#SHAREorg

SHARE
Technology · Connections · Results

Automatic Restart Manager (ARM)

Mark A Brooks
IBM
mabrook@us.ibm.com

Tuesday August 7, 2012
Session 11782

SHARE in Anaheim 2012

Trademarks

System z Social Media

- System z official Twitter handle:
  - @ibm_system_z
- Top Facebook pages related to System z:
  - Systemz Mainframe
  - IBM System z on Campus
  - IBM Mainframe Professionals
  - Millennial Mainframer
- Top LinkedIn Groups related to System z:
  - Mainframe Experts Network
  - Mainframe
  - IBM Mainframe
  - System z Advocates
  - Cloud Mainframe Computing
- YouTube
  - IBM System z
- Leading Blogs related to System z:
  - Evangelizing Mainframe (Destination z blog)
  - Mainframe Performance Topics
  - Common Sense
  - Enterprise Class Innovation: System z perspectives
  - Mainframe
  - MainframeZone
  - Smarter Computing Blog
  - Millennial Mainframer

Why Automatic Restart?

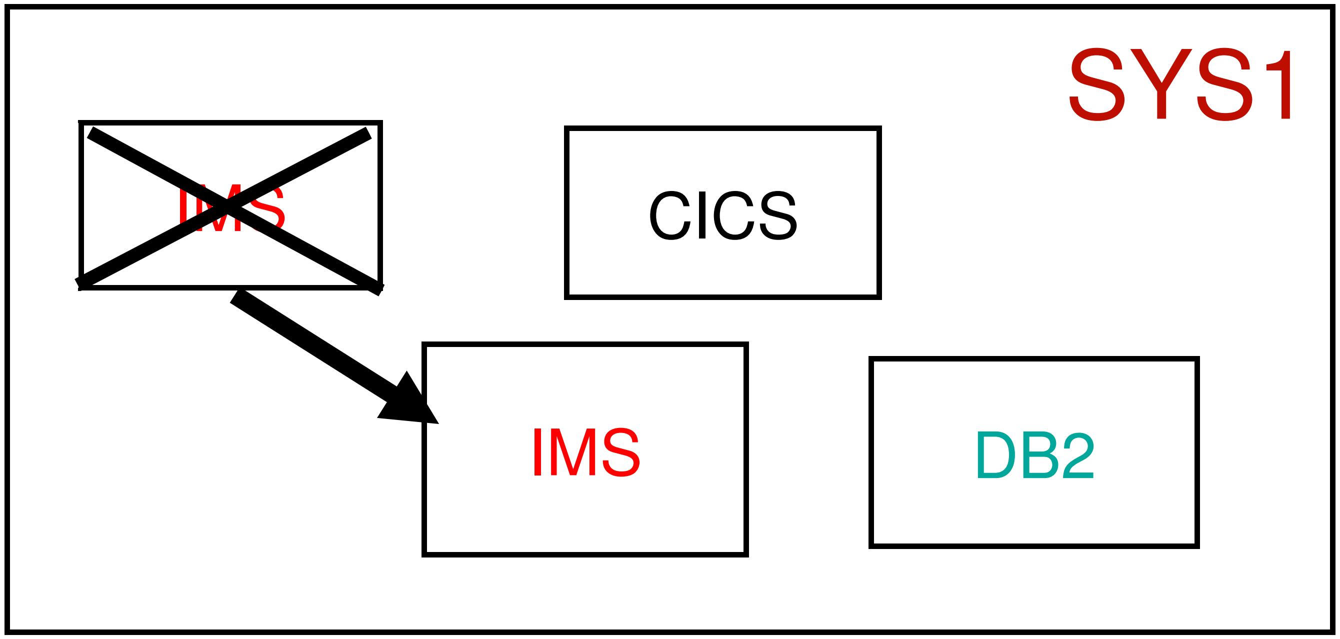
- Sysplex offers the promise of high availability through
  - Redundancy
  - Data sharing with integrity
  - Work load management
- But applications and systems will fail
  - Ideally, the survivors transparently absorb the work
- And when a failure occurs
  - Capacity may be diminished
  - Function may not be available
  - Recovery may be needed
- So restoring the application to service as quickly as possible could be critical to your business

Automatic Restart Manager (ARM)

- Automatically restarts “critical” applications after they fail

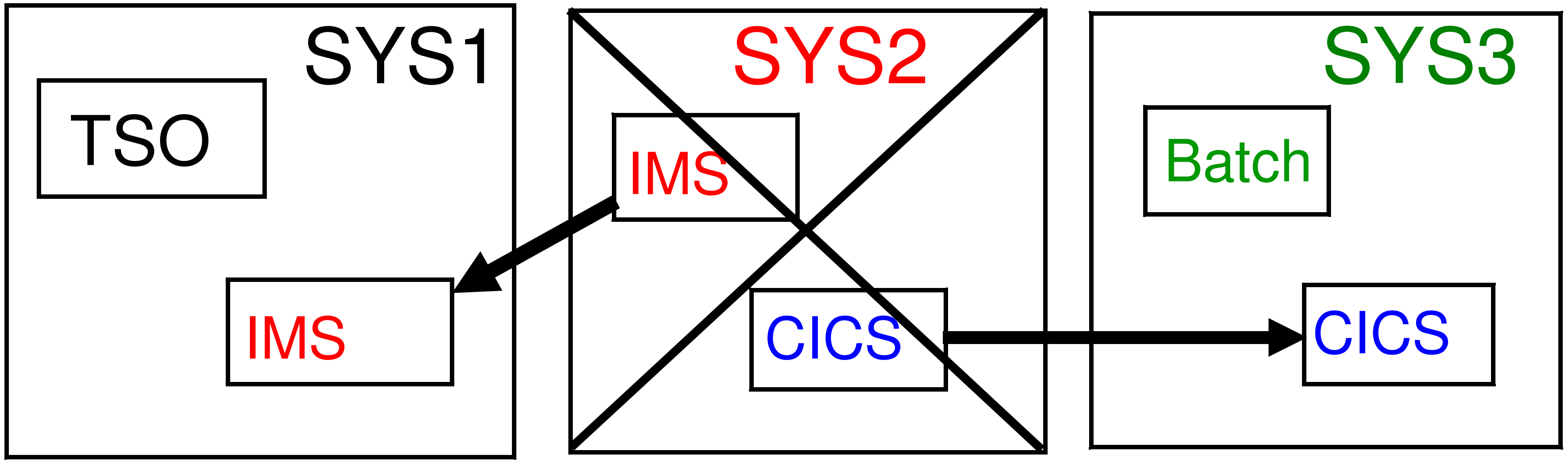
- ARM is an integral part of z/OS
  - Available as part of the system
  - Hooks into resource managers and XCF sysplex partitioning
- ARM is sysplex enabled
  - Can restart applications on other systems in the sysplex
  - Systems chosen with input from Work Load Manager (WLM)
- Most “critical” applications are enabled for ARM
- Integrates with existing automation software

