](http://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.