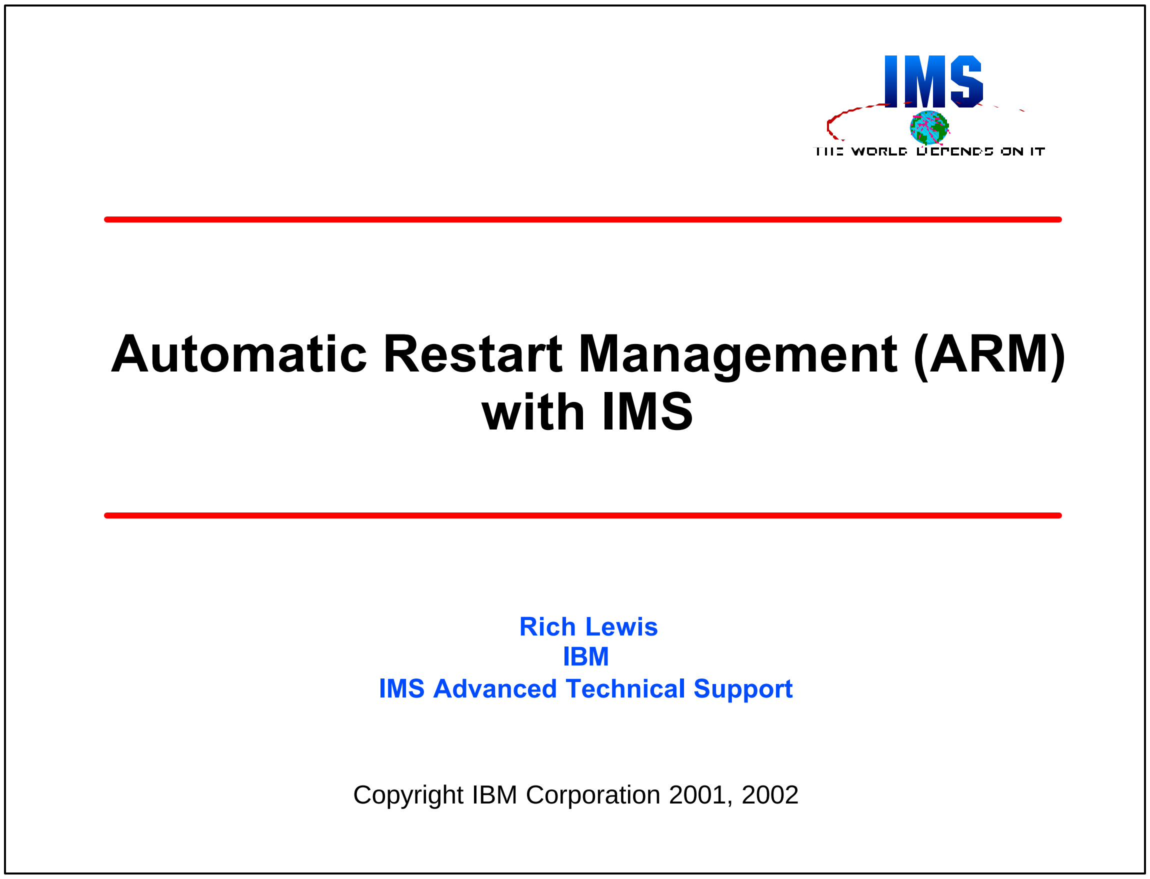


# Automatic Restart Management (ARM) with IMS

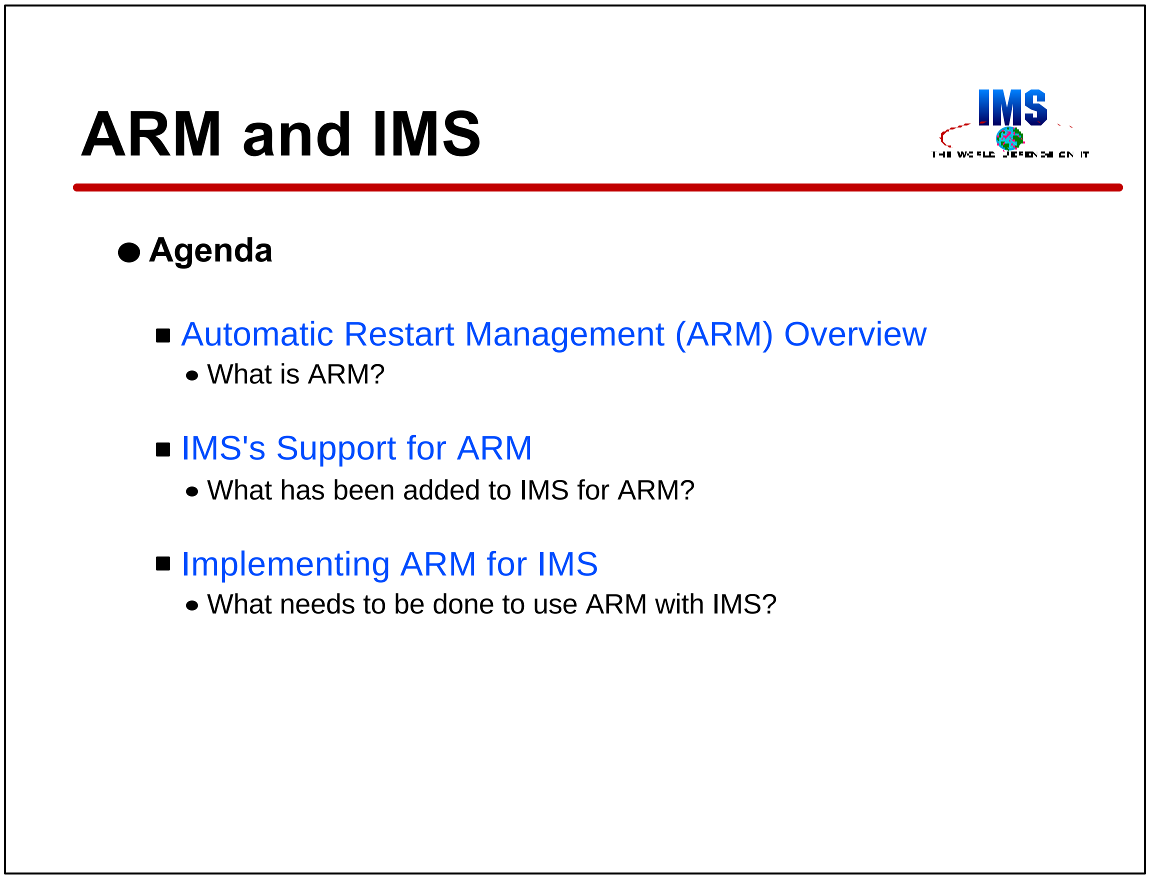
**Rich Lewis**
**IBM**
**IMS Advanced Technical Support**

This presentation was developed by Rich Lewis of the IBM Dallas Systems Center. Its contents are described in the abstract on the next page.

## Abstract and Trademarks

**Abstract**

Automatic Restart Management (ARM) is a function of OS/390. It may be used to automatically restart MVS jobs and started tasks. IMS/ESA 5.1 and later releases include ARM support. If IMS abends, ARM can restart it without any operator intervention. If MVS fails, ARM can restart IMS on another MVS system in the sysplex. ARM can also restart any CICS and DB2 systems used with IMS. This presentation includes an overview of ARM, an explanation of IMS's support for it, and recommendations for implementing ARM with IMS systems. It explains how to increase the availability of IMS systems using data sharing, shared queues, DBCTL, or DB2.

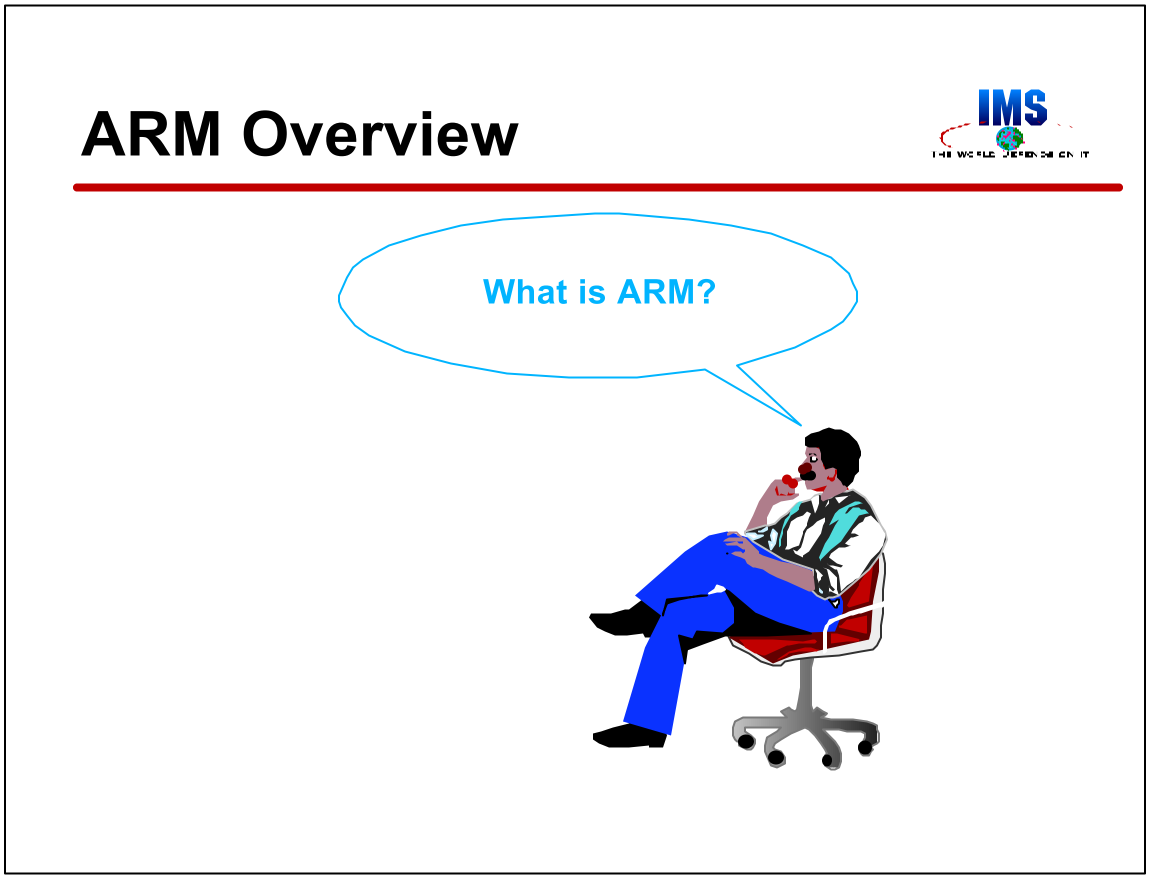


## ARM and IMS

- **Agenda**
  - Automatic Restart Management (ARM) Overview
    - What is ARM?
  - IMS's Support for ARM
    - What has been added to IMS for ARM?
  - Implementing ARM for IMS
    - What needs to be done to use ARM with IMS?

This presentation is organized in three sections. The first explains what ARM is. The second explains IMS's support for ARM is covered. The third explains how ARM may be implemented for IMS systems.

Advice on using both DB2 and CICS with IMS and ARM is included. VTAM and TCP/IP considerations are also covered.



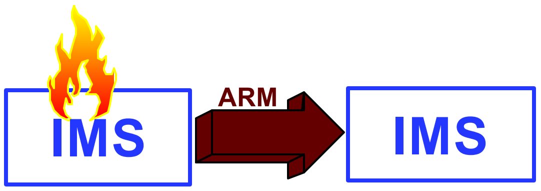
## ARM Overview

What is ARM?

First, we will answer the question, "What is ARM?"

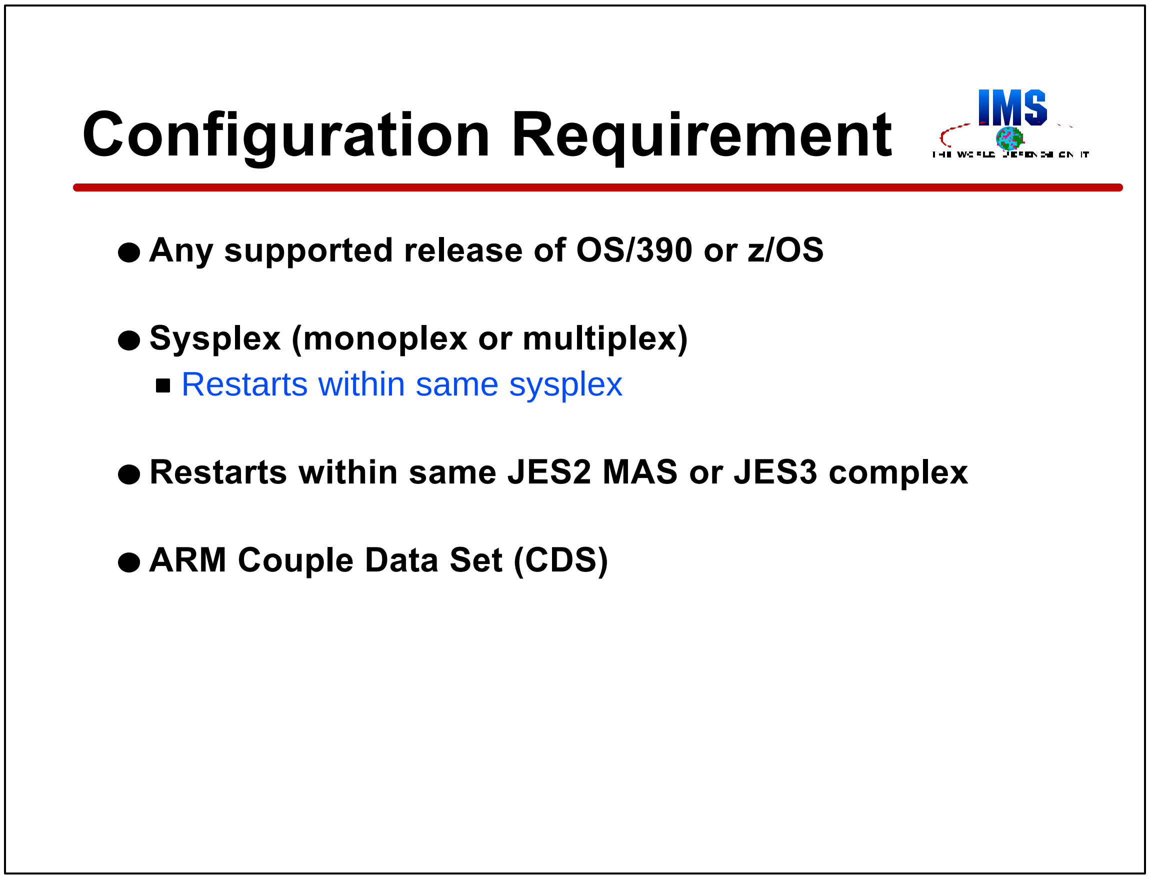
## ARM Purpose

- **What ARM is:**
  - Automated job and started task restart
  - Managed by MVS
  - Sysplex wide
- **ARM benefits:**
  - Minimized outage times
  - Multiple processors exploitation
  - Increased availability



Automatic Restart Management (ARM) provides for automated restarts of jobs and started tasks. The restarts are managed by MVS (OS/390). This is done sysplex wide. That is, restarts may occur across different systems in a Parallel Sysplex.

The benefits of these restarts are probably obvious. Since they are automated, outage times are minimized. Since the restarts are sysplex wide, users may exploit multiple processors. If one processor fails, jobs and started tasks with ARM support may be moved to a surviving processor. This produces increased availability. That's the point of ARM support!



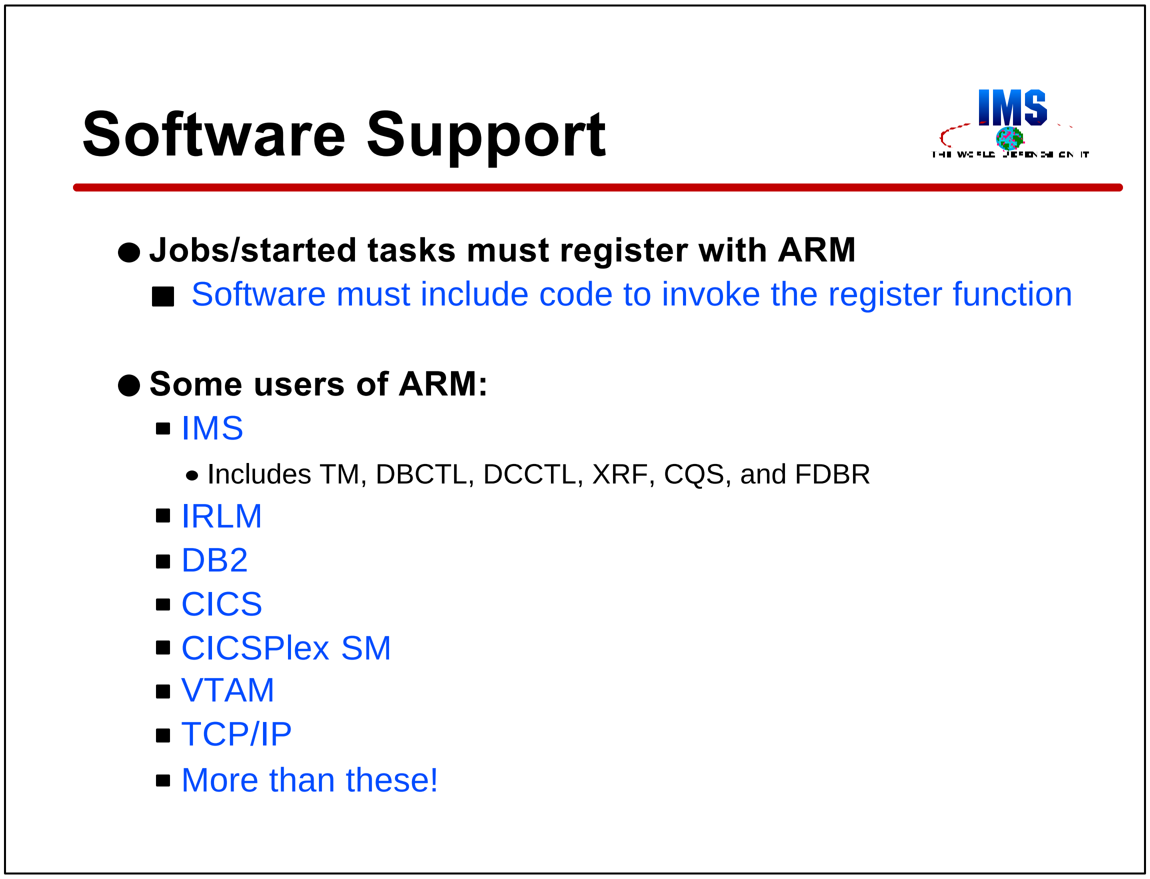
## Configuration Requirement

- Any supported release of OS/390 or z/OS
- Sysplex (monoplex or multiplex)
  - Restarts within same sysplex
- Restarts within same JES2 MAS or JES3 complex
- ARM Couple Data Set (CDS)

MVS SP 5.2.0 and all releases of OS/390 provide ARM support.