Application failure



ARM restarts the failed application on the same system

System Failure

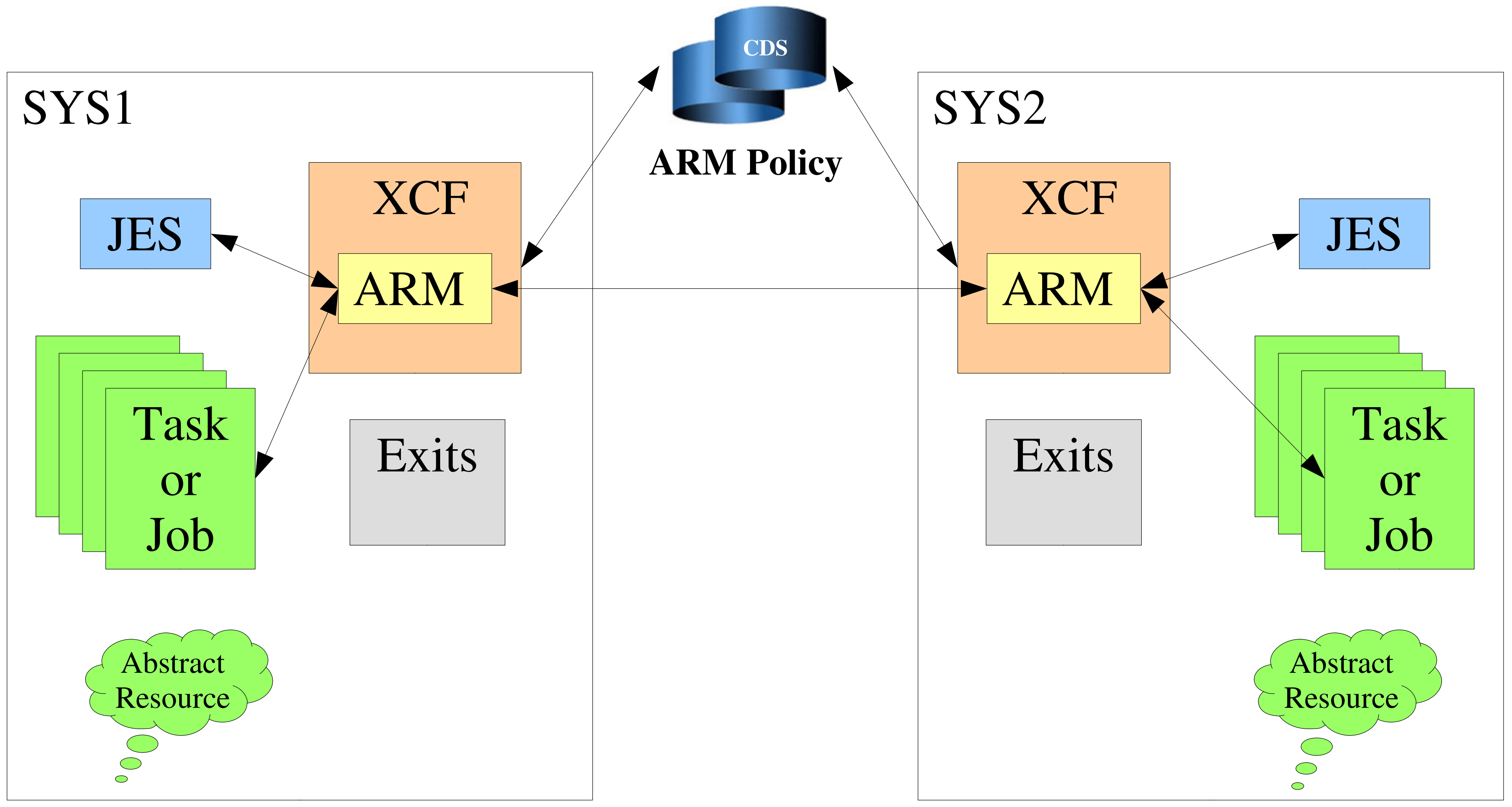


If a system fails, ARM restarts the applications on other systems in the sysplex.

Objectives

- Which applications will ARM deal with?
- Under what conditions will ARM act?
- Where will the applications be restarted?
- How will they be restarted?
- What is the restart process?
- How do I activate ARM?

Our Context


CDS
ARM Policy
SYS1
XCF
ARM
JES
Task or Job
Exits
Abstract Resource
SYS2
XCF
ARM
JES
Exits
Task or Job
Abstract Resource

Applications Must Register with ARM

- Applications must explicitly indicate that ARM is to restart them
  - By invoking a programming interface to **register** with ARM
  - Applications that register with ARM are said to be **ARM Enabled**
- If an application is not ARM Enabled
  - Complain to your vendor
  - Might be able to use the **ARM Wrapper** to overcome the deficiency
- ARM uses the term **element** to refer to an application, or a “piece” of an application, that registers. An element could be:
  - Job
  - Started Task
  - **Abstract Resource** (think “system”)

Many Products are ARM Enabled

- z/OS Communications Server
- CICS Transaction Server for z/OS®
- CICS Transaction Gateway
- WebSphere MQ
- WebSphere® Decision Server for z/OS®
- WebSphere Application Server
- DB2
- IMS™ in some environments:TM-DB, DCCTL, DBCTL, XRF, FDBR
- z/OS MVS Capacity Provisioning
- Tivoli Workload Scheduler
- Tivoli System Automation for z/OS
- Component Broker
- SDSF, RRMS, VTAM, IRLM …
- Products form other software vendors as well

Check product documentation for guidance

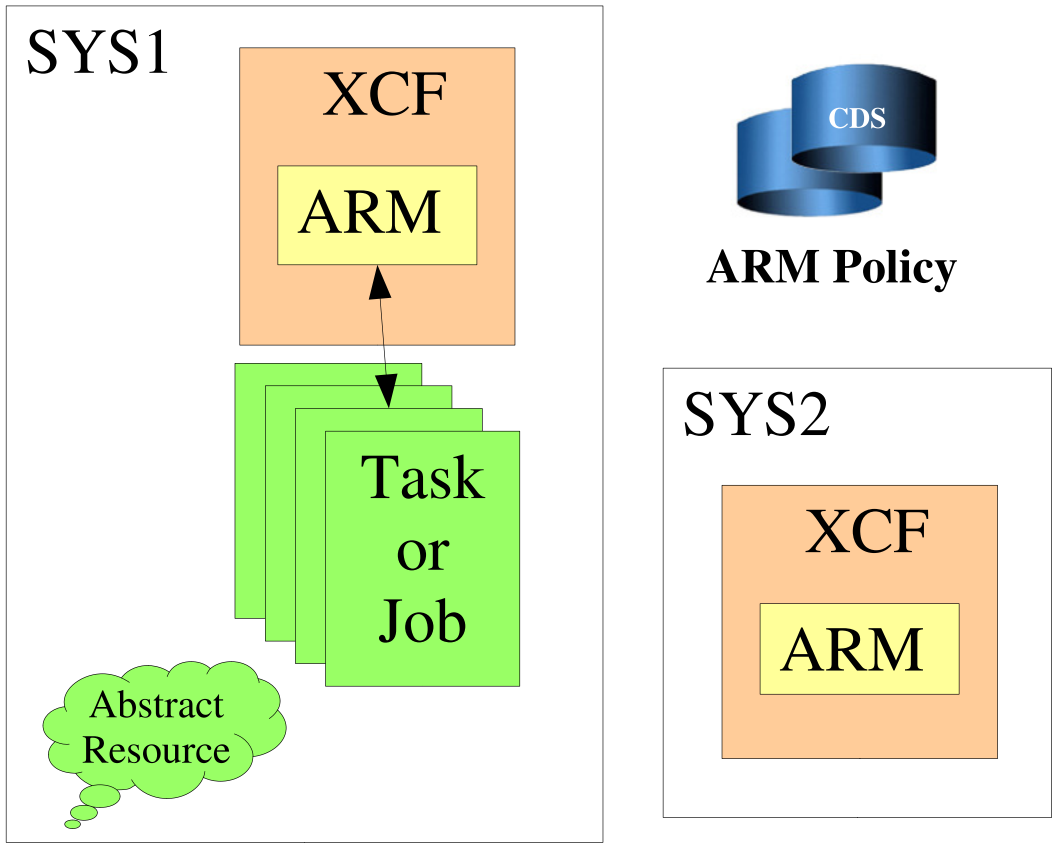
Terminated Element Eligible for Restart ?