ARM requires sysplex support; however, a monoplex is sufficient. All restarts are within the same sysplex. Also, restarts are restricted to the same JES2 MAS or JES3 complex.

Finally, an ARM couple data set must be provided and formatted.



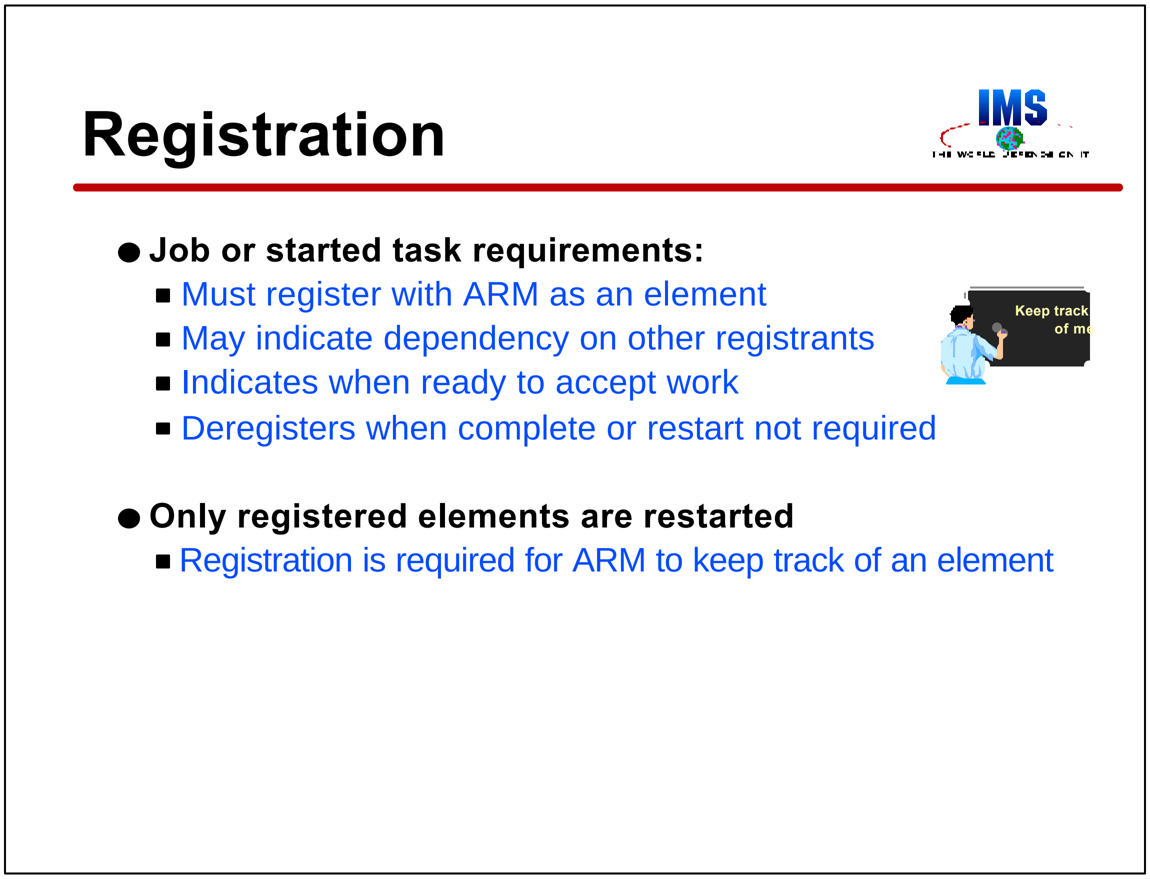
## Software Support

- **Jobs/started tasks must register with ARM**
  - Software must include code to invoke the register function
- **Some users of ARM:**
  - IMS
    - Includes TM, DBCTL, DCCTL, XRF, CQS, and FDBR
  - IRLM
  - DB2
  - CICS
  - CICSPlex SM
  - VTAM
  - TCP/IP
  - More than these!

Only those jobs and started task which register with ARM may be restarted by ARM. These jobs and started tasks must include code to do this registration.

Many software products which have written code to register with ARM are listed here. It should be noted that VTAM's support will cause it to be restarted on the same MVS when it ABENDs. It will not be restarted on a different MVS when its MVS fails. For the other products, both types of restarts are supported.

There are more implementers of ARM support. For example, Resource Recovery Management Services (RRMS, sometimes seen as RRS) of OS/390 Release 3, CA-OPS/MVS II Version 4.2, and TME 10 NetView for OS/390 V1R1 have ARM support.



## Registration

- **Job or started task requirements:**
  - Must register with ARM as an element
  - May indicate dependency on other registrants
  - Indicates when ready to accept work
  - Deregisters when complete or restart not required

- **Only registered elements are restarted**
  - Registration is required for ARM to keep track of an element

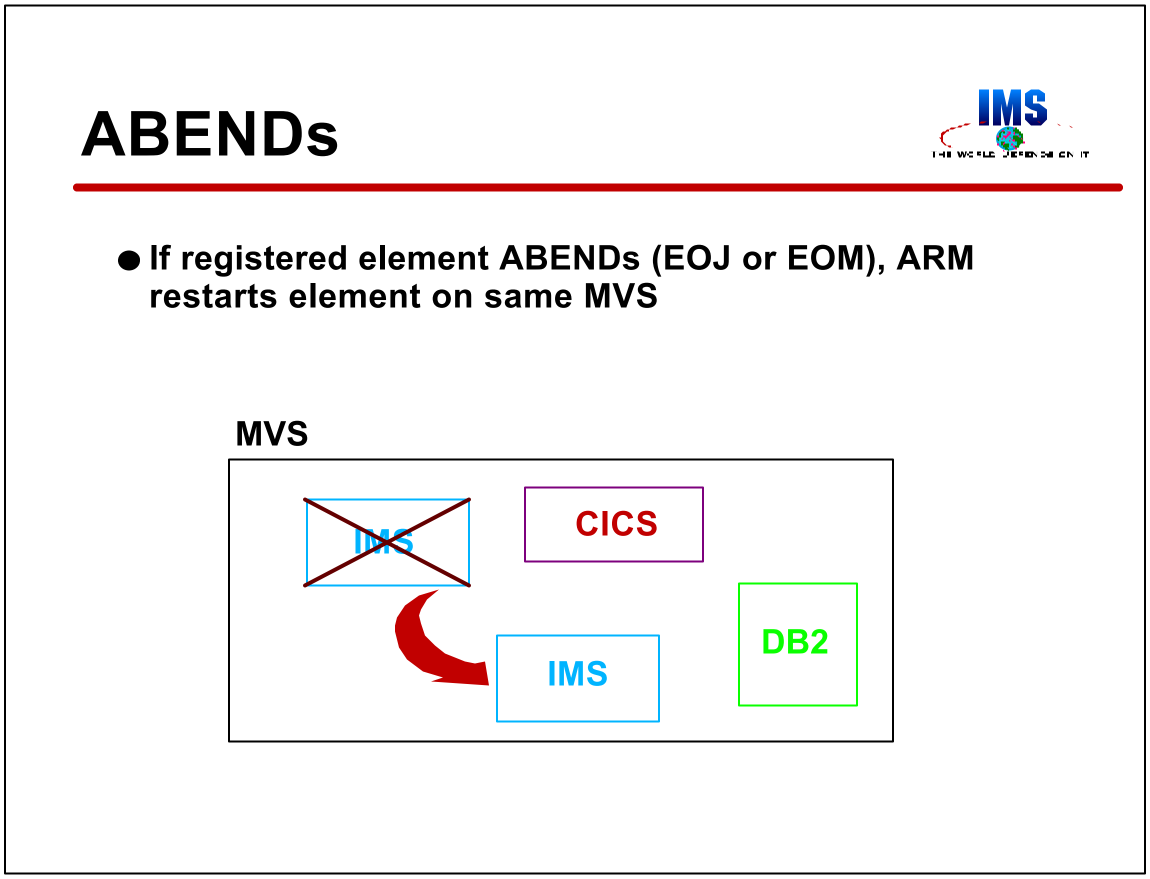
To use ARM, a job or started task must register as an element. Element is the term that ARM uses for those entities that it will track and restart.

When an element registers, it may indicate if it is dependent on some other element, that is another registrant. This dependency will be explained later.

When an element registers it may also indicate when it is ready to accept work. This also will be explained later.

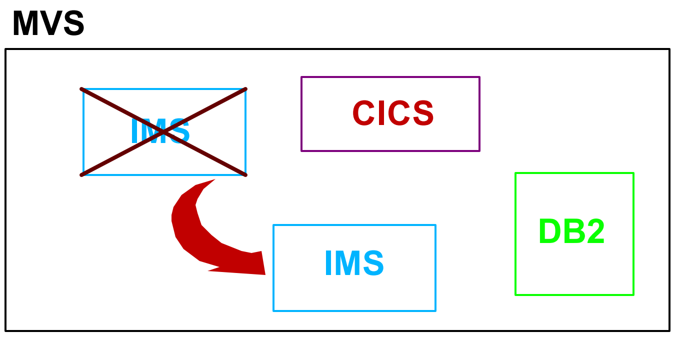
Elements must also deregister. This tells ARM not to restart them when they terminate.

ARM only restarts registered jobs and started tasks. This registration is required for ARM to keep track of them.



## ABENDs

- If registered element ABENDs (EOJ or EOM), ARM restarts element on same MVS



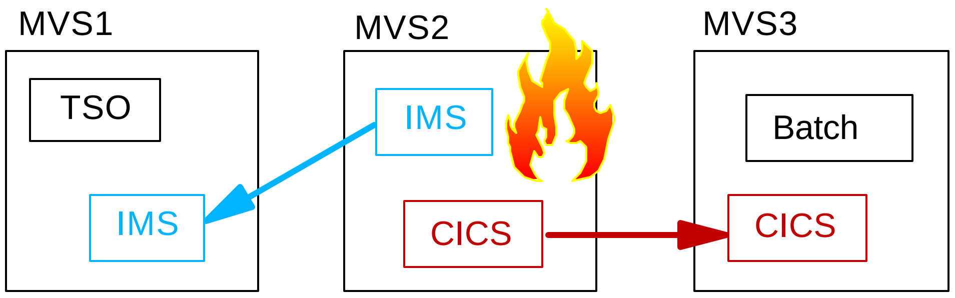
When a registered elements reaches end-of-job (EOJ) or end-of-memory (EOM), ARM will restart it. Typically, this will be an ABEND of the job or started task. This restart will always be on the same MVS system where the job or started task was running. It makes sense to restart the element on the same MVS. This is where we wanted it to run before it ABENDed. We still want it to run there.

A further explanation of the distinction between EOJ or EOM and ABEND may be needed. EOJ is the end of a job. EOM is the end of a started task. A user of ARM will register at initialization and deregister before terminating. A failure between these times is an abnormal termination (ABEND). ARM does not actually require an ABEND to restart an element. Anytime that an element is still registered when it terminates, ARM will invoke its restart processes for the element. Since all reasonable users of ARM deregister before normal terminations, we will refer to the termination of registered elements as ABENDs.

In this example, IMS ABENDs and it is restarted by ARM.

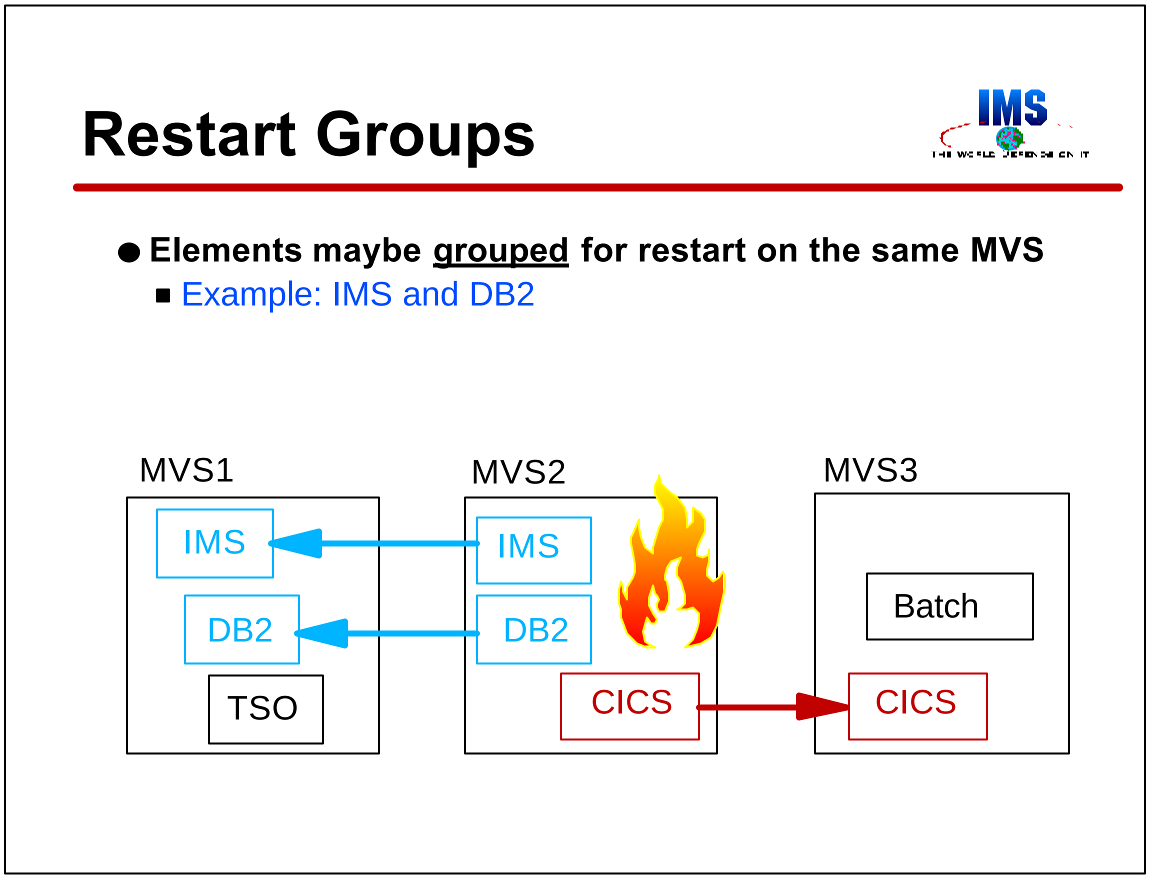
## System Failures

- If  MVS fails, ARM restarts elements elsewhere in sysplex.



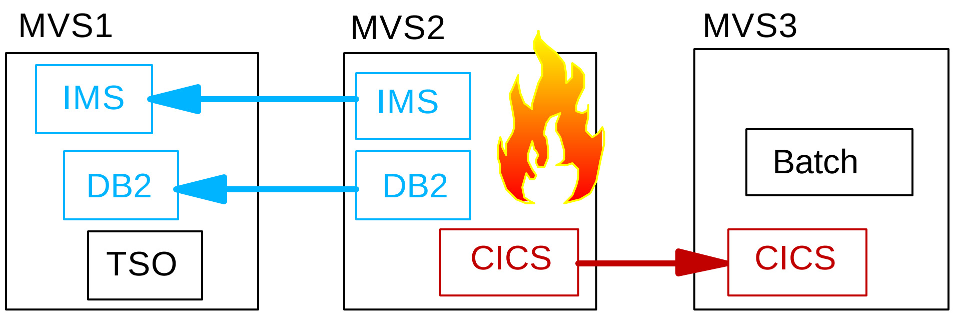
If an MVS system fails or the processor on which it is running fails, ARM will restart its registered elements on another MVS system in the sysplex.  These are called "cross-system restarts".

In this example, MVS2 fails.  ARM restarts IMS on MVS1 and restarts CICS on MVS3.



## Restart Groups

- Elements maybe grouped for restart on the same MVS
  - Example: IMS and DB2



ARM allows an installation to define groups of elements. When an MVS system fails, all elements of a group are restarted on the same MVS. A typical group would be an IMS DBCTL system and all of the CICS systems using it. Another group might be a DB2 and the IMSs and CICSs using it.

In the example shown here, IMS and DB2 are in a group. CICS is not in the group. When MVS2 fails, ARM restarts IMS and DB2 on the same MVS. In this case, it is MVS1. ARM may restart CICS on either MVS. In this case, it is restarted on MVS3.