<u>Application Specifications</u>

- Element Bind
  - Job/task
  - System
- When to restart me
  - Only if job/task terminates
  - Only if system terminates
  - Per element bind

<u>ARM Policy Specifications</u>

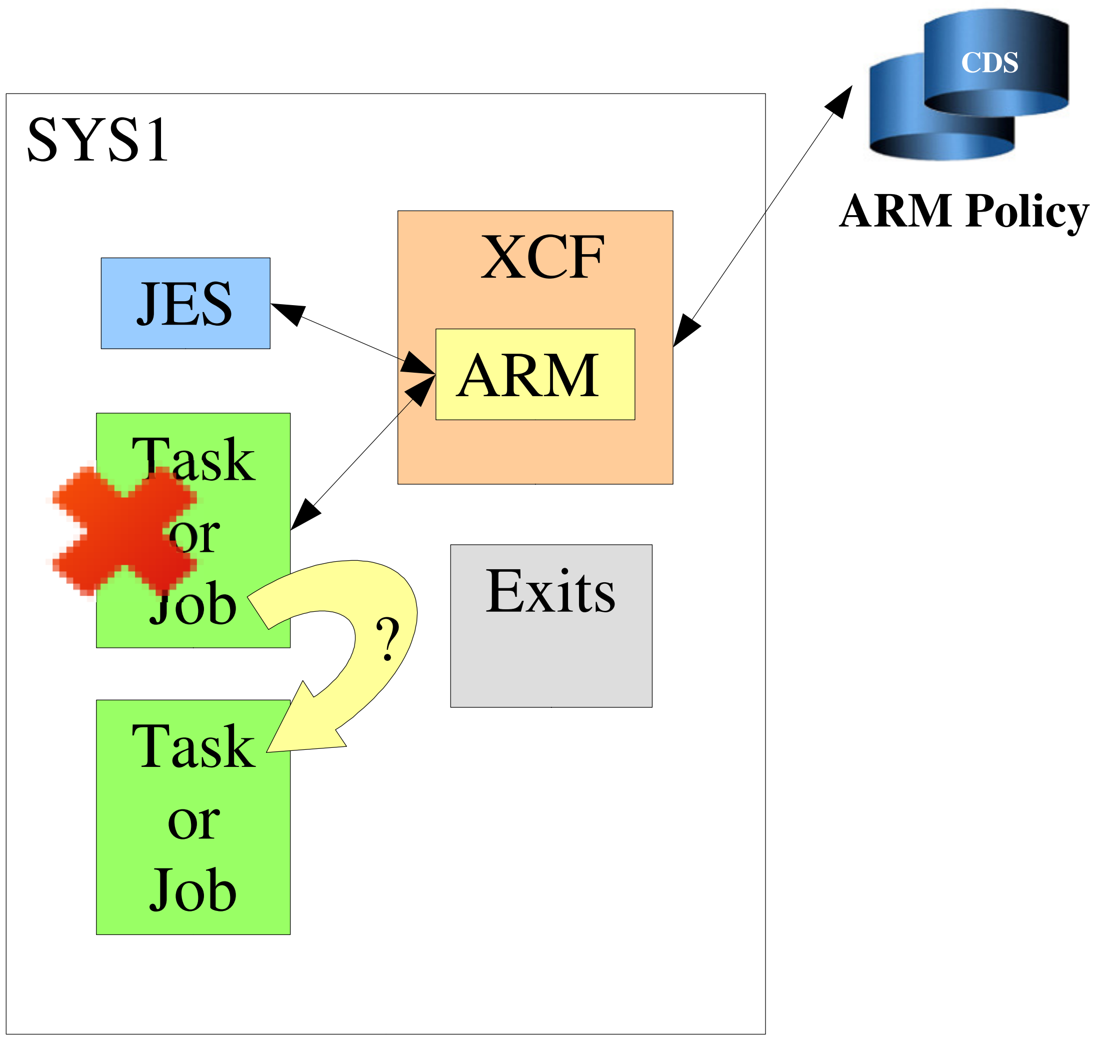
- TERMTYPE(scope)
- Restart only if job/task terminates
- Restart if either job/task or system terminates



Restarting Terminated Application

- Assume ARM detects termination of element
- And determines that the element is to be restarted

- ARM needs to know how to accomplish the restart: (**Restart Method**)
  - What JCL
  - What START command



Restart Method

- By default, ARM restarts the application the same way that it was most recently started. ARM will:
  - Resubmit the same JCL that started the job, or
  - Reissue the same START command that created the task
- ARM refers to this as **persistent restart text**, either:
  - Persistent JCL
  - Persistent command text
- But “start again the same way” might not always be appropriate
  - For the type of termination (element vs system)
  - For the way the application implements its recovery

Alternative Restart Methods

- Application Specifications
  - If a started task registers with ARM while running authorized, it can specify alternative command text to be used when restarting the application
  - A job has no ability to request that different JCL be used for restart
- ARM Policy Specifications
  - RESTART_METHOD(TermType, TextType, RestartText)
  - You might provide alternative restart text
    - Based on type of termination (element or system)
    - Per application guidance
    - For your own reasons
- But these are static choices ...

Dynamic Alternatives for Restart Method

- Might need to determine restart method dynamically. Perhaps it:
  - Must be unique to system where element is restarted
  - Depends on state of the system
    - Batch window, peak transaction period, …

- IXC_ELEM_RESTART installation exit
  - Can dynamically change the restart method (or not)
  - Can also cancel the restart (or not)
  - ARM calls the exit once for each element that is to be restarted on the system, just prior to the restart

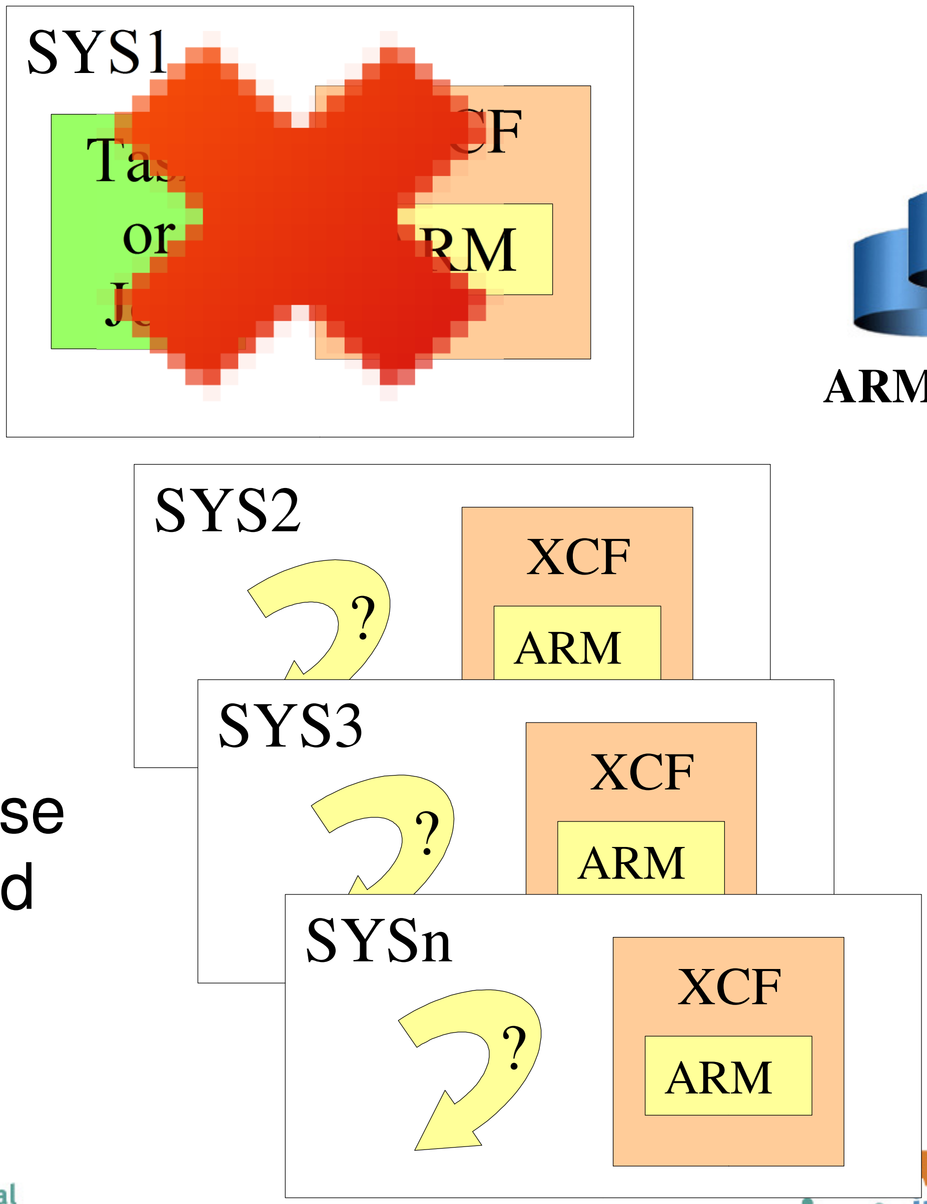
Automation products usually provide this exit to ensure that ARM does not conflict with their actions (cancel restart)