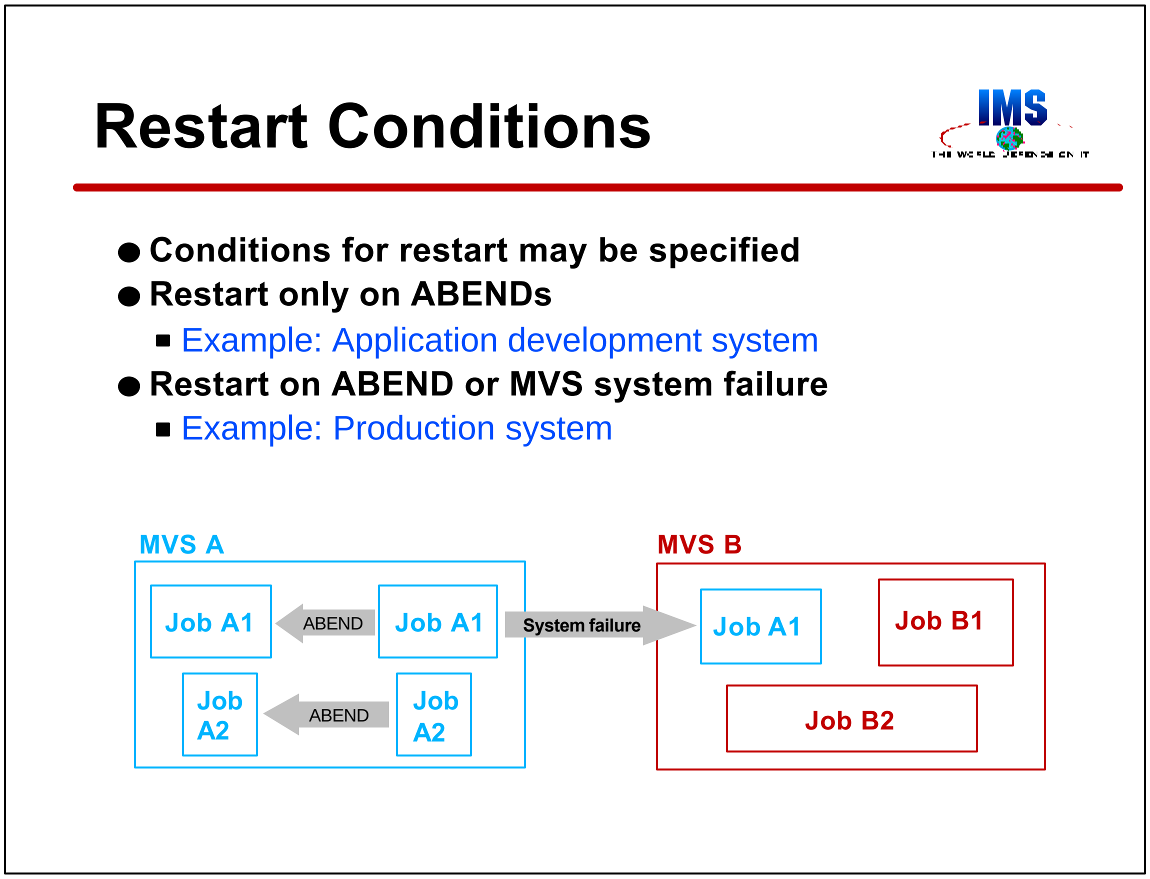
## Restart Groups

- Elements in group may be restarted in specified order
  - Example: DBCTL before CICS
- Elements in group may be restarted at pacing intervals
  - Example: At 20 second intervals
- Minimum CSA/ECSA may be specified for target MVS system
- Target MVS selected by available capacity
  - If using WLM goal mode

When groups are defined to ARM, users may specify the order in which elements of the group are started.

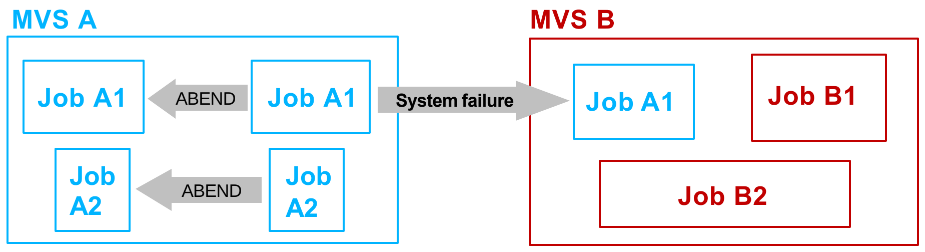
Users may specify a pacing interval for restarts. The pacing interval determines the amount of time between restarts of elements in a group. This may be used prevent the flooding of a system with restarts for many elements, such as CICS AORs.

Users may specify minimum amounts of CSA and ECSA memory that must be available on a system for a group to be restarted on it. This may be used to prevent unsuccessful restarts when a group's CSA and ECSA requirements exceed what are available on a system.



## Restart Conditions

- Conditions for restart may be specified
- Restart only on ABENDs
  - Example: Application development system
- Restart on ABEND or MVS system failure
  - Example: Production system



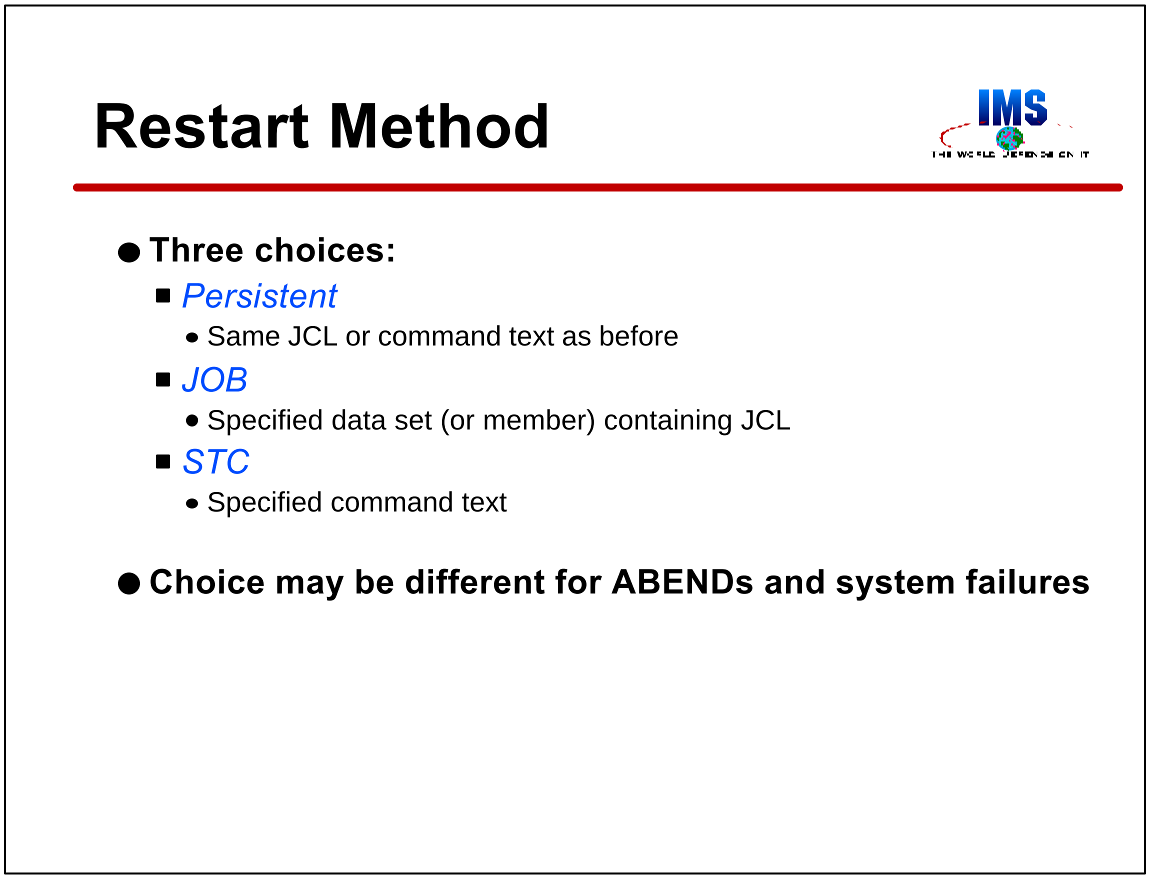
Users may control when ARM will restart an element. There are two choices.

First, the element may be restarted only if it ABENDs, not when its MVS system fails. This might be used for application development systems. An installation might not want such a system moved to a system where production work is being processed.

Second, the element may be restarted on ABENDS and system failures. This is the expected choice for production systems.

In the example which is shown here, A1 is restarted for both ABENDs and system failures. A2 is restarted only for ABENDs.

Actually, there is a third choice for restarts. Users may specify that an element should never be restarted. We will see how this is done when we look at ARM policies.



## Restart Method

- **Three choices:**
  - *Persistent*
    - Same JCL or command text as before
  - *JOB*
    - Specified data set (or member) containing JCL
  - *STC*
    - Specified command text
- **Choice may be different for ABENDs and system failures**

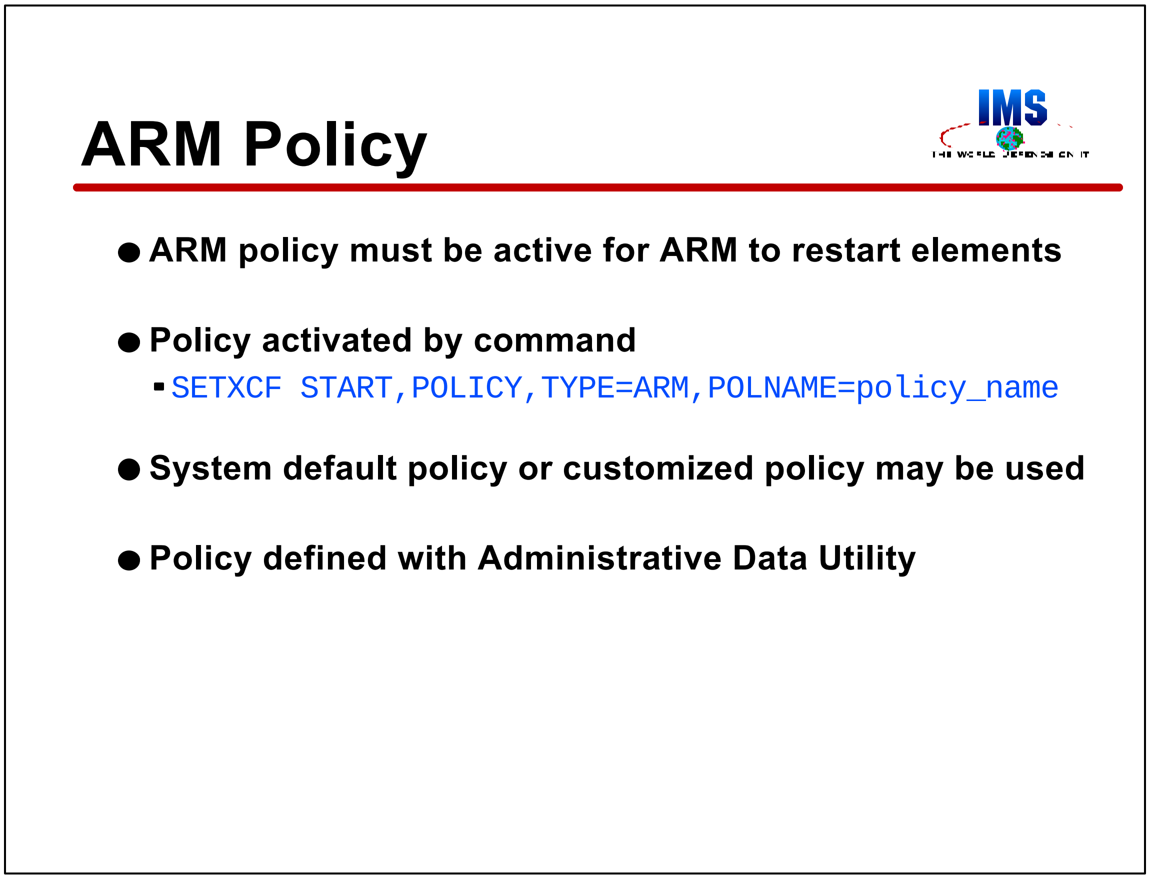
Users may control how a job or started task is restarted.

First, they may choose to have the restart done with the same job JCL or the same start command as was used for the original execution.

Second, they may specify that the restart be done as a JOB using the JCL stored in a data set or member of a data set. This may be done even when the element was originally a started task.

Third, they may specify that the restart be done by issuing a start command. In this case, they specify the text of the start command. This may be done even when the element was originally a job.

The choices may be different for ABENDs and system failures. For example, an installation may specify that on ABENDs the element is to be restarted using the same JCL or start command that was originally used. They may also specify that on system failures, the element should be restarted using a particular set of JCL.



## ARM Policy

- ARM policy must be active for ARM to restart elements
- Policy activated by command
  - `SETXCF START,POLICY,TYPE=ARM,POLNAME=policy_name`
- System default policy or customized policy may be used
- Policy defined with Administrative Data Utility

ARM actions are controlled by its policy. For ARM to restart any element, an ARM policy must be active.

A policy is activated with a `SETXCF,START` command. The command indicates that an ARM policy is to be started. This is done by specifying TYPE=ARM. The command also identifies the policy to start via the NAME parameter. If the NAME parameter is not included, the default policy is started by the command.

Policies are defined with the MVS Administrative Data Utility. This is the same utility that is used to define many other Parallel Sysplex policies, such as Coupling Facility Resource Management (CFRM) policies and Sysplex Failure Management (SFM) policies.

## ARM Policy (cont.)

- **For *elements* the policy defines:**
  - Number of restarts in an interval
    - Zero indicates that the element should not be restarted
  - Whether to restart on system failures and ABENDs
    - Or only on ABENDs
  - Method of restart
    - Persistent, JOB, or STC
  - Maximum wait times on restart
    - Registration - time to register
    - Ready - time to become ready

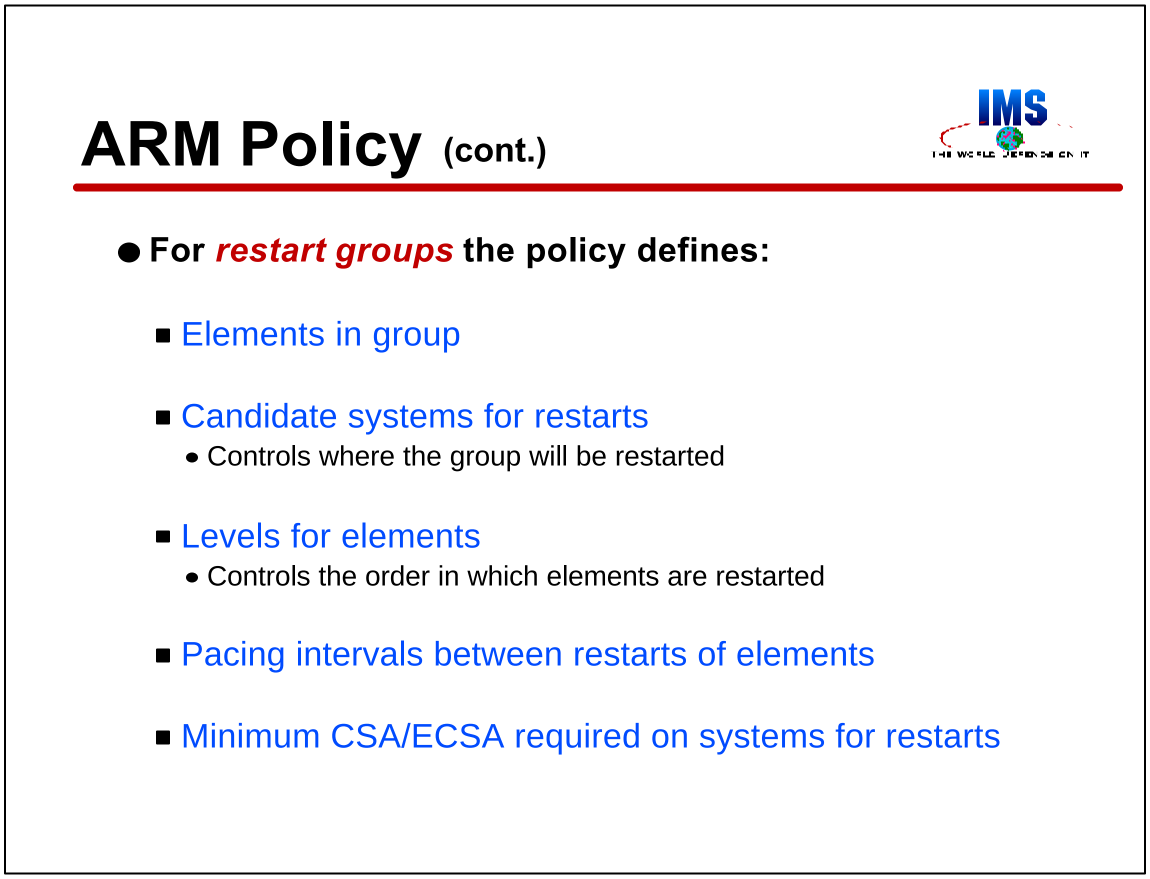
A policy may specify several parameters for an element.

The number of restarts within a time interval may be limited.  For example, an installation may specify that a particular job may be restarted only three time in a five minute interval.  This is done to avoid continually restarting an element  which fails after every restart.  If an installation does not want a registered element restarted, it may specify zero restarts in the time interval.

Whether a restart for an element will occur only for ABENDs or for both ABENDs and system failures is specified in the policy.

The method of restart, either persistent, JOB, or STC is specified in the policy.  When JOB is chosen a data set or data set member containing the JCL is specified.  When STC is chosen, the start command text is specified.

The policy includes two wait time specifications.  Registration wait time is the time between the restart of an element and the time it reregisters with ARM.  If this time is exceeded, ARM deregisters the element.  That is, ARM discards its information about the element.  Also, any elements waiting for this element to become ready are released from their wait. The second wait time is the time that a restarted element has to become ready.  If this time is exceeded, any elements waiting for this element to become ready are released from their wait.



## ARM Policy (cont.)

- For ***restart groups*** **the policy defines:**
  - Elements in group
  - Candidate systems for restarts
    - Controls where the group will be restarted
  - Levels for elements
    - Controls the order in which elements are restarted
  - Pacing intervals between restarts of elements
  - Minimum CSA/ECSA required on systems for restarts

A policy may specify several parameters for groups.

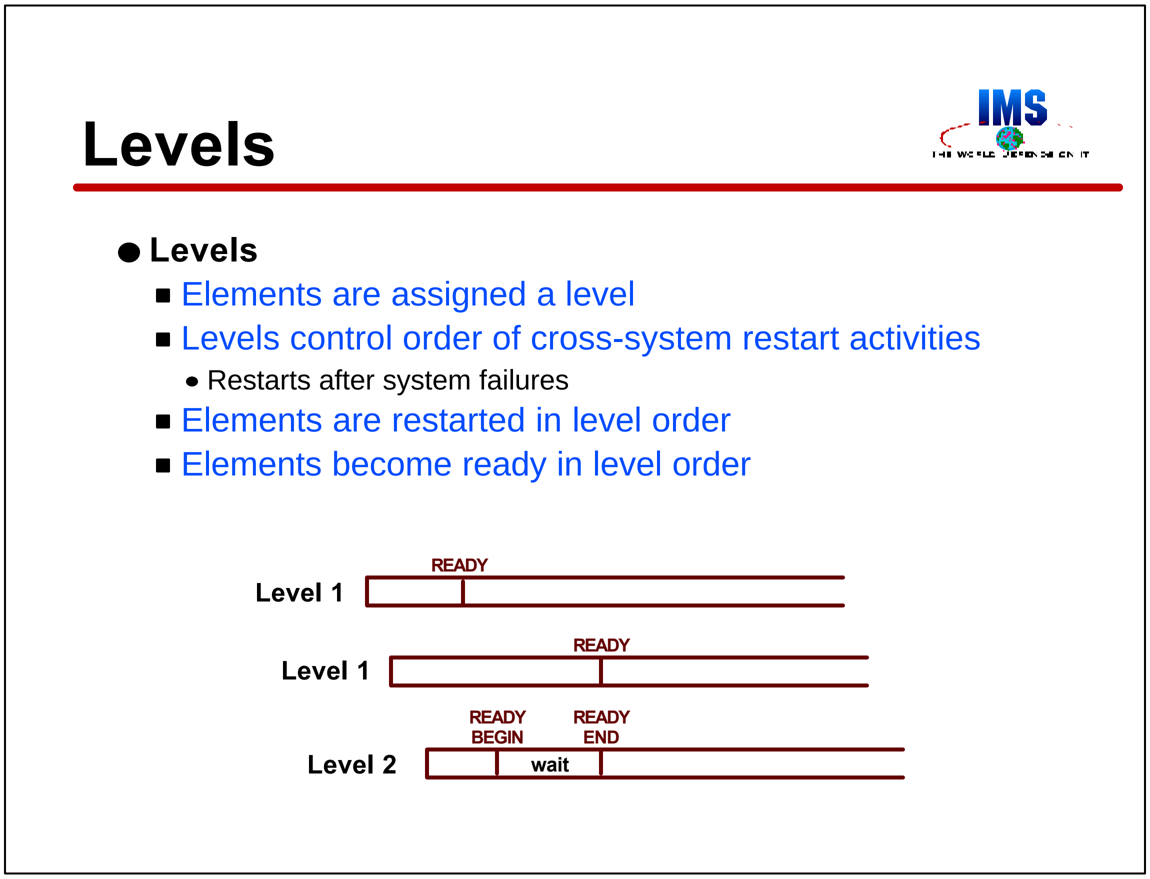
The policy defines groups and the members of each group.

The systems on which a group may be restarted are specified. A Parallel Sysplex may have many MVS systems. Some groups may be restricted to running on only certain systems. For example, an installation may not allow a group to be restarted on a system which does not have enough processing power to adequately serve it.

Levels may be assigned to elements in a group. When a group is restarted, ARM will start all of the elements in Level 1 before any in Level 2 are started. Level 2 elements are started before those in level 3, etc.

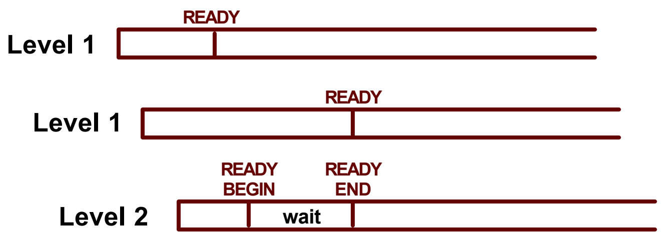
The policy may include a pacing interval for the elements in a group. The pacing interval specifies the delay between the starts of elements in a group. This allows an installation to stagger restarts when they might cause contention in the system.

Each group may have free CSA specifications. There are separate specifications for CSA (below the 16M line) and ECSA. They tell ARM not to restart the group on a system which does not have at least these amounts of CSA and ECSA available. This specification is not effective when Workload Manager is running in compatibility mode on a system.



## Levels

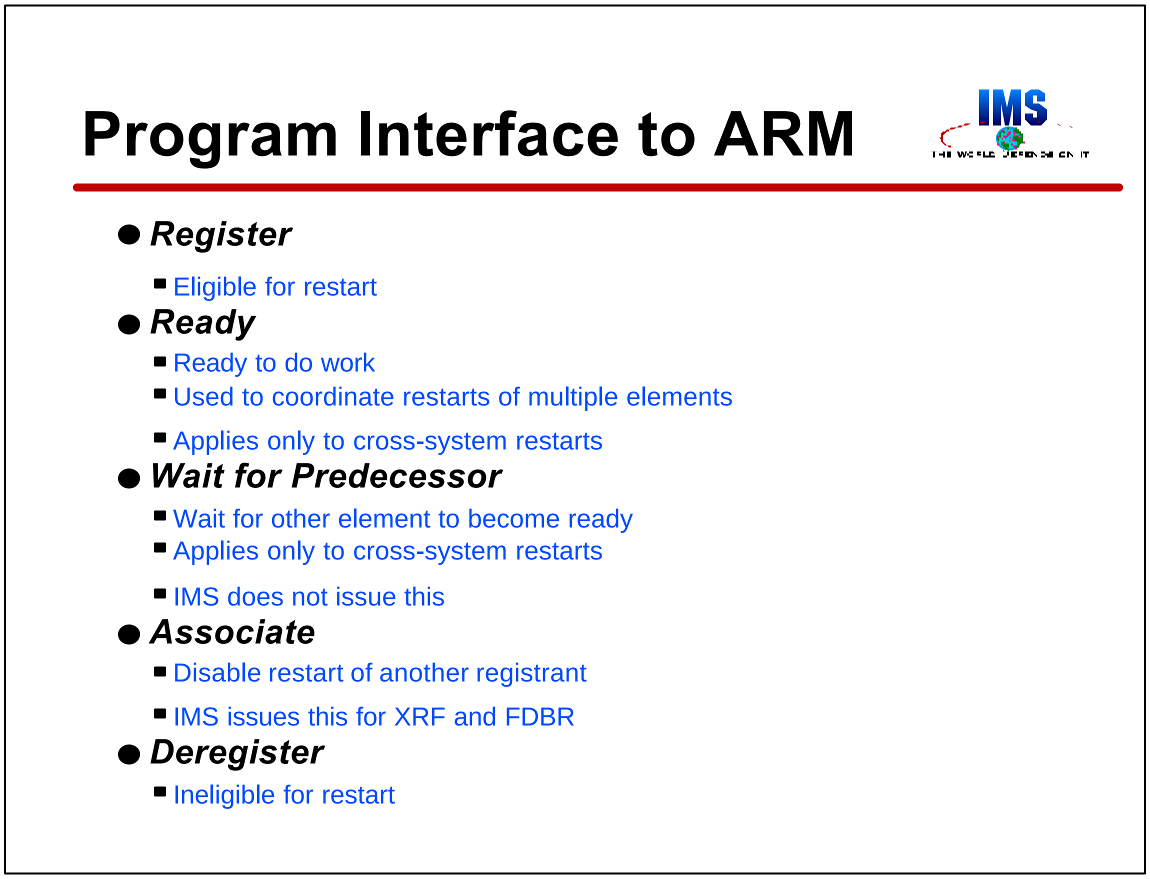
- **Levels**
  - Elements are assigned a level
  - Levels control order of cross-system restart activities
    - Restarts after system failures
  - Elements are restarted in level order
  - Elements become ready in level order



Levels were mentioned on the previous page. They are only used on cross-system restarts. They control two things. First, they control the order in which elements of a group are restarted. Second, they control the order in which the elements of a group become ready.

When an element is restarted it must do two things. First, it must issue a register request to ARM. Second, it must tell ARM that it is ready to accept new work. If lower level elements in the group are not yet ready, ARM will make an element wait when it issues the ready request.

The example on this page shows two elements in level 1 and one in level 2. The level 1 elements are restarted before the level 2 element. The level 2 element issues its ready request to ARM before the second of the level 1 elements issues its ready request. ARM makes the level 2 element wait. When both level 1 elements are ready, the level 2 element is released from its wait.



## Program Interface to ARM

- ***Register***
  - Eligible for restart
- ***Ready***
  - Ready to do work
  - Used to coordinate restarts of multiple elements
  - Applies only to cross-system restarts
- ***Wait for Predecessor***
  - Wait for other element to become ready
  - Applies only to cross-system restarts
  - IMS does not issue this
- ***Associate***
  - Disable restart of another registrant
  - IMS issues this for XRF and FDBR
- ***Deregister***
  - Ineligible for restart

ARM provides programs, such as IMS, with five requests types. They are used to inform ARM of the status of elements.

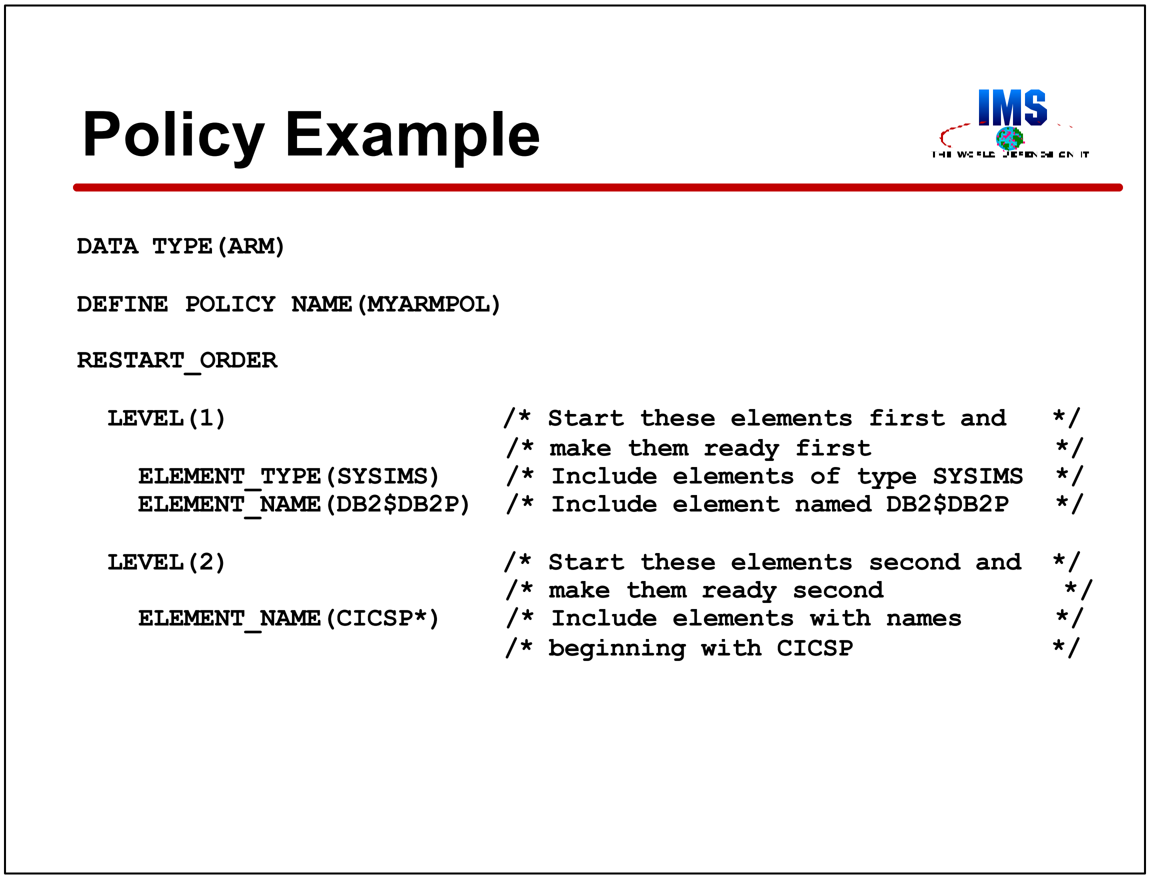
A program must register with ARM for ARM to track it and restart it. This is typically done at initialization time for a program. ARM places information about the element in the ARM couple data set.

Ready indicates that a program is ready to do work. It is only applicable to cross-system restarts. Its only effect is to cause a member of a restart group to wait on another member to become ready. Elements wait on other members which are in a lower level.

Wait for predecessor is similar to ready. It may be used to cause a member of a restart group to wait on another member of the group to become ready. It differs in that it does not cause the requester to become ready and the other member does not have to be at a lower level. IMS does not use the wait for predecessor capability.

Associate is used to inform ARM not to restart another element. It is used when a program will handle the work of another program when it fails. This is used by XRF systems. When an IMS XRF alternate is started, it registers to ARM and then it issues the associate request. This tells ARM not to restart the XRF active if it should fail. Similarly, an IMS Fast Database Recovery (FDBR) system registers and issues the associate for the IMS system it is tracking.

Deregister causes ARM to stop tracking an element. The information about the element is deleted from the ARM couple data set.



## Policy Example

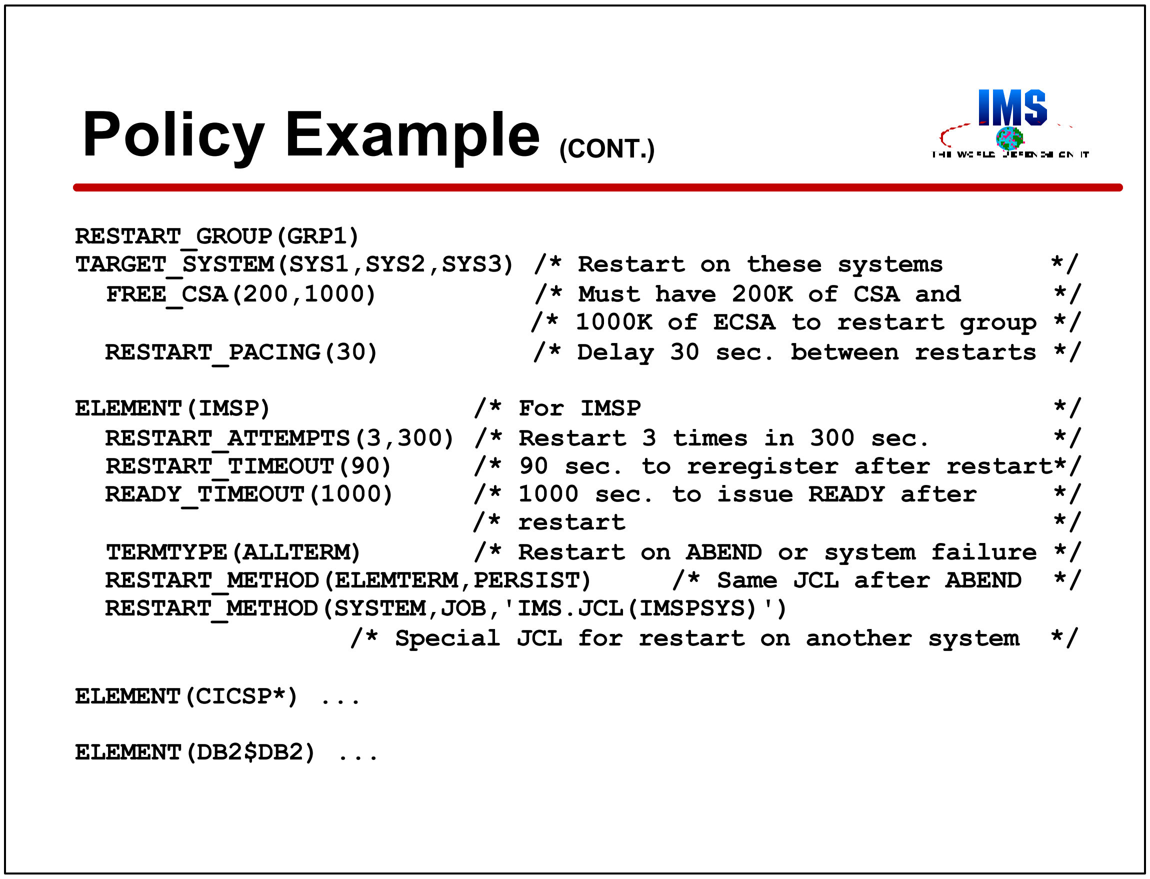
```
DATA TYPE(ARM)

DEFINE POLICY NAME(MYARMPOL)

RESTART_ORDER

  LEVEL(1)                  /* Start these elements first and   */
                            /* make them ready first            */
    ELEMENT_TYPE(SYSIMS)    /* Include elements of type SYSIMS  */
    ELEMENT_NAME(DB2$DB2P)  /* Include element named DB2$DB2P   */

  LEVEL(2)                  /* Start these elements second and  */
                            /* make them ready second           */
    ELEMENT_NAME(CICSP*)    /* Include elements with names      */
                            /* beginning with CICSP             */
```

This is a sample ARM policy definition.  It is not a recommended policy.  Instead, it is designed to show some of the capabilities that an installation might use with ARM.

The policy name is MYARMPOL.  To use this policy we would issue the SETXCF START command specifying a policy name of MYARMPOL.

The policy assigns all elements of type SYSIMS to level 1.  Similarly, it assigns any element with the name DB2$DB2P to level 1.  Any element with a name beginning with CICSP is assigned to level 2.  We will see how element names and element types are determined later.

The policy is continued on the next page.