Restarting Abstract Resources

- Abstract resources are not bound to any task or job
  - So they do not have permanent restart text
  - Someone has to provide the appropriate restart text
- Restart text can be provided by:
  - Application when it registers
  - ARM Policy Specification
  - IXC_ELEM_RESTART installation exit
- If no restart text is provided, the element will not be restarted
  - ARM may not discover this until after the installation exit runs

Restarting After System Termination

- Assume ARM detects system termination
- And determines that an element that ran on that system is to be restarted elsewhere
    (**Cross System Restart**)
- ARM still needs to know the restart method
- But ARM also needs to choose a system to host the restarted element



Cross System Restart Concerns

- How might the workloads running on a system be affected if the application got restarted there?
- Does the system have enough CPU capacity to meet the needs of the application?
- Does the system have enough memory?
- Does the system has access to the resources needed to run the application (DASD, Coupling Facilities, …)
- Does the system have the class of initiator required by batch jobs that might need to be restarted there

- What about symbolic substitution in the restart text? How is that resolved?

Candidate Systems

- ARM Policy Specifications
  - TARGET_SYSTEM(sysname, ...)
    - Set of systems to be considered as new host for restarted element
      - *A "set", not a "preference list"*
    - Default is all active systems in the sysplex
  - FREE_CSA(csa,ecsa)
    - Indicates amount of CSA and ECSA that must be available on the system selected to host the restarted element
    - Expressed in units of KB
    - 0 is default, implies no need to consider

ARM System Selection

- Start with TARGET_LIST from the ARM Policy
- Discard systems that:
  - Are not connected to the ARM Couple Data Set
  - Are not in the same JES2 MAS or JES3 Complex
  - Lack sufficient CSA/ECSA
- From the systems that remain, choose one based on:
  - CPU Capacity
  - “Scattering” techniques

Restart After Element Terminates

ARM will:

- Select the current host system to be the new host
- Give control to the IXC_ELEM_RESTART installation exit to:
  - Optionally change the restart method
  - Optionally cancel the restart
- Gives control to the application event exit to:
  - Optionally prepare for restart of the element
  - Optionally cancel the restart
- Restart the element using the resultant restart method
- Issue an ENF signal when the element re-registers with ARM

Restart after System Termination

ARM on one system will:

- Determine which elements need to be restarted
- Select a system to be the new host for the element

ARM on every selected system will then:

- Give control to the IXC_WORK_RESTART installation exit
- For each element that is to be restarted
  - Give control to the IXC_ELEM_RESTART installation exit
  - Give control to the application event exit
  - Restart the element using the resultant restart method
  - Issue an ENF signal when the element re-registers ARM

IXC_WORK_RESTART Installation Exit

- Gets control before ARM restarts elements on the system
- ARM provides
  - Information about the system
  - Information about the elements that are to be restarted
- The exit is expected to perform any work needed to prepare the system to handle the elements that are to be restarted
  - For example, cancel lower priority work to ensure that the restarted applications will have sufficient system resources
- This exit cannot cancel the restart or modify the restart method
  - Use IXC_ELEM_RESTART exit if you need to do that

We Actually Skipped A Step

- When a system terminates, it may be holding resources
- Surviving instances may need to perform cleanup to reclaim or release those resources
  - ENQ's, command prefixes, etc
- ARM should not restart elements on a surviving system until such cleanup is complete.
  - The element might fail to restart if it is unable to obtain a resource because it still appears to be held by the failed system
- So how does ARM know when it is safe to restart the element?
  - Which resources are relevant?
  - Are they held by the dead system?
  - Has the cleanup completed?

Special Case: XCF SysCleanupMem

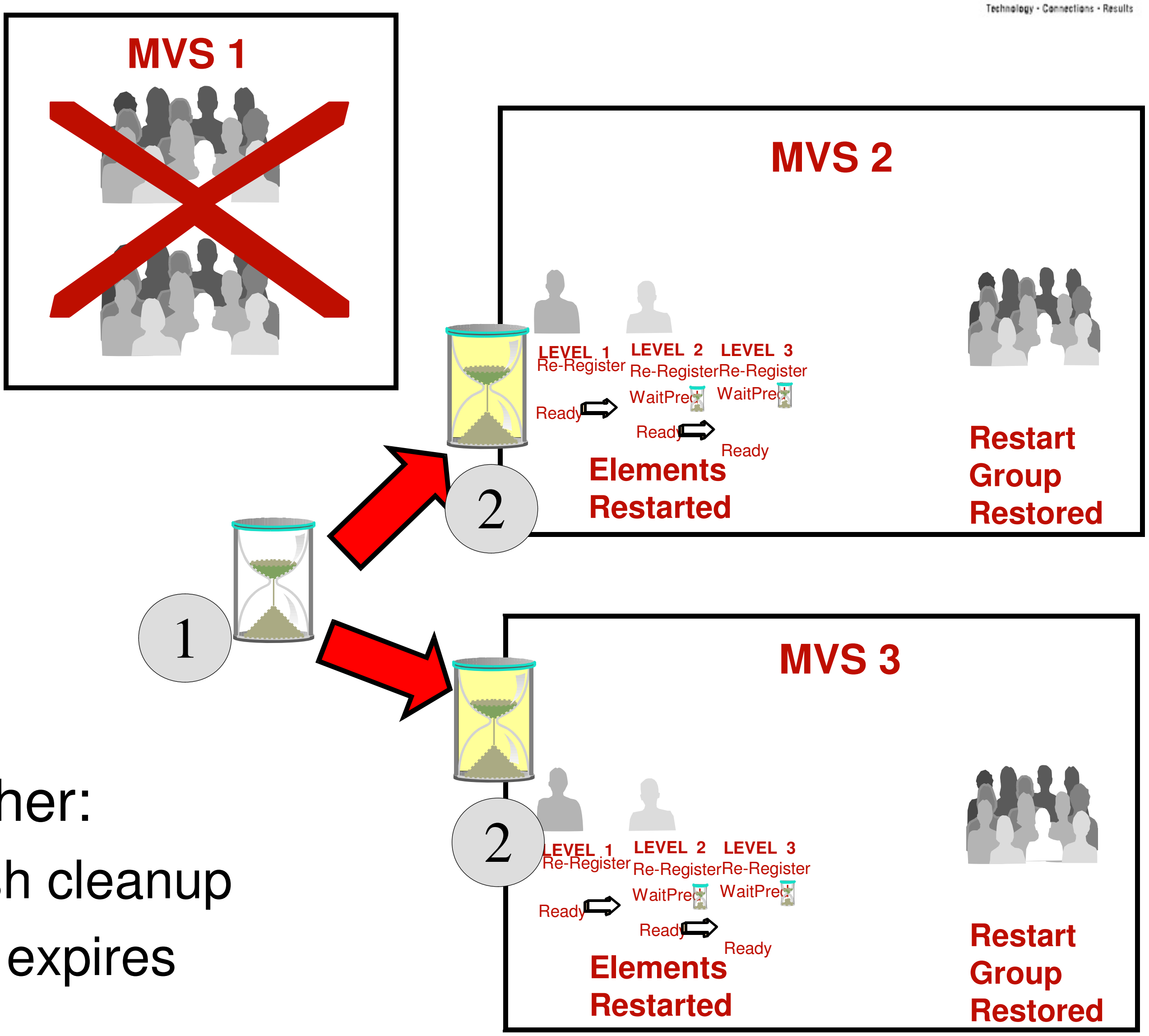
- When a member joins its XCF group, it can indicate to XCF that it needs to perform “system cleanup” when a peer system terminates
- When a peer system terminates:
  - XCF tells member that system was removed from sysplex
  - The member performs whatever system cleanup is needed
  - When cleanup is complete, the member tells XCF that it is done
  - When all such members have “signed off”, XCF tells ARM that cleanup by the local system on behalf of the failed system is done