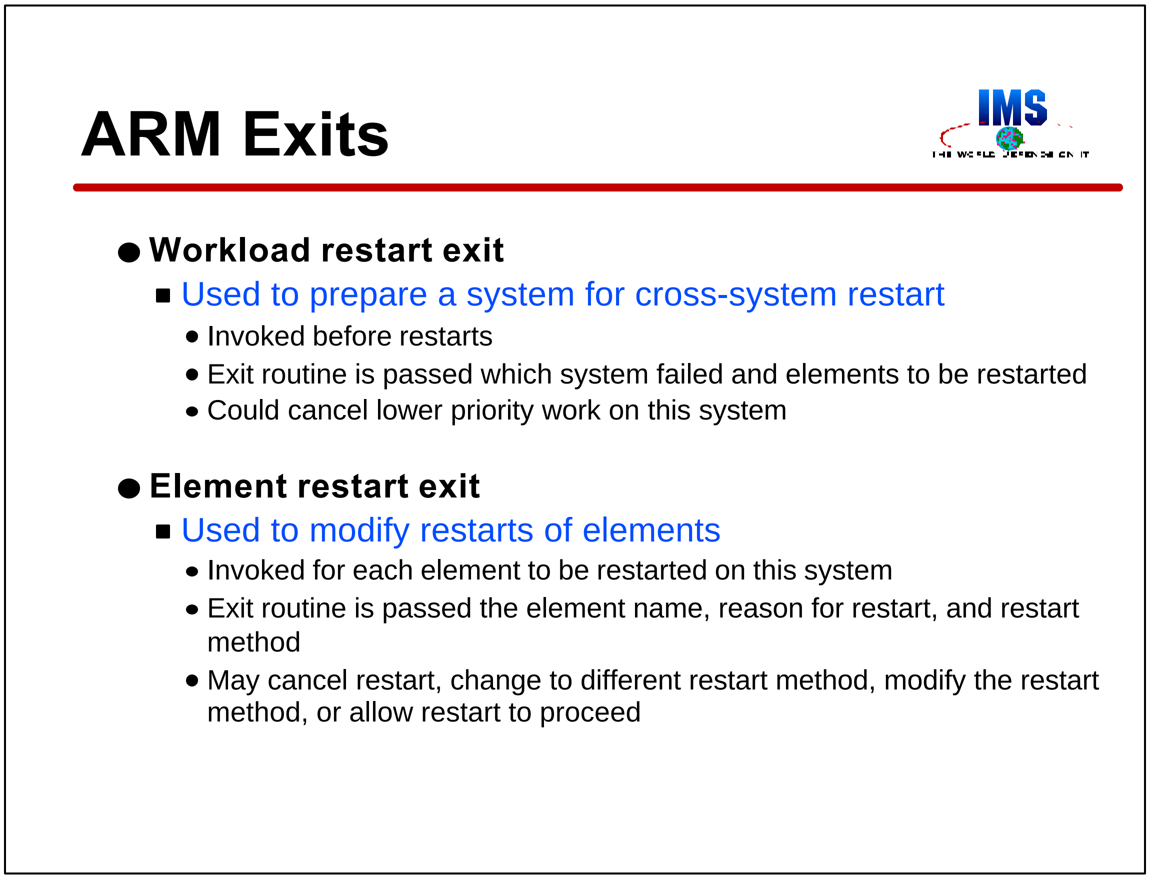
## Policy Example (CONT.)

```
RESTART_GROUP(GRP1)
TARGET_SYSTEM(SYS1,SYS2,SYS3) /* Restart on these systems       */
  FREE_CSA(200,1000)          /* Must have 200K of CSA and      */
                              /* 1000K of ECSA to restart group */
  RESTART_PACING(30)          /* Delay 30 sec. between restarts */

ELEMENT(IMSP)             /* For IMSP                          */
  RESTART_ATTEMPTS(3,300) /* Restart 3 times in 300 sec.       */
  RESTART_TIMEOUT(90)     /* 90 sec. to reregister after restart*/
  READY_TIMEOUT(1000)     /* 1000 sec. to issue READY after    */
                          /* restart                           */
  TERMTYPE(ALLTERM)       /* Restart on ABEND or system failure */
  RESTART_METHOD(ELEMTERM,PERSIST)     /* Same JCL after ABEND  */
  RESTART_METHOD(SYSTEM,JOB,'IMS.JCL(IMSPSYS)')
                  /* Special JCL for restart on another system  */

ELEMENT(CICSP*) ...

ELEMENT(DB2$DB2) ...
```

These statements define group GRP1. It may restarted on systems SYS1, SYS2, and SYS3. The group requires at least 200K of CSA and 1000K of ECSA on the system. Elements will be restarted at 30 second intervals.

The members of the group are IMSP, all elements with names beginning with CICSP, and element DB2$DB2.

IMSP will be restarted a maximum of three times in 300 seconds. When restarted, it must reregister in 90 seconds and become ready in 1000 seconds. It will be restarted on all types of terminations, that is, both ABENDs and system failures. After ABENDs (ELEMTERM) it will be restarted as it was previously started (PERSIST). After system failures, it will be restarted as a job using the JCL in IMS.JCL(IMSPSYS).



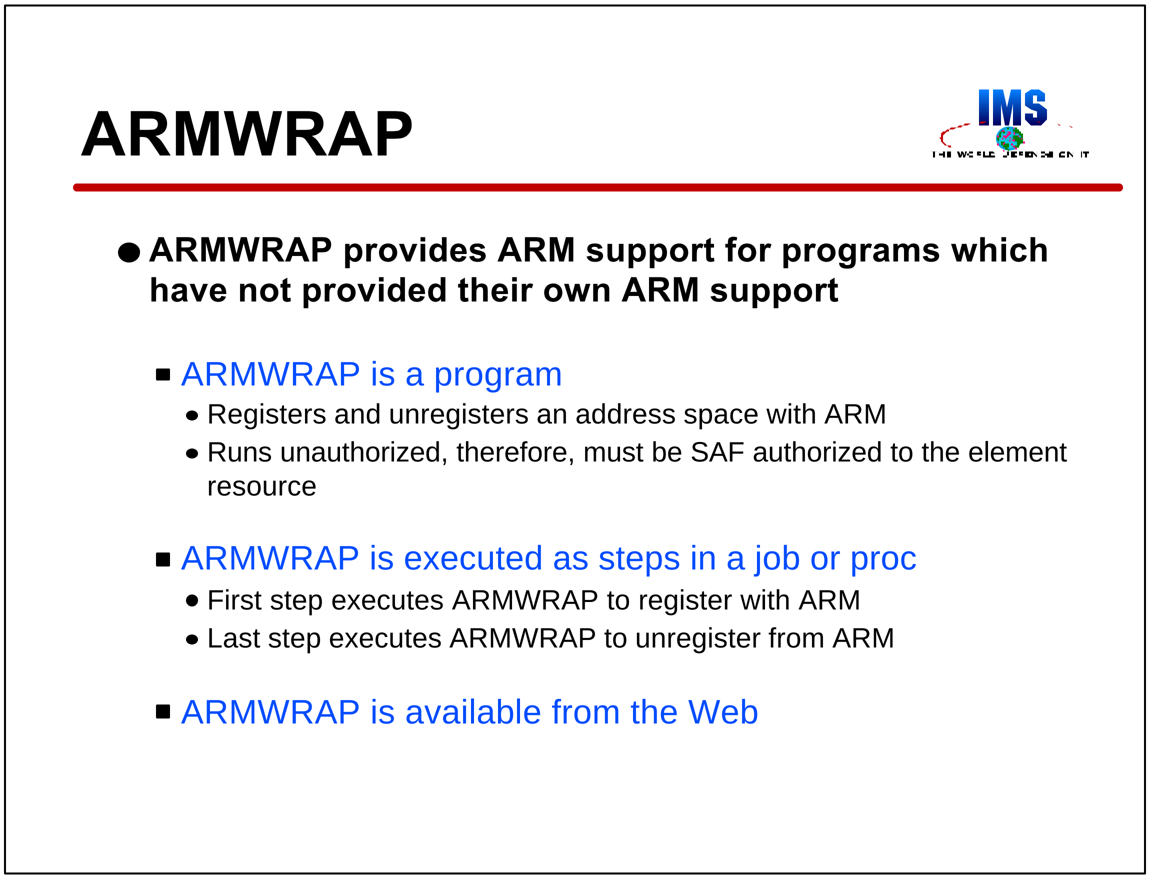
## ARM Exits

- **Workload restart exit**
  - Used to prepare a system for cross-system restart
    - Invoked before restarts
    - Exit routine is passed which system failed and elements to be restarted
    - Could cancel lower priority work on this system
- **Element restart exit**
  - Used to modify restarts of elements
    - Invoked for each element to be restarted on this system
    - Exit routine is passed the element name, reason for restart, and restart method
    - May cancel restart, change to different restart method, modify the restart method, or allow restart to proceed

The workload restart exit may be used to prepare a system that will be receiving additional workload from a failing system. It is invoked when a system fails. It is invoked on each system on which restarts will occur. The exit routine is passed the name of the failing system, the number of elements to be restarted on the system, and the names of those elements. It might be used to cancel work currently running on the system.

The element restart exit may be used to modify or cancel the restart of an element. It is invoked for both cross-system restarts and for restarts due to ABENDs. It is invoked once for each element. The routine is passed the name of the element to be restarted, the reason for the restart (element or system termination), and the method of restart (job or start command). The routine may cancel the restart. It may change the restart method; for example, it may change from persistent JCL to a start command. It may modify the restart method; for example, it may change the name of the data set containing the JCL or change the start command text. Of course, it can let the restart proceed without any changes.

These exits are documented in the *OS/390 MVS Installation Exits* manual.



## ARMWRAP

- **ARMWRAP provides ARM support for programs which have not provided their own ARM support**
  - ARMWRAP is a program
    - Registers and unregisters an address space with ARM
    - Runs unauthorized, therefore, must be SAF authorized to the element resource
  - ARMWRAP is executed as steps in a job or proc
    - First step executes ARMWRAP to register with ARM
    - Last step executes ARMWRAP to unregister from ARM
  - ARMWRAP is available from the Web

ARMWRAP is a program which provides ARM restart capability for programs which have not implemented their own ARM support. ARMWRAP may be used to register and unregister with ARM. This is done through parameters passed to ARMWRAP.

ARMWRAP runs unauthorized. Registration with ARM by unauthorized callers requires SAF (RACF or equivalent) authorization to the element resource. This is the element name specified in the ARMWRAP parameters.

A typical use of ARMWRAP includes making it the first and last steps in a job or PROC that was previously one step. In the first step ARMWRAP registers with ARM. In the last step ARMWRAP unregisters with ARM. If the job or started procedure fails between the registration and unregistration, ARM will restart it.

ARMWRAP will soon be available for download from the Web. There is no charge for ARMWRAP.

## ARMWRAP (CONT.)

- Example JCL to provide ARM support for APPLPGM
  - Step REGSTEP registers with ARM
  - Step UNREGSTP unregisters from ARM

```
//PROCNAME PROC ...
//*  Register element 'EXAMPLE' using element type of
//*  'XAMP'with ARM.  Restart on all types of terminations.
//REGSTEP  EXEC PGM=ARMWRAP,
//       PARM=(REQUEST=REGISTER,READYBYMSG=N,
//       TERMTYPE=ALLTERM,ELEMENT=EXAMPLE,ELEMTYPE=XAMP)
//APPLSTEP EXEC PGM=APPLPGM
                 .
                 .
                 .
//UNREGSTP EXEC PGM=ARMWRAP,PARM=(REQUEST=UNREGISTER)
```

This is an example of using ARMWRAP.

The REGSTEP step registers with ARM.  The element name is EXAMPLE and the element type is XAMP.  This PROC will be restarted on all types of failures (ALLTERM).  The ARM READY function occurs in this step, not by the issuing of a user message.  This is contolled by the READYBYMSG parameter.

The APPLSTEP step is the usual application program.

The UNREGSTP step unregisters the element in this address space.  That is, it unregisters element name EXAMPLE.

## IMS's Support for ARM

Now, we will look at what has been added to IMS so that it may use ARM.

## IMS's Support for ARM

- **IMS TM, DBCTL, DCCTL, XRF, and FDBR are supported**
  - ARMRST=Y|N execution parameter
  - ARMRST=Y causes IMS to register
- **Control region**
  - Registers with ARM
  - Control region starts DLI SAS and DBRC which do not register
- **Dependent regions, DLI and DBB batch, utilities**
  - Do not register
  - Are not restarted
- **XRF and FDBR**
  - Register with ARM
  - Special treatment for tracked control regions

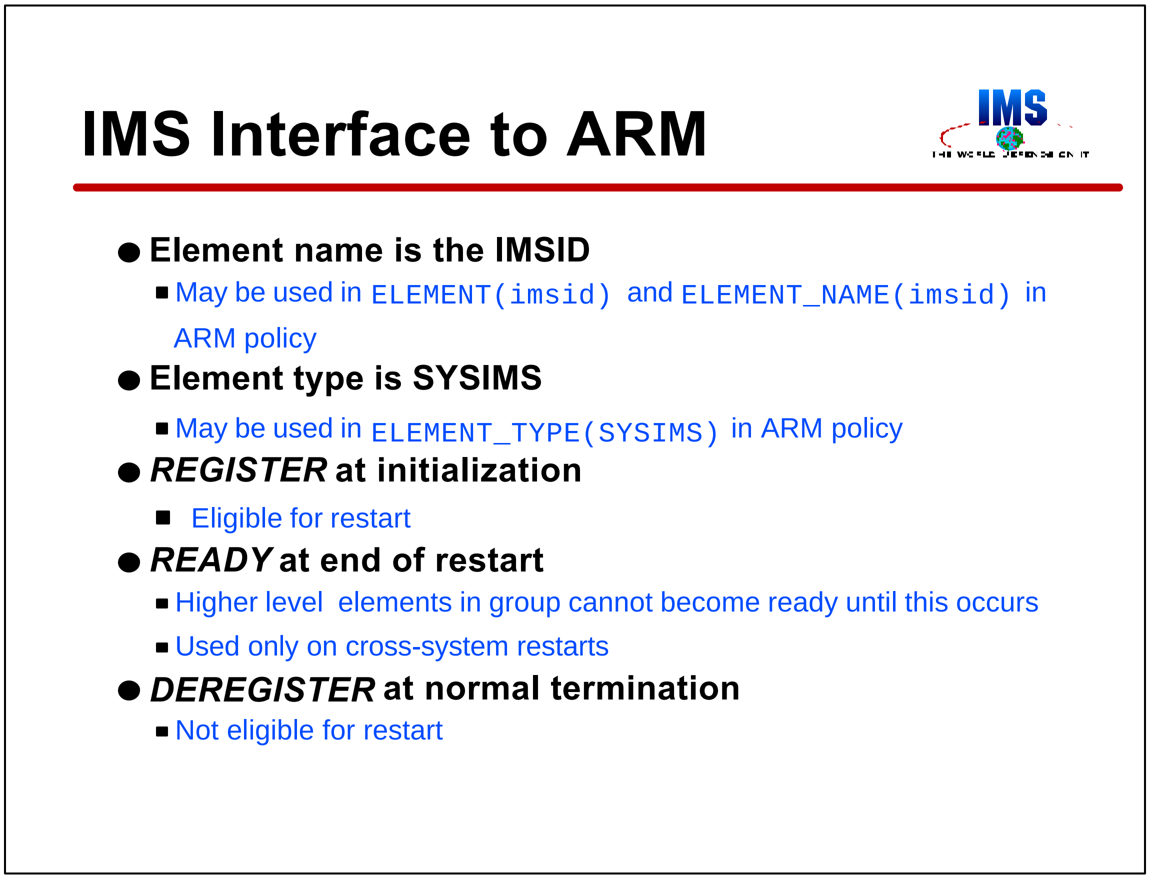
ARM support was added to IMS/ESA 5.1 with APAR PN71392. This support is included with later releases.

The ARMRST execution parameter determines if the IMS system will register with ARM. The default is to register. The parameter applies to the control region for IMS TM, DBCTL, DCCTL, and XRF systems. It is also used by CQS and FDBR.

Only control regions, not dependent regions, register with ARM. The control region starts the DLI SAS and DBRC regions. Dependent regions (MPRs, BMPs, and IFPs) are not started automatically by ARM. The normal procedures, including automated ones, are assumed to be sufficient.

IMS batch jobs (DBB and DLI) and utilities do not have ARM support.

XRF and FDBR have special considerations that will be discussed later.



## IMS Interface to ARM

- **Element name is the IMSID**
  - May be used in `ELEMENT(imsid)` and `ELEMENT_NAME(imsid)` in ARM policy
- **Element type is SYSIMS**
  - May be used in `ELEMENT_TYPE(SYSIMS)` in ARM policy
- ***REGISTER* at initialization**
  - Eligible for restart
- ***READY* at end of restart**
  - Higher level  elements in group cannot become ready until this occurs
  - Used only on cross-system restarts
- ***DEREGISTER* at normal termination**
  - Not eligible for restart

When IMS registers to ARM the element name is the IMSID.  This is the name that is coded in ARM policy statements.

The element type is always SYSIMS.  It may be used on ELEMENT_TYPE parameters in the  ARM policy.

IMS registers to ARM during initialization and issues the ready request at the end of restart.

IMS deregisters from ARM at normal termination.

Information on element names and element types for CQS, FDBR, and IRLM is given later.

## Special Treatment

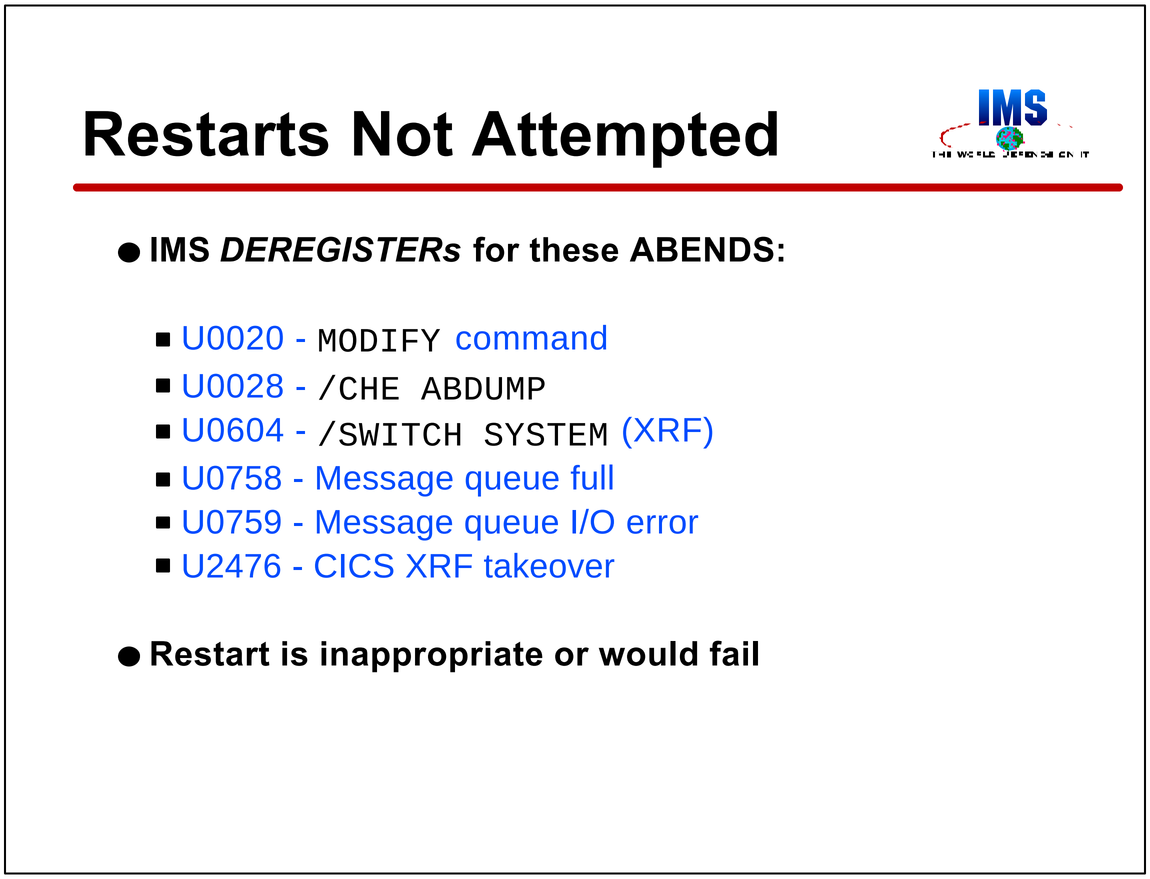
- **Restarts by ARM ignore AUTO=N**
  - /ERE OVERRIDE set if not XRF alternate
  - /ERE BACKUP set for XRF alternate
- **ABEND before restart completes causes *DEREGISTER***
  - Avoids restart that is likely to fail
- **When XRF alternate or FDBR begins tracking, restarting of tracked IMS is disabled.**
  - XRF alternate will be restarted if it fails
  - FDBR will be restarted if it fails

IMS has special treatment for some situations.