Waiting for System Cleanup Members

1 ARM begins restarts simultaneously on all systems when all members on every system finish cleanup within 2 minutes

2 After 2 minutes, each system will individually begin restarts when either:

- All local members finish cleanup
- CLEANUP_TIMEOUT expires



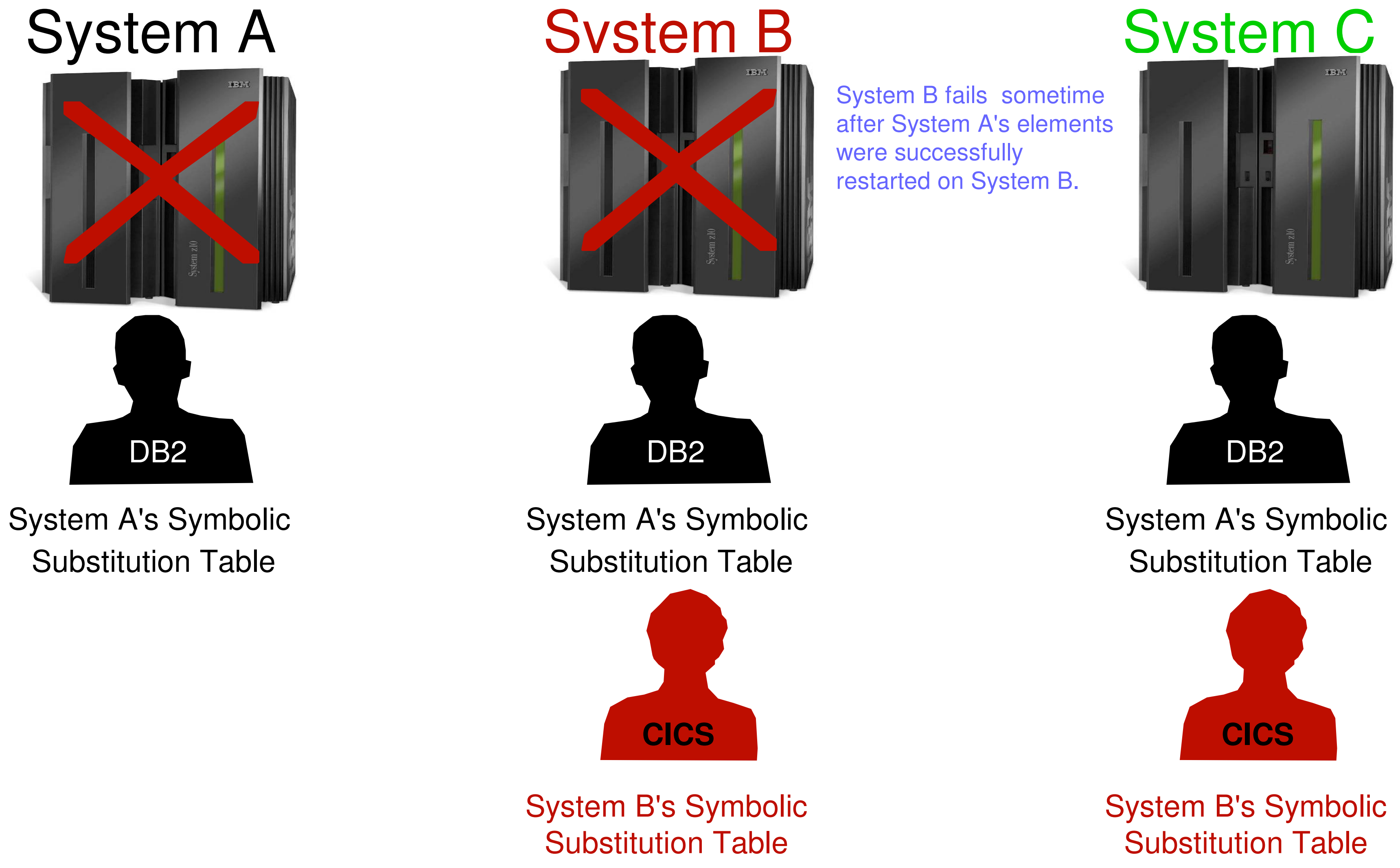
ARM Policy CLEANUP_TIMEOUT

- Maximum time to wait for XCF “system cleanup” group members to finish their cleanup for a failed system
- If too short
  - Group members may not have enough time to finish cleanup
  - Restart of elements could then fail
    - Might not be any that care
- If too long
  - MTTR elongated, perhaps needlessly
    - Long running resource cleanup may not be relevant to element
    - Sympathy sickness impact induced by member that forgets or fails to tell XCF that cleanup was completed

Symbolic Substitution

- Can be used in restart text
  - Restart JCL
  - Restart Command
- When an element is restarted on some other system, whose symbolic substitution tables should be used?
  - Original system where element started?
  - Local system where being restarted?
- ARM uses the original so that substitutions in things like data set names get resolved the same way
  - Care may be needed since some variables from the restart system's substitution table will not be available to the element

ARM Moves the System Symbolic Substitution Table with the Element


System A
System B
System C
DB2
System A's Symbolic Substitution Table
DB2
System A's Symbolic Substitution Table
CICS
System B's Symbolic Substitution Table
DB2
System A's Symbolic Substitution Table
CICS
System B's Symbolic Substitution Table

System B fails sometime after System A's elements were successfully restarted on System B.

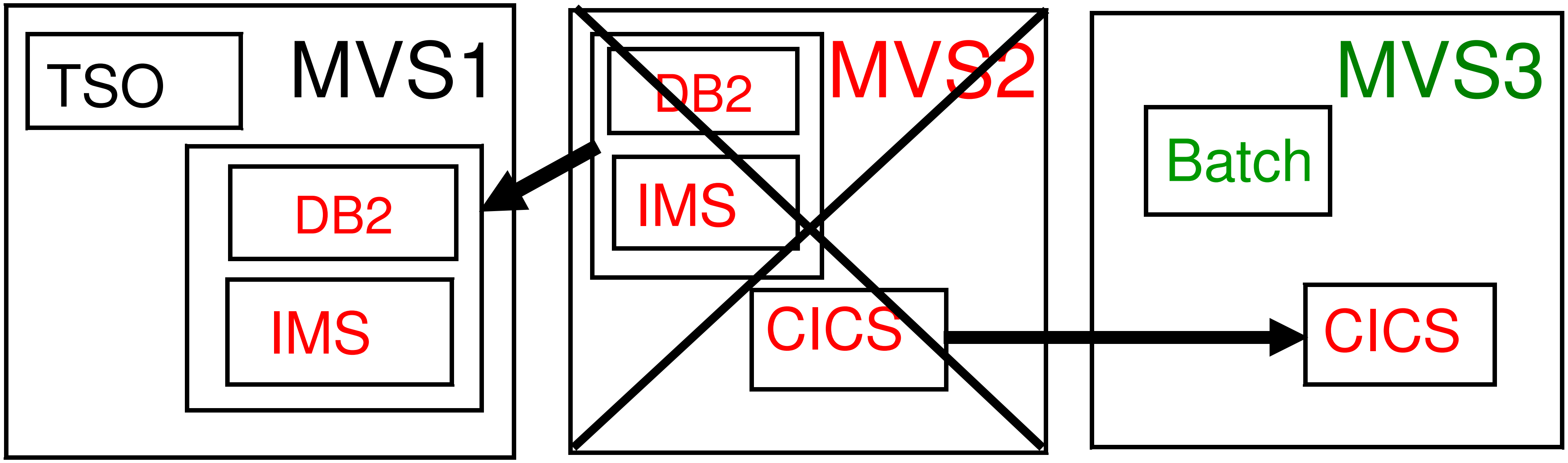
Pacing Restarts after System Termination

- Restarting all the elements at the same time could cause a spike on the new host system
- Pacing will stagger the restarts which might help lower the spike
- But might also reduce parallelism, which could increase MTTR

- ARM Policy Specifications
  - RESTART_PACING(delay)
  - Number of seconds to wait before starting the next element in the **restart group**
  - Only applies when performing restarts after system termination

Restart Groups

Elements may be grouped so that they will be restarted on the same target system



Why Group Elements?