When IMS registers with ARM, it is told if this is a restart invoked by ARM.  IMS uses this information to affect the restart processing.  When ARM restarts IMS, IMS ignores any AUTO=N specification.  This allows the restart to be done without operator intervention.  It also allows restarts to be successful using the "PERSIST" method even when the original start was done with AUTO=N.  The OVERRIDE keyword is always used for non-XRF alternates.  This allows the restart of IMS after a system failure to be successful.  When XRF alternates fail, they are restarted with the BACKUP keyword.

If IMS ABENDs before restart processing is complete, IMS deregisters from ARM.  This causes ARM not to restart IMS.  A restart would likely fail, since it would do the same processing that caused the ABEND.

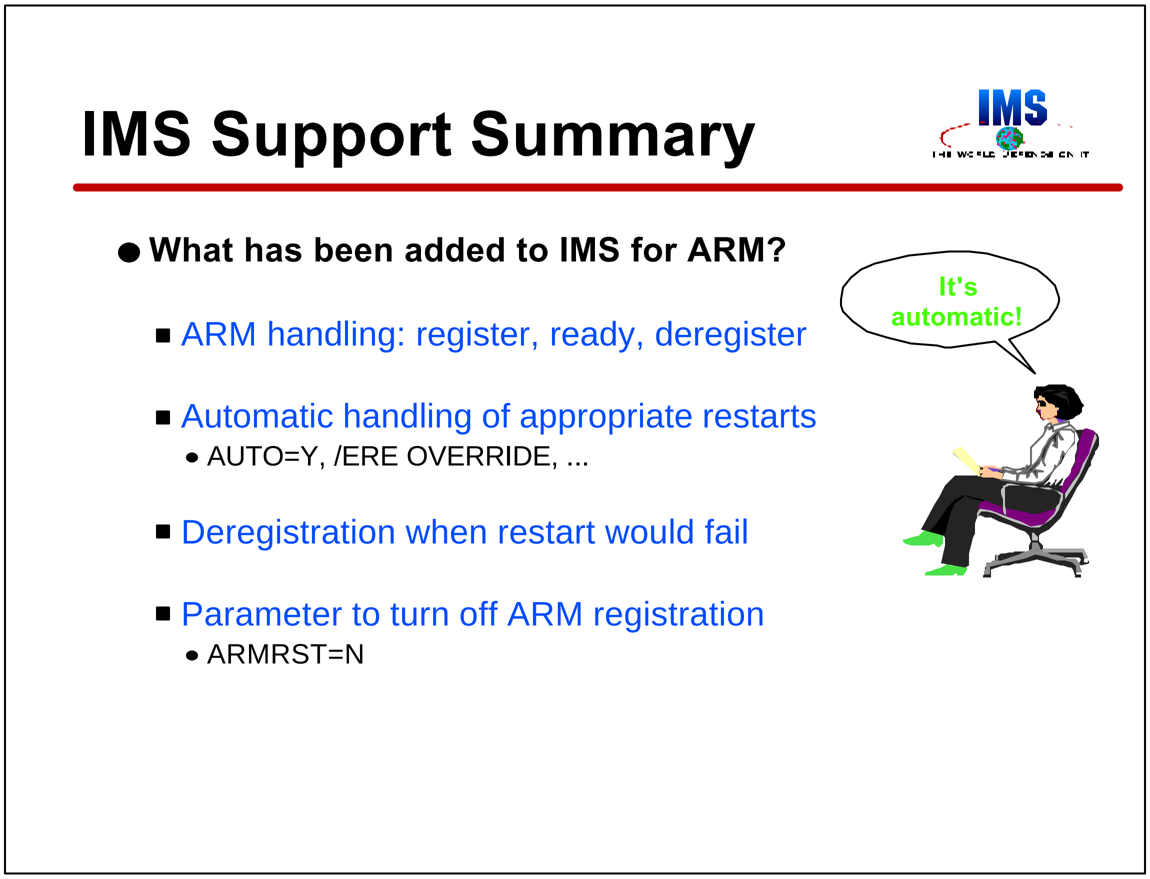
IF ARM is specified for XRF alternate or FDBR regions, these regions will issue an associate request to ARM when they begin tracking a control region.  This associate request specifies the IMS system that is being tracked.  ARM will not restart the tracked system if it fails.  This associate request is only made if the XRF alternate or FDBR region invokes ARM support.  That is, they do not specify ARMRST=N.  An XRF active  should not be restarted by ARM.  If it uses ARM, its XRF alternate should also use ARM.



## Restarts Not Attempted

- **IMS *DEREGISTERs* for these ABENDS:**
  - U0020 - `MODIFY` command
  - U0028 - `/CHE ABDUMP`
  - U0604 - `/SWITCH SYSTEM` (XRF)
  - U0758 - Message queue full
  - U0759 - Message queue I/O error
  - U2476 - CICS XRF takeover
- **Restart is inappropriate or would fail**

Before issuing certain ABENDs, IMS deregisters from ARM. This keeps ARM from restarting IMS. The ABENDs are listed here. Restarts following these ABENDs would either be inappropriate or would fail. For example, a U0028 indicates that the operator terminated IMS with a /CHECKPOINT ABDUMP command. The operator may make the decision whether to restart IMS or not. After a U0759 ABEND, the message queue data sets need to be reallocated before the restart is done.



## IMS Support Summary

- **What has been added to IMS for ARM?**
  - ARM handling: register, ready, deregister
  - Automatic handling of appropriate restarts
    - AUTO=Y, /ERE OVERRIDE, ...
  - Deregistration when restart would fail
  - Parameter to turn off ARM registration
    - ARMRST=N

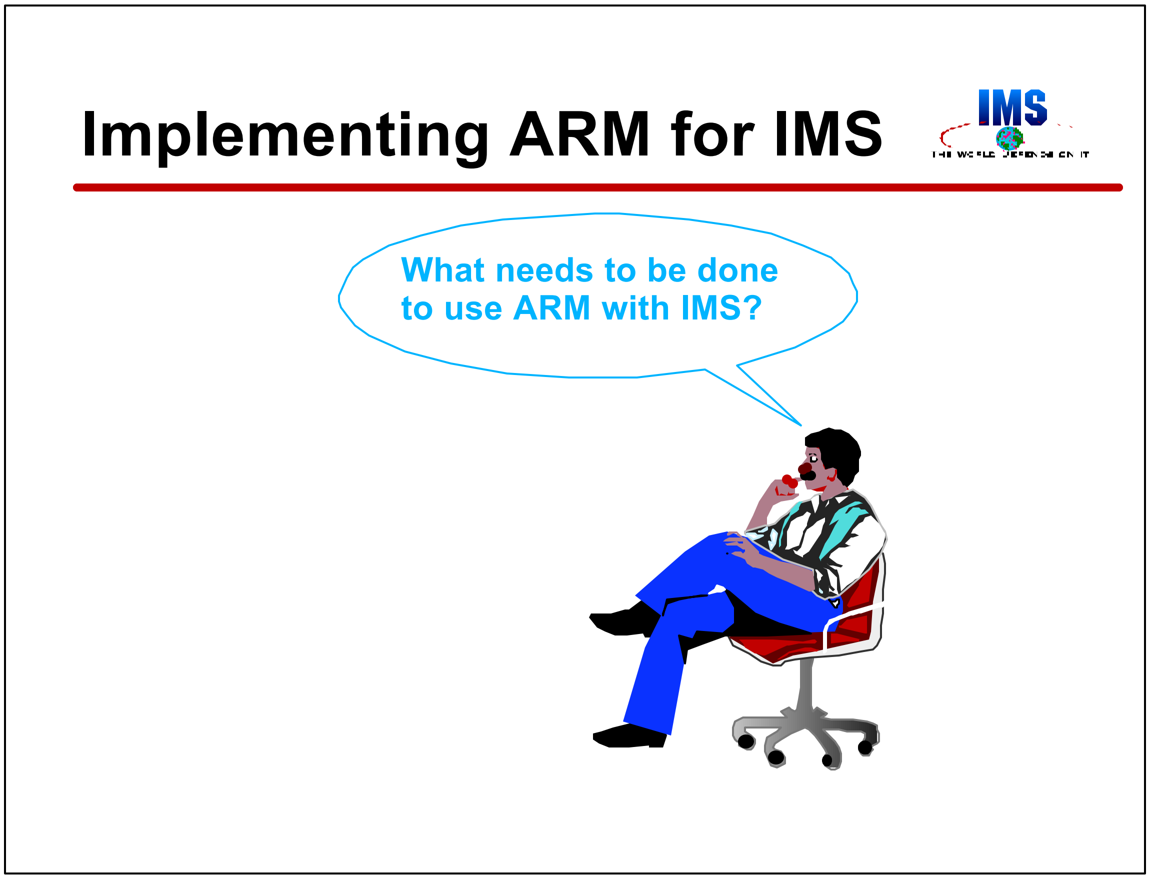
In summary, IMS includes the following ARM support.

IMS online systems issue ARM register, ready, and deregister requests.  Also, XRF alternates and FDBR systems issue associate requests.

When ARM restarts an IMS system, IMS automatically uses AUTO=Y and OVERRIDE when appropriate.

IMS deregisters before ABENDing when a restart would fail.  These include conditions such as full message queues.

IMS provides an execution parameter to disable ARM registration.



## Implementing ARM for IMS

Now, we will look at the implementation of ARM with IMS.

## Implementing ARM for IMS

- ARM restarts IMS by default if an ARM policy is active
  - `SETXCF START,POLICY,TYPE=ARM`
- Ways to turn off restarts:
  - `ARMRST=N` in JCL

  **or**
  - `RESTART_ATTEMPTS(0)` in ARM policy

ARM will restart IMS control regions by default when an ARM policy is active.

Since a user may not want to restart some IMS systems, such as test systems, there is a way to override this default. There are two options. First, a user may specify ARMRST=N in the IMS execution parameters. Second, a user may specify zero restart attempts for certain elements in the ARM policy. Since ARMRST=Y is the default, when ARM is used for other elements in the sysplex, ARM is likely to begin restarting IMS systems. This is only avoided by overtly taking one of these actions.

## What to Restart

- **Typical restart decisions**
  - Production IMS systems
    - ARM restart for ABENDs and system failures
  - Development IMS systems
    - ARM restart for ABENDs
    - Do not ARM restart for system failures
  - Test IMS systems
    - Never ARM restart, unless testing ARM functions

An installation must decide which ARM supported programs it wants to restart. Since most installations have multiple instances of IMS, CICS, and DB2 systems, this is an important decision. Typical decisions are shown here. Production, development, and test systems will be treated differently.

## Restart Groups with IMS

- **DEFAULT group**
  - All elements register in some group
  - If not specified in policy, element is in `DEFAULT` group
- **Reasonable scheme:**
  - Do not restart any element in `DEFAULT` group
    - `RESTART_GROUP(DEFAULT)RESTART_ATTEMPTS(0)`

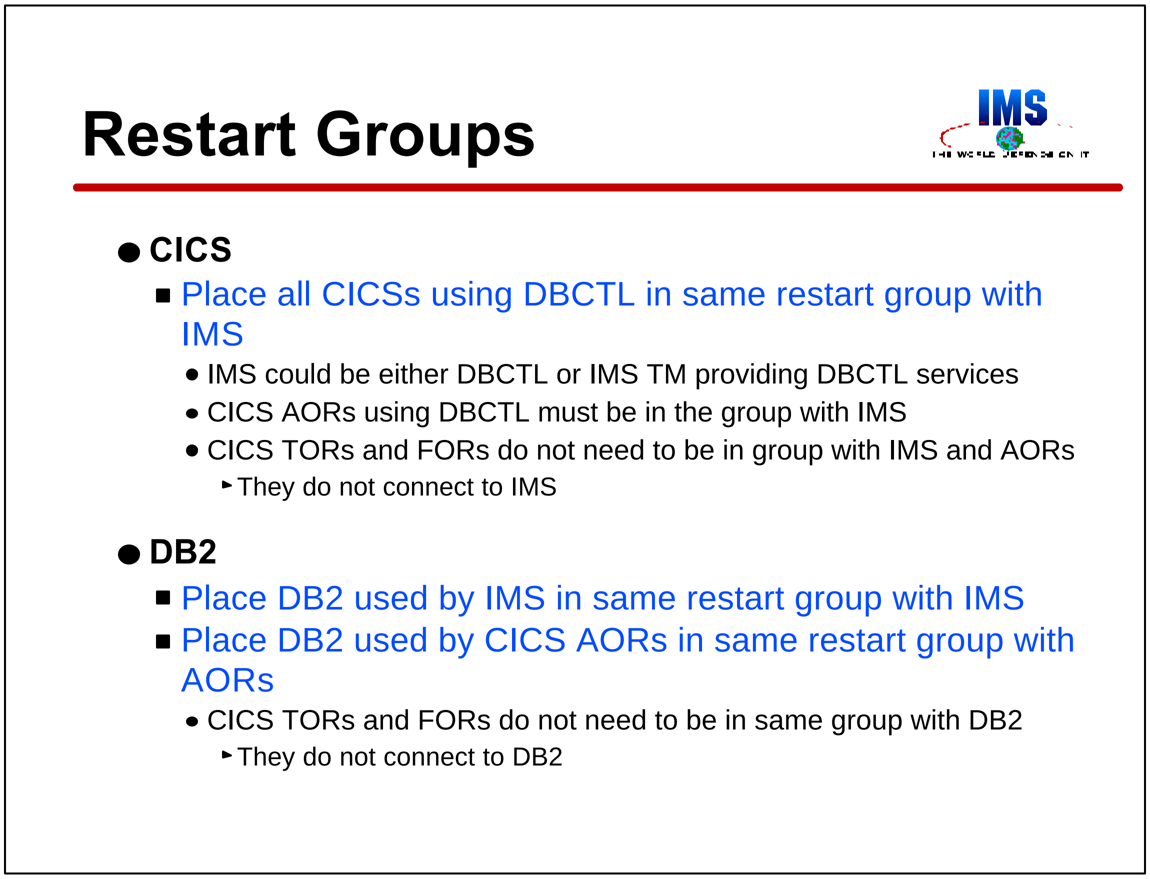
      ➨This prevents accidental restarts of IMS, CICS, DB2, etc.
  - Assign restartable elements to defined groups

    ➨This allows intentional restarts of IMS, CICS, or DB2

Grouping of elements must be done. If not otherwise specified, ARM puts all elements in a DEFAULT group. The default for the DEFAULT group is to restart the elements for both ABENDs and system failures using the PERSIST restart method. This is rarely optimal. It is hard to imagine a system where all ARM elements should be in one group.

Most installations will know which elements should be restarted. They should be defined with appropriate parameters in the ARM policy. Elements which should not be restarted usually include test systems. These may not be so well known. This makes it difficult to specify them explicitly in a policy. It is easier to let them fall in the DEFAULT group and change its default actions. This may be done by explicitly defining this group in the policy. Typically, one would specify zero restart attempts for the elements in the DEFAULT group. This prevents accidental restarts of subsystems that fall in this group.

Disallowing restarts via the policy is especially important for CICS and DB2 users. Neither of these products has an execution parameter to indicate that ARM registration should not be used. This means that all systems, including test, will register with ARM. The ARM policy is the only means to avoid automatic restarts of these CICS and DB2 systems.



## Restart Groups

- **CICS**
  - Place all CICSs using DBCTL in same restart group with IMS
    - IMS could be either DBCTL or IMS TM providing DBCTL services
    - CICS AORs using DBCTL must be in the group with IMS
    - CICS TORs and FORs do not need to be in group with IMS and AORs
      - They do not connect to IMS
- **DB2**
  - Place DB2 used by IMS in same restart group with IMS
  - Place DB2 used by CICS AORs in same restart group with AORs
    - CICS TORs and FORs do not need to be in same group with DB2
      - They do not connect to DB2

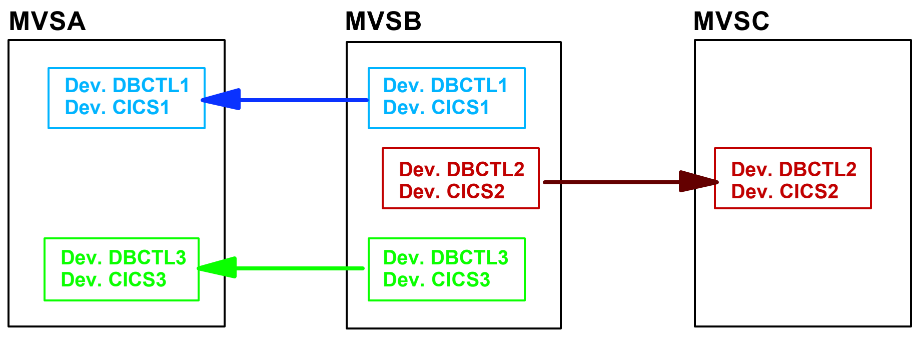
Subsystems which must execute on the same MVS system should be placed in the same group.

All CICSs receiving DBCTL services from an IMS subsystem must be in the same group with the IMS. There could be several such CICS systems; however, only those actually connecting to IMS must be in the group. Application owning regions (AORs) connect to IMS. Terminal owning regions (TORs) and file owning regions (FORs) do not connect to IMS. When CICS MRO uses XCF, it is not necessary for CICS regions connected by MRO to execute on the same MVS with each other.