- The elements are related somehow
  - Cooperate to provide some function or service
  - Might have dependencies on each other
- May not function unless the elements reside on same system
- May have performance issues if separated
- You simply want them co-located

- ARM Policy Specification
  - RESTART_GROUP(name of group)
    - ELEMENT(element name)
    - ELEMENT(element name)
    - etc

Restart Order

- The order in which elements are restarted may be important
  - Element 2 might need services provided by Element 1
  - If Element 1 is not restarted first, the restart of Element 2 could fail

- ARM Policy Specifications
  - RESTART_ORDER
    - LEVEL(sequence)
      - ELEMENT_NAME(list of element names)
      - ELEMENT_TYPE(list of element types)
    - LEVEL(sequence)
      - ELEMENT_NAME(list of element names)
      - ELEMENT_TYPE(list of element types)

Why Element Types ?

- We might need IRLM to restart before DB2 because DB2 depends on IRLM services
- But IRLM could have many, many elements and they might change over time with new releases or maintenance
- It would be hard to keep up with all those element names that represent the pieces of IRLM that need to be restarted before DB2 can be restarted
- So we define an element type (or category) so that all the IRLM elements can indicate that they are of type IRLM
- ARM can then ensure that all elements whose type is IRLM are restarted before any elements whose type is DB2 are restarted

Element Type

- Specified by application when registers with ARM
- Facilitates assignment of a “level” that is used to determine the order in which elements are restarted

- By default, ARM has several predefined (default) element types which are used by the various ARM Enabled products

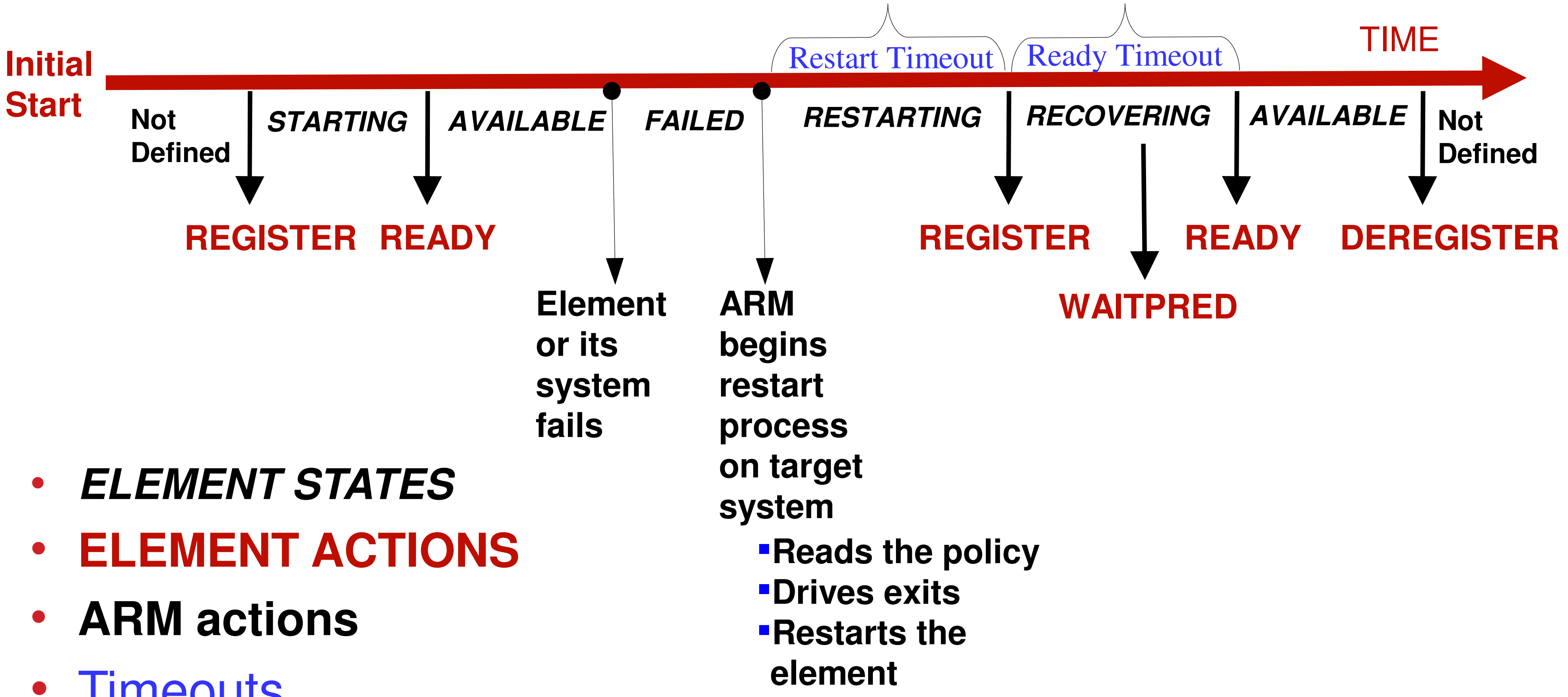
How to manage all those relationships?

- In general, you don't have to know inter-element relationships
- Many are appropriately pre-defined via ARM element types

- In particular, it is really up to the application to identify them
- The application should deal with them in software rather than forcing every installation to define restart order via ARM policy
- The inter-element dependencies could be more complicated than what can be expressed in policy
  - For example, elements whose initialization needs to be interleaved

So we look at how elements interact with ARM

Life of an ARM Element



- *ELEMENT STATES*
- ELEMENT ACTIONS
- ARM actions
- Timeouts

ARM Policy Timeout Specifications

- RESTART_TIMEOUT = maximum time to wait for a restarted element to re-register with ARM. If it expires:
  - Element is deregistered
  - Message IXC803I “deregistered element due to timeout”
  - Waiting elements are released
    - They may or may not fail depending on the actual state of the element that timed out
- READY_TIMEOUT = after restarted element registers, maximum time to wait for it to indicate that it is ready. If it expires:
  - Element is placed into “available due to timeout” state
  - Waiting elements are released
    - They may or may not fail depending on the actual state of the element that timed out

Handling Dependencies within a Restart Group

	DB2	CICS	Installation Application
	LEVEL 1	LEVEL 2	LEVEL 3
1.	Re-Register	Re-Register	Re-Register
2.		WaitPred	Ready
3.	Ready	Can use level 1's services to complete initialization	▪Implicit WaitPred
4.	Services are available	Ready	Services are available Can use level 1's and 2's services now
5.		Services are available	

TIME

TIME

Unconditionally Restart Element ?

- So ARM sees an application terminate. Based on the application specifications and the ARM policy, it can determine the method by which the application is to be restarted, and it can find some system to host the restarted element.
- But …
  - Should ARM *always* restart the application?
  - Might there be conditions for which we would not want to restart it?

Some Reasons Not To Restart

- Application is suffering repeated failures
- Application is being tested
- Application is intentionally cancelled
- Application is ARM Enabled, but you don't want it restarted

Influencing Restart Decision

- ARM Policy Specifications
  - RESTART_ATTEMPTS(limit, interval)
    - Max number of restart attempts allowed within given interval
  - RESTART_ATTEMPTS(0)
    - Do not restart the element

- CANCEL entity
- FORCE entity

ARM will not restart

- CANCEL entity,ARMRESTART
- FORCE entity,ARMRESTART

ARM allowed to restart

Security Considerations XCFAS needs trusted attribute

- ARM issues commands and submits JCL while running in the XCF address space (XCFAS)
- So XCF needs to have sufficient authority to be issue whatever commands are needed to restart the element. Either:
  - Put XCFAS in the started procedures table with the trusted attribute
    - *I believe this is typical for many installations*
  - Or define XCFAS in the RACF STARTED class
    - *or the equivalent for other security products*

Security Considerations: Users May Need SAF Authorization

- ARM supports both authorized and unauthorized users
- Unauthorized users:
  - Cannot change the restart method
  - Depending on whether the application specifies ELEMTYPE when registering with ARM, require UPDATE access to the one of the following profiles in the FACILITY class, either:
    - IXCARM.elemtype.elemname
    - IXCARM.DEFAULT.elemename

See *MVS Programming: Sysplex Services Reference, Chapter 4: Request Automatic Restart Management Services (IXCARM)*