When IMS uses DB2, they must be in the same group. When this DB2 is also used by CICS, the CICS AORs must also be in the group. Since TORs and FORs do not connect to DB2, they do not have to be in the group.

## Restart Groups (cont.)

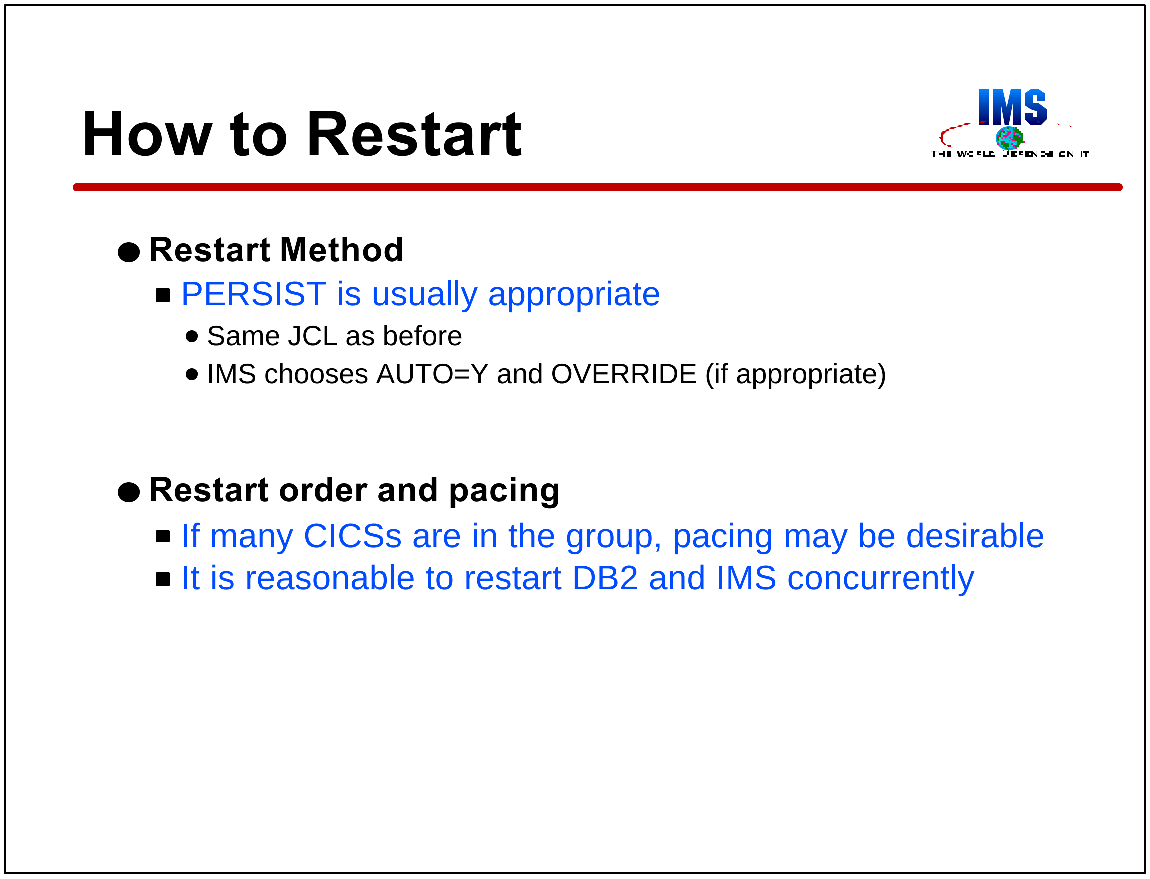
- **Make groups small**
  - Minimize number of elements in a group
  - Do not force ARM to keep multiple subsystems in one MVS unnecessarily
  - If elements can run on separate MVSs, place them in separate groups
  - Control candidate MVSs for restart with `TARGET_SYSTEM` parameter



Groups should be kept to the minimum number of members. If two elements do not need to run on the same MVS, ARM should not be forced to keep them together.

Once a group is determined, the MVS systems on which it may run must be determined. Candidate MVSs for groups are specified in the TARGET_SYSTEM parameter of the ARM policy.

The example shown here contains three groups. Each group contains one DBCTL and one associated CICS. When MVSB fails, the groups are restarted independently. In this case, ARM chose to restart two groups on MVSA and one on MVSC.

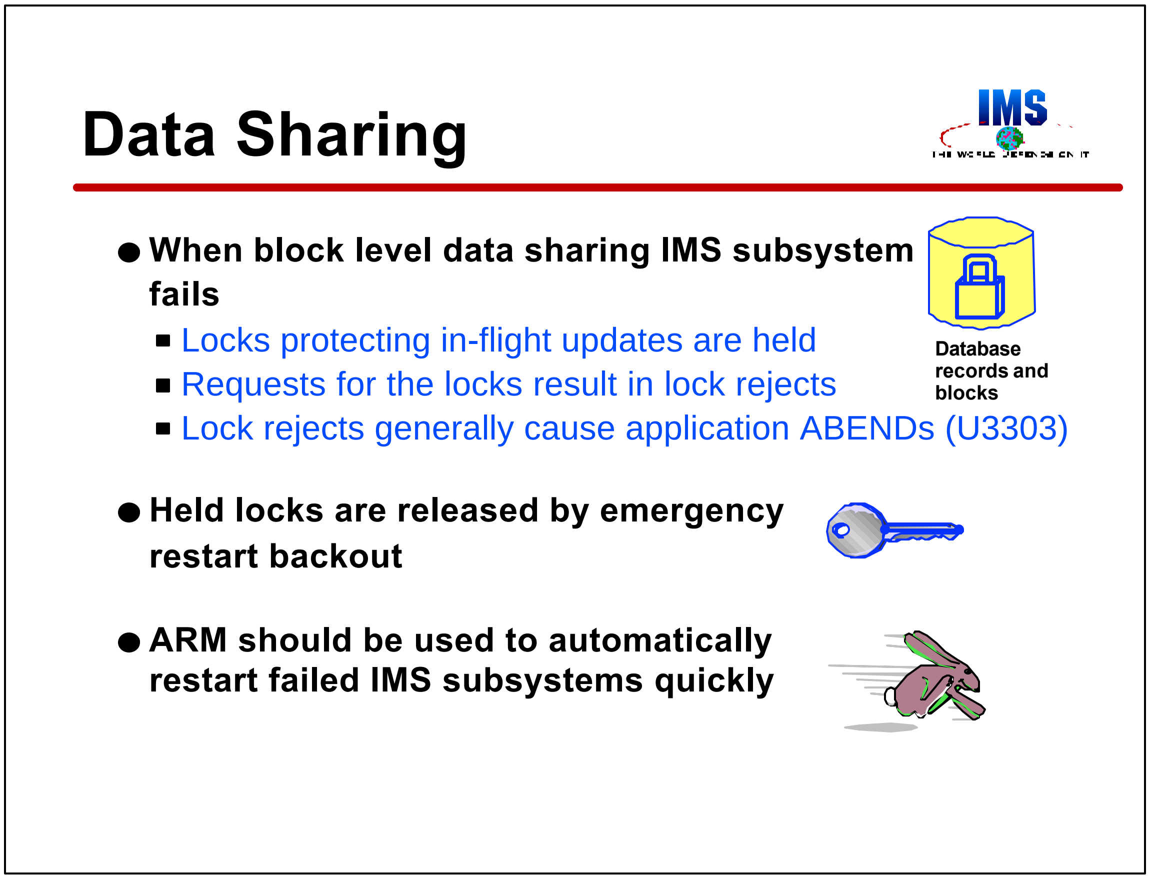


## How to Restart

- **Restart Method**
  - PERSIST is usually appropriate
    - Same JCL as before
    - IMS chooses AUTO=Y and OVERRIDE (if appropriate)
- **Restart order and pacing**
  - If many CICSs are in the group, pacing may be desirable
  - It is reasonable to restart DB2 and IMS concurrently

The method of restart must be selected for each element. Usually, PERSIST is the right choice. This uses the same JCL that was used for the original execution. IMS has special processing to use AUTO=Y for all ARM restarts and OVERRIDE except for XRF. This makes PERSIST usable for most situations.

Restart order and pacing may be chosen. Many CICS users find that pacing is desirable when restarting a large number of CICS regions in a group. Most users of IMS and DB2 find that starting them at the same time is OK.



## Data Sharing

- When block level data sharing IMS subsystem fails
  - Locks protecting in-flight updates are held
  - Requests for the locks result in lock rejects
  - Lock rejects generally cause application ABENDs (U3303)

Database records and blocks

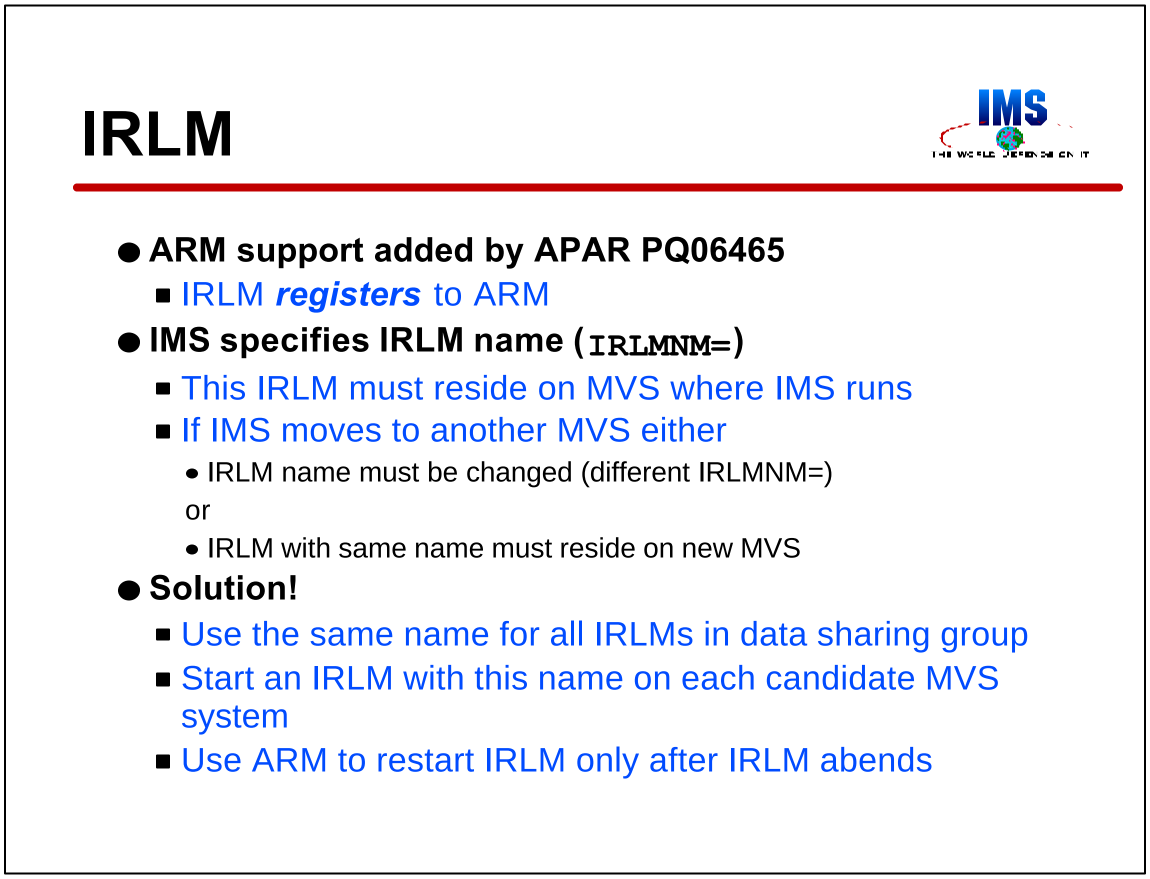
- Held locks are released by emergency restart backout
- ARM should be used to automatically restart failed IMS subsystems quickly

ARM is particularly desirable for users of IMS block level data sharing.

The failure of a data sharing IMS subsystem may effect surviving sharing partners. The locks protecting updates by in-flight units of work in the failed IMS system are not released at failure time. Any transaction or batch job requesting one of these locks will receive a lock reject condition. Generally, this will cause the requester to get a U3303 ABEND. If the requester has issued an INIT STATUS GROUPx call, it will receive a "data unavailable" status code. Since most applications do not issue this INIT call, most users will receive the U3303 ABEND.

The exposure to lock rejects will exist until the locks held by the failed system are released. Emergency restart of the failed IMS system releases these locks when it backs out the in-flight updates or completes the redo processing for committed Fast Path updates. FDBR may also be used to release these locks. FBDR does not restart the failed IMS subsystem or resolve in-doubt units of wirk with DB2 or CICS.

Obviously, the restart of IMS should be done as quickly as possible. ARM may be used to accomplish this.



## IRLM

- **ARM support added by APAR PQ06465**
  - IRLM ***registers*** to ARM
- **IMS specifies IRLM name (`IRLMNM=`)**
  - This IRLM must reside on MVS where IMS runs
  - If IMS moves to another MVS either
    - IRLM name must be changed (different IRLMNM=)

    or
    - IRLM with same name must reside on new MVS
- **Solution!**
  - Use the same name for all IRLMs in data sharing group
  - Start an IRLM with this name on each candidate MVS system
  - Use ARM to restart IRLM only after IRLM abends

IMS block level data sharing requires the use of IRLM.  Users of data sharing need to handle the recovery of IMS systems which use IRLM.

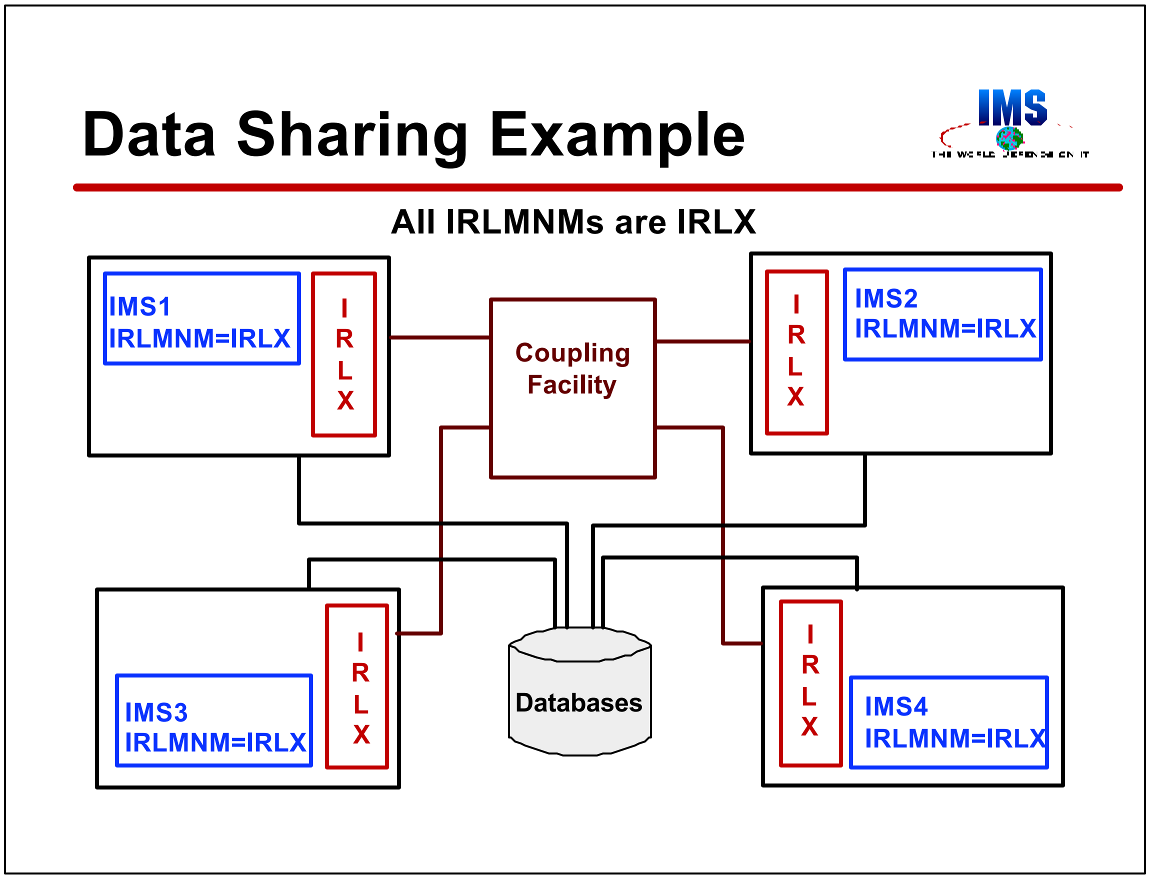
APAR PQ06465 added support for ARM to IRLM 2.1.  If an ARM policy is active, IRLM always registers.  It does not have an execution parameter like IMS's ARMRST to disable the use of ARM.  If you do not want ARM to restart IRLM, you should create a policy specifying not to restart it.  The IRLM element type is SYSIRLM.  For an IRLM with SCOPE=GLOBAL, the element name is a concatenation of the group name, the IRLM subsystem name, and the IRLM ID.  For an IRLM with SCOPE=LOCAL, the element name is a concantenation of the IRLM subsystem name and the IRLM ID.

When installing APAR PQ06465 for the IRLM it may be necessary to install the following APARs associated with the listed products: PQ04116 for DB 4.1, PQ04116 for DB2 5.1, PQ06747 for IMS 4.1, PQ06758 for IMS 5.1,  PQ06759 for IMS 6.1, and OW28526 for MVS 5.2 and above.

When IMS uses IRLM, it must specify the name of the IRLM it will use.  When ARM restarts IMS, IMS will use the same IRLM name as in its previous execution unless the JCL is changed.  Obviously, we want to avoid making a change.