Security Considerations: Security Environment For Restarts

Persistent JCL

Override JCL

- Set security environment to that of last registered TCB
- Open override JCL dataset
- Open an internal reader
- Submit the JCL
- Close dataset, internal reader, and env back to XCF

ARM

Exactly the same JCL/START text

- Security environment is exactly the same

Override JCL

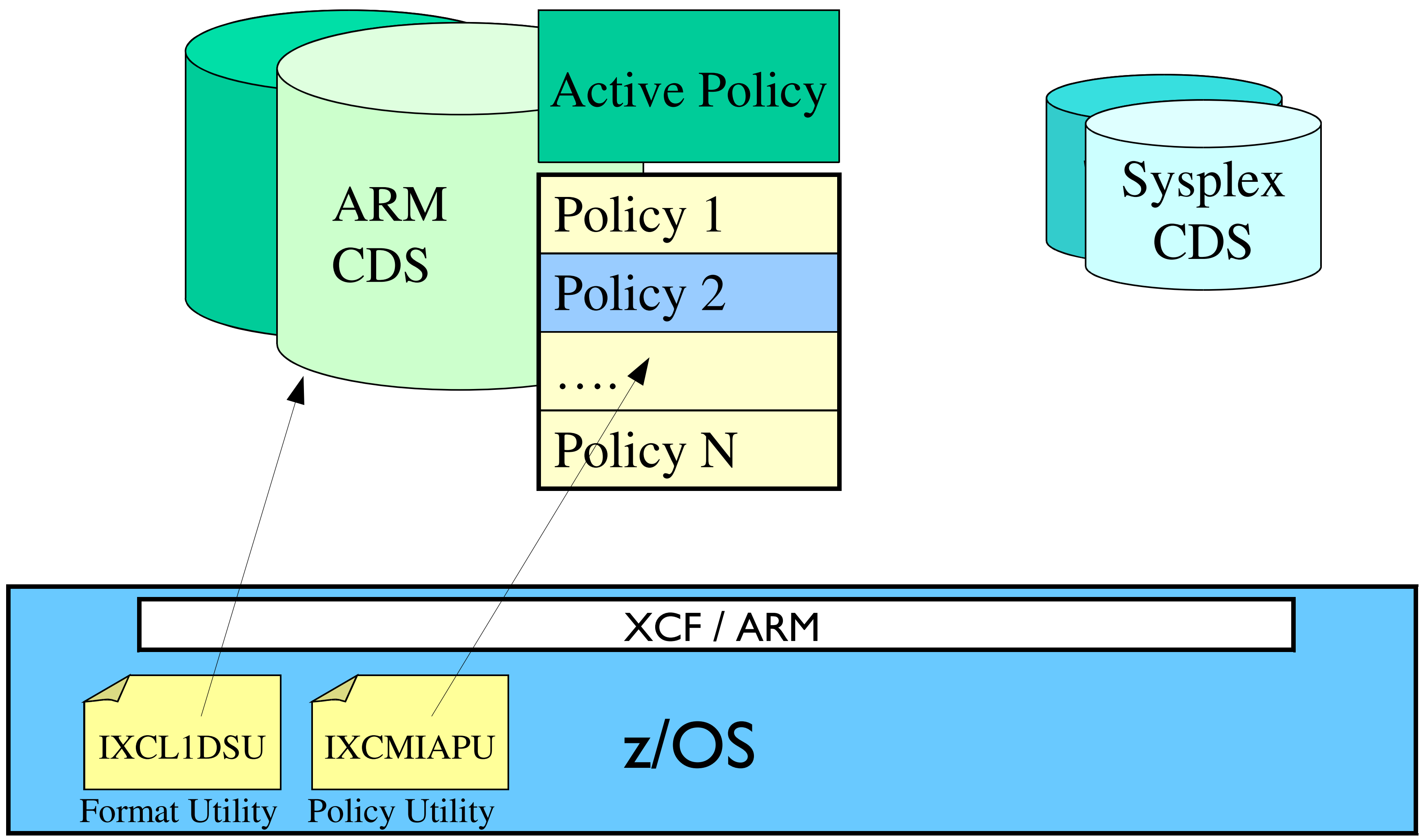
- Security environment is propagated except for:
  - –POE will be INTRDR
  - –If USER=id NOT in JCL then the userid is propagated but the groupid is not.
  - –If GROUP=id NOT in JCL then the default group for the userid is used

Watch out for CICS regions which have PROPCNTL userid restrictions

JES

See *MVS Programming: Sysplex Services Guide, Chapter 4: Using the Automatic Restart Management Function of XCF*

ARM Policy Componentry



Couple Data Set Format Utility

- The format utility program IXCL1DSU creates and formats a Couple Data Set (CDS) using parameters that you specify
- These parameters determine type and number of records that need to be created in the CDS
- For ARM:
  - POLICY – number of ARM policies the CDS needs to hold
  - MAXELEM – maximum number of elements to be defined in a single policy
  - TOTLELEM – at run time when a policy is in use, the maximum number of elements that will register with ARM at one time
- See SYS1.SAMPLIB(IXCARMF)

Operations: ARM Couple Data Sets

Get sysplex to use an ARM CDS

- SETXCF COUPLE,TYPE=ARM,PCOUPLE=dsn
- SETXCF COUPLE,TYPE=ARM,ACOUPLE=dsn
  - Start using a previously formatted ARM CDS as the alternate CDS

Which ones are in use by the sysplex?

- D XCF,COUPLE,TYPE=ARM
  - Show information about the ARM Couple Data Set

Promote the alternate to be primary

- SETXCF COUPLE,TYPE=ARM,PSWITCH

Administrative Policy Utility

- The policy utility program IXCL1APU is used to manipulate administrative policies in a formatted ARM Couple Data Set
- The parameters you specify define the policy, which in turn describes how ARM is to behave
- The policy utility can
  - Create new policies
  - Replace a existing policies
  - Delete a policies
  - Report on the policies currently defined in the CDS
- See SYS1.SAMPLIB(IXCARMP0)

ARM Policy Statements Control:

- Whether restarts will occur for some, none, or all elements
- The method by which an element is to be restarted
- Grouping and dependencies among elements
- Which systems are candidates for hosting a restarted element
- Restart intervals, thresholds, and time out values

- Note: the policy that governs the restart is the policy that is active when ARM processes the restart, not the policy that was active when the element registered with ARM

ARM Policy Example

```
DATA  TYPE(ARM)
REPORT(YES)
DELETE POLICY NAME(ARMPOL00)
DEFINE POLICY NAME(ARMPOL01)
REPLACE(YES)
RESTART_ORDER
    LEVEL(3)
         ELEMENT_NAME(INSTAPP*)
RESTART_GROUP(*)
   TARGET_SYSTEM(SYS1,SYS2,SYS3)
RESTART_GROUP(DEFAULT)
    ELEMENT(*)
        RESTART_ATTEMPTS(0)
```

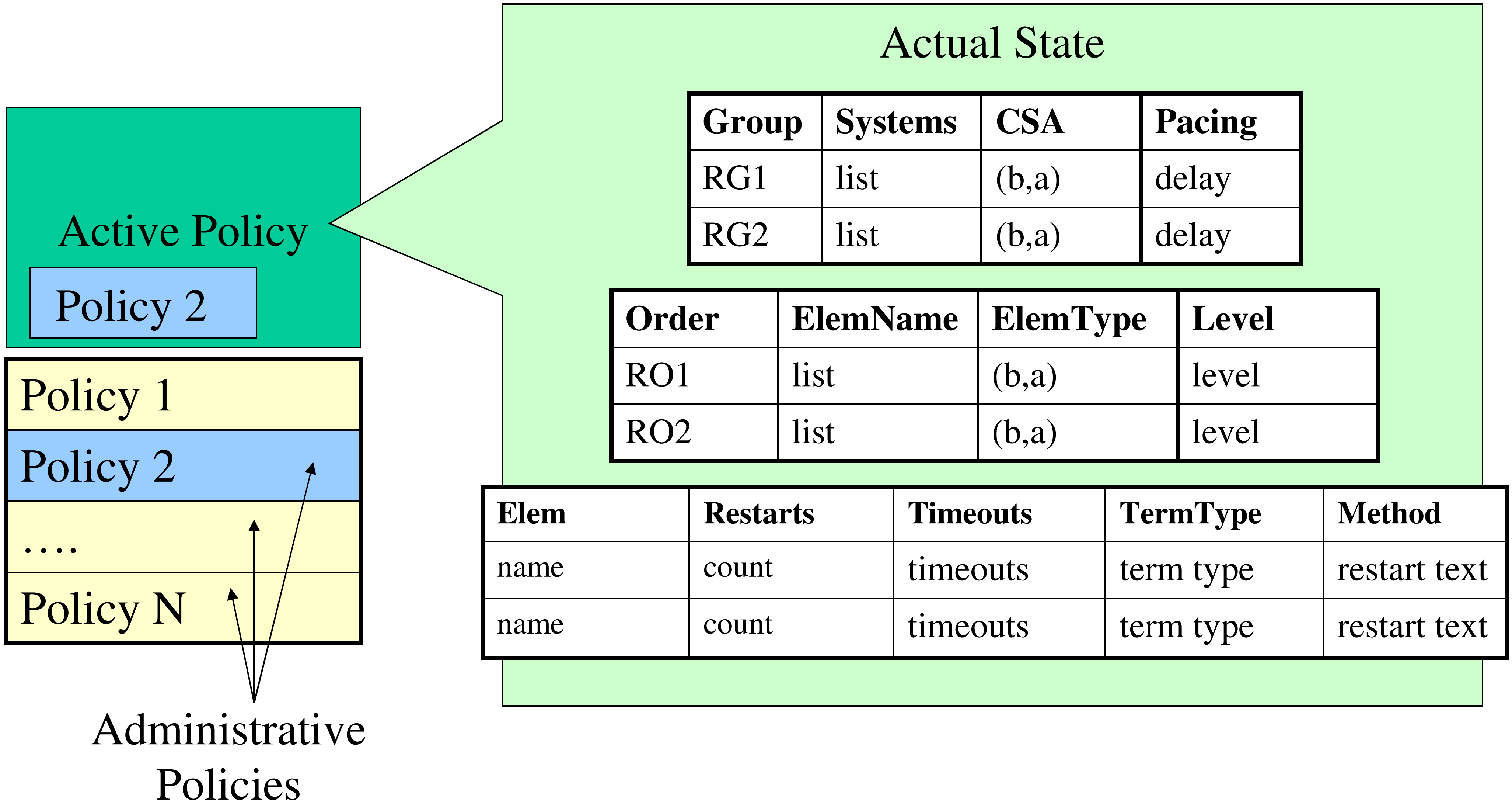
ARM Policy Example Continued

```
RESTART_GROUP(GROUP1)
  TARGET_SYSTEM(SYS1,SYS2)   /* Eligible systems                      */
   FREE_CSA(100,500)          /* system must have 100k of CSA and
                                500K of ECSA if in WLM goal mode*/
   ELEMENT(DSNDB1GDB81)       /* DB2 Data sharing element              */
   ELEMENT(SYSCICS_CIC3A8)    /* CICS AOR                              */
   ELEMENT(INSTAPP1)          /* Your installation application         */
```

ARM Policy Example Continued

```
RESTART_GROUP(GROUP2)
  ELEMENT(IRLMGRP2IRL8002)    /* IRLM element                                    */
  ELEMENT(IMS8)
    RESTART_ATTEMPTS(3,1200)  /* If the element fails 3 times within 20 minutes
                                 then do not restart it again.                   */
    RESTART_TIMEOUT(600)      /* If the element does not re-register within 10
                                 minutes then deregister it.                     */
    READY_TIMEOUT(600)        /* If the element does not finish initialization
                                 within another 10 minutes, then assume that it
                                  is initialized after that point.              /
    RESTART_METHOD(ELEMTERM,PERSIST)
    RESTART_METHOD(SYSTERM,STC,
                   'S IMSR6.IMS8,IMAGE=8,'
                   'APPL=IMSPETJ8,IRLMNM=IRL8')  /*  STC for systerm             */
  ELEMENT(SYSCICS*)
```

ARM Policies



Group	Systems	CSA	Pacing
RG1	list	(b,a)	delay
RG2	list	(b,a)	delay

Order	ElemName	ElemType	Level
RO1	list	(b,a)	level
RO2	list	(b,a)	level

Elem	Restarts	Timeouts	TermType	Method
name	count	timeouts	term type	restart text
name	count	timeouts	term type	restart text

Operations: ARM Policy

Activate a policy

- SETXCF START,POLICY,TYPE=ARM,POLNAME=xxx
- SETXCF START,POLICY,TYPE=ARM

Which policy is active?

- D XCF,POLICY,TYPE=ARM

Stop using a policy

- SETXCF STOP,POLICY,TYPE=ARM

Operations: Elements

- D XCF,ARMSTATUS
  - Show status of ARM and Elements
  - Summary and detail forms
  - Wildcards supported for element names

- CANCEL entity,ARMRESTART
- FORCE   entity,ARMRESTART
  - After cancel/force of a given job, user, or space, ARM is to restart any elements that may have been bound to that entity

D XCF,ARMSTATUS

```
D XCF,ARMS,DETAIL
 IXC392I  06.03.29  DISPLAY XCF 134
 ARM RESTARTS ARE ENABLED
 -------------- ELEMENT STATE SUMMARY --------------   -TOTAL-  -MAX-
 STARTING  AVAILABLE  FAILED  RESTARTING  RECOVERING
        0          2       0            0           0        2     50
 RESTART GROUP:DB2                PACING   :    0    FREECSA:    0        0
 ELEMENT NAME :DB2$DB2X           JOBNAME :DB2XMSTR STATE   :AVAILABLE
   CURR SYS :SYS1                 JOBTYPE :STC       ASID    :0074
   INIT SYS :SYS1                 JESGROUP:XCFSYS1  TERMTYPE:ELEMTERM
   EVENTEXIT:DSN7GARM             ELEMTYPE:SYSDB2   LEVEL   :    1
   TOTAL RESTARTS :         0     INITIAL START:09/02/2011 06:02:22
   RESTART THRESH :    0 OF 1     FIRST RESTART:*NONE*
   RESTART TIMEOUT:     21600     LAST  RESTART:*NONE*
 RESTART GROUP:DEFAULT            PACING   :    0    FREECSA:    0        0
 ELEMENT NAME :DBXL001            JOBNAME :DB2XIRLM STATE   :AVAILABLE
   CURR SYS :SYS1                 JOBTYPE :STC       ASID    :0075
   INIT SYS :SYS1                 JESGROUP:XCFSYS1  TERMTYPE:ALLTERM
   EVENTEXIT:DXRRL0F1             ELEMTYPE:SYSIRLM  LEVEL   :    0
   TOTAL RESTARTS :         0     INITIAL START:09/02/2011 06:02:22
   RESTART THRESH :    0 OF 0     FIRST RESTART:*NONE*
   RESTART TIMEOUT:       300     LAST  RESTART:*NONE*
```

Summary

- ARM improves availability and MTTR by automatically restarting terminated applications
  - Without operator intervention or looking at messages
  - Integrated into the system for fast failure detection
  - Restarts elements with maximum parallelism
- ARM can restart failed work on other systems in the sysplex
- Applications must be ARM Enabled to be eligible for restart
- The ARM Policy controls how ARM behaves
- Automation packages can be integrated with ARM

For More Information

z/OS Publications

- MVS Setting Up A Sysplex
- MVS Programming: Sysplex Services Guide
- MVS Programming: Sysplex Services Reference
- MVS System Commands
- MVS Installation Exits

All available at www.ibm.com/systems/z/os/zos/bkserv/r13pdf/

For More Information ...

Redbooks

- ABCs of z/OS System Programming: Volume 5
  - Chapter 6 covers ARM
  - www.redbooks.ibm.com/redbooks/pdfs/sg246985.pdf
- z/OS Automatic Restart Manager
  - Describes ARMWRAP which can be used to insert JCL that allows ARM to restart an application that is not ARM Enabled
  - www.redbooks.ibm.com/abstracts/redp0173.html