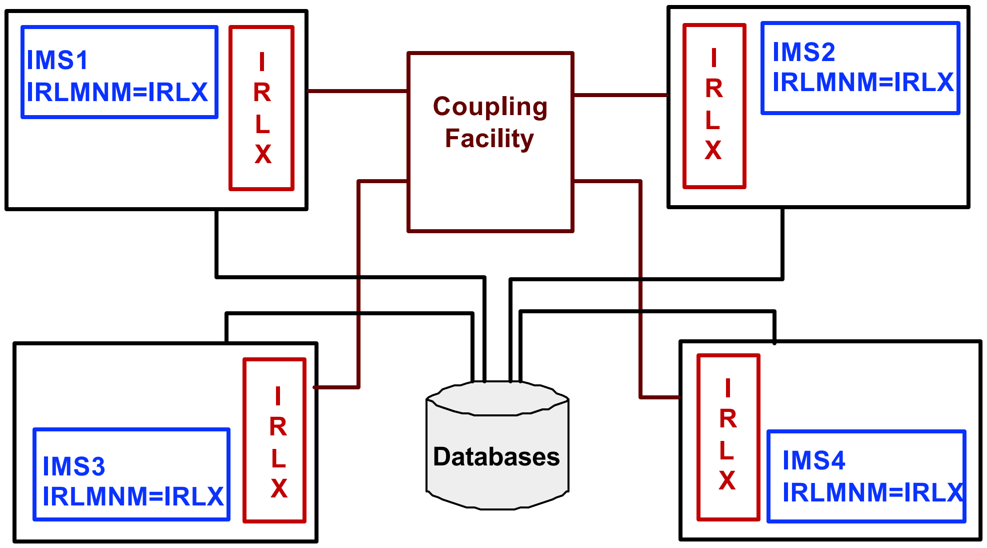
When using block level data sharing, giving all IMS IRLMs the same name is highly recommended.  This allows an IMS to run with any of the IRLMs without changing its JCL.  Each candidate MVS should have an IRLM with this name.  When an IMS is started on any MVS, an IRLM will be available for its use.  IRLM supports concurrent use by multiple IMSs.  That is, two IMSs may use the same IRLM.  This may occur after a system failure.  That is, ARM may restart an IMS on an MVS where another IMS is already running.

So, we don't need ARM to restart the IRLM after system failures.  It can do this; however, it is not required if we already have an IRLM with the right name on each MVS.  We do need ARM to restart the IRLM after IRLM ABENDs.  We should specify TERMTYPE(ELEMTERM)  in our ARM policy to accomplish this.



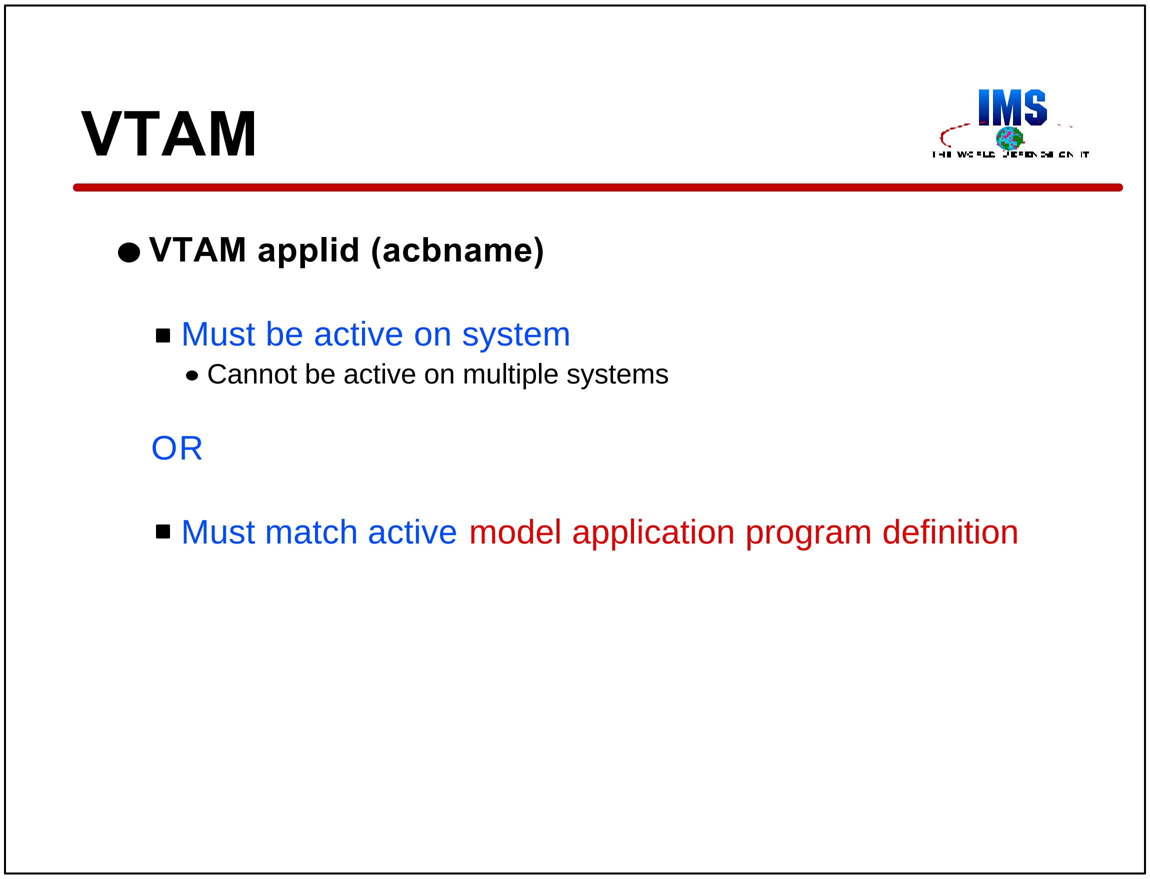
## Data Sharing Example

### All IRLMNMs are IRLX



This is an example of the IRLM configuration suggested on the previous page.

Each MVS system has an IRLM with name IRLX. All IMSs specify that they will use IRLX. If any system fails, its IMS will be restarted on another MVS. That MVS will contain an IRLM with name IRLX.



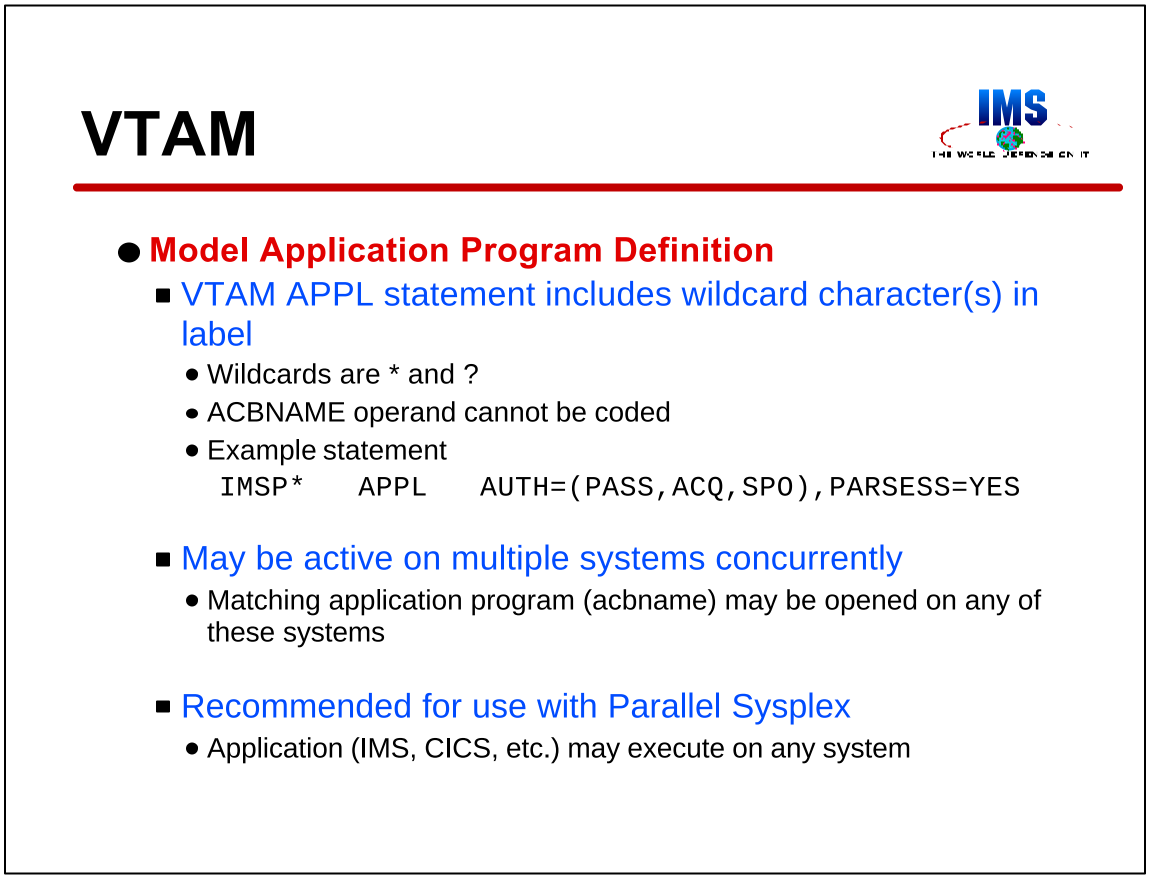
## VTAM

- **VTAM applid (acbname)**
  - Must be active on system
    - Cannot be active on multiple systems

  OR

  - Must match active model application program definition

Since ARM may restart an IMS system on any of the candidate MVS systems, the VTAM on each of those MVSs must be set up to accept IMS. IMS's VTAM applid must either be active on the MVS where IMS executes or must match an active model application program definition. Since an applid cannot be active on more than one VTAM at any time, making it active would have to be done after an IMS failure. Clearly, this is not a practical alternative. Instead, a model application program definition may be used.



## VTAM

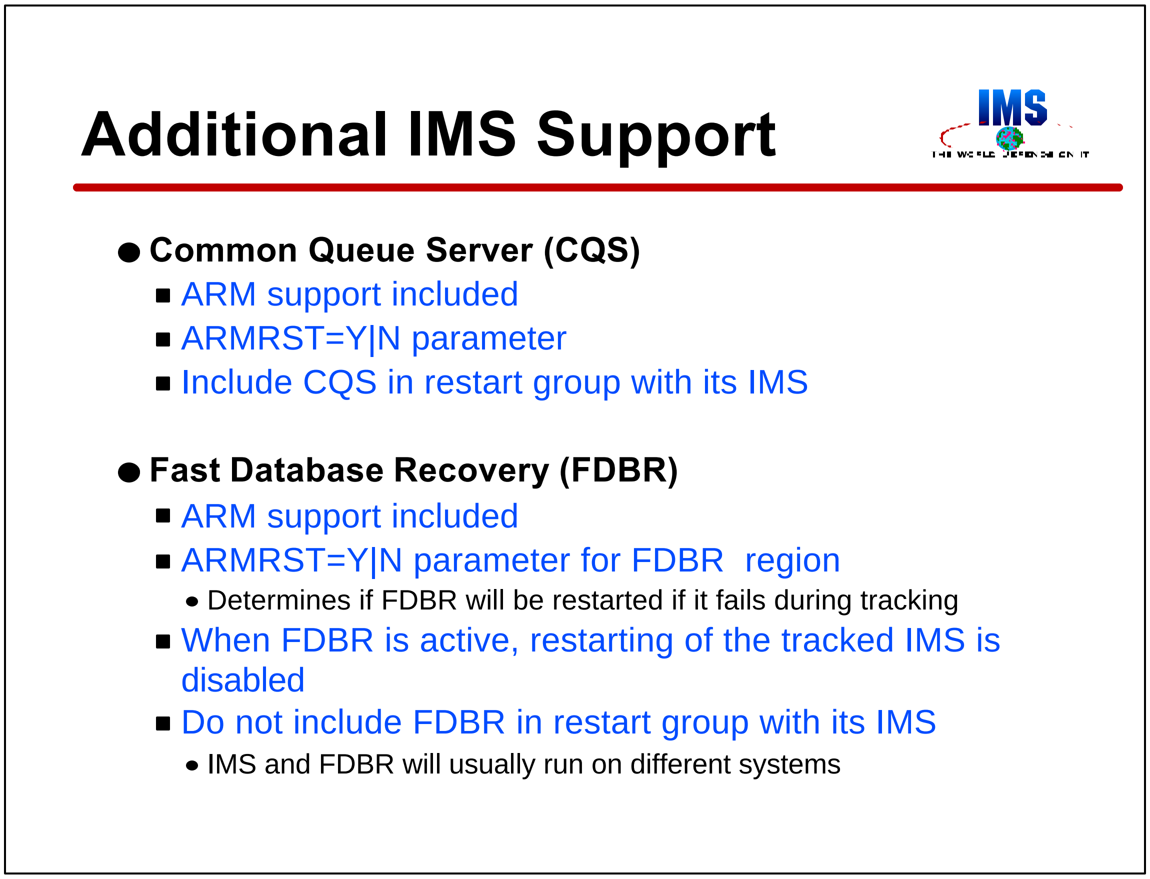
- **Model Application Program Definition**
  - VTAM APPL statement includes wildcard character(s) in label
    - Wildcards are * and ?
    - ACBNAME operand cannot be coded
    - Example statement

```
IMSP*    APPL    AUTH=(PASS,ACQ,SPO),PARSESS=YES
```

  - May be active on multiple systems concurrently
    - Matching application program (acbname) may be opened on any of these systems
  - Recommended for use with Parallel Sysplex
    - Application (IMS, CICS, etc.) may execute on any system

Model application program definitions are created by using wildcard characters in the label on the APPL statement. The ACBNAME operand cannot be used on them. In the example shown here, the model application program definition may be used by any applid beginning with "IMSP". Model application program definitions, unlike applids, may be active on multiple systems concurrently. They have two uses. First, they facilitate cloning of systems in a sysplex. One definition may serve multiple similar VTAM applications. Second, it facilitates the movement of VTAM applications across MVS systems.

Model application program definitions are highly recommended for users of ARM.



## Additional IMS Support

- **Common Queue Server (CQS)**
  - ARM support included
  - ARMRST=Y|N parameter
  - Include CQS in restart group with its IMS
- **Fast Database Recovery (FDBR)**
  - ARM support included
  - ARMRST=Y|N parameter for FDBR region
    - Determines if FDBR will be restarted if it fails during tracking
  - When FDBR is active, restarting of the tracked IMS is disabled
  - Do not include FDBR in restart group with its IMS
    - IMS and FDBR will usually run on different systems

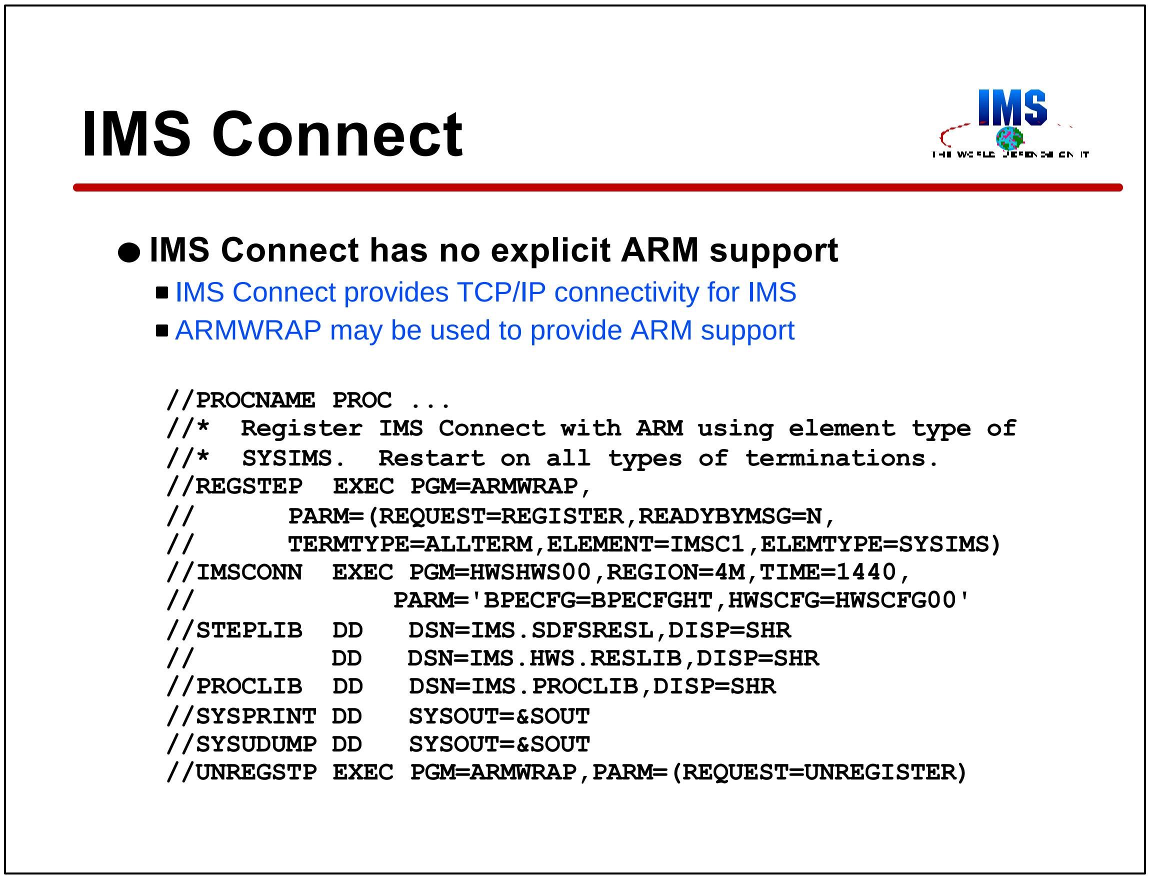
ARM support is included with some additional IMS facilities.

IMS TM shared queues uses the common queue server (CQS). CQS has ARM support similar to that for the IMS control region. It registers with ARM and ARM will restart it after ABENDs or system failures. ARM may be disabled by specifying ARMRST=N in the execution parameters. The default is to use ARM. The element type for CQS is SYSIMS. The element name is the CQSID which is 'CQS' concatenated with the CQS subsystem name.

Since IMS and its CQS system must execute on the same MVS, they must be included in the same restart group.

Fast Database Recovery is a facility used to quickly invoke database recovery operations following the failure of an IMS online system. FDBR tracks the online system by reading its log. When the online system fails, FDBR backs out in-flight full function database updates and reapplies committed but unwritten Fast Path database updates. ARM support for FDBR is provided. The ARMRST parameter is like that for IMS control regions and CQS. It determines if FDBR will register with ARM. If FDBR uses ARM, it will be restarted when it fails. Like an XRF alternate, FDBR issues an associate request when it tracks an IMS system. This disables restarts of the tracked IMS system. The element type for FDBR is SYSIMS. The element name is the IMSID of the FDBR region.

FDBR should not share a restart group with any other elements. It will typically execute on a different MVS system from the IMS which it tracks.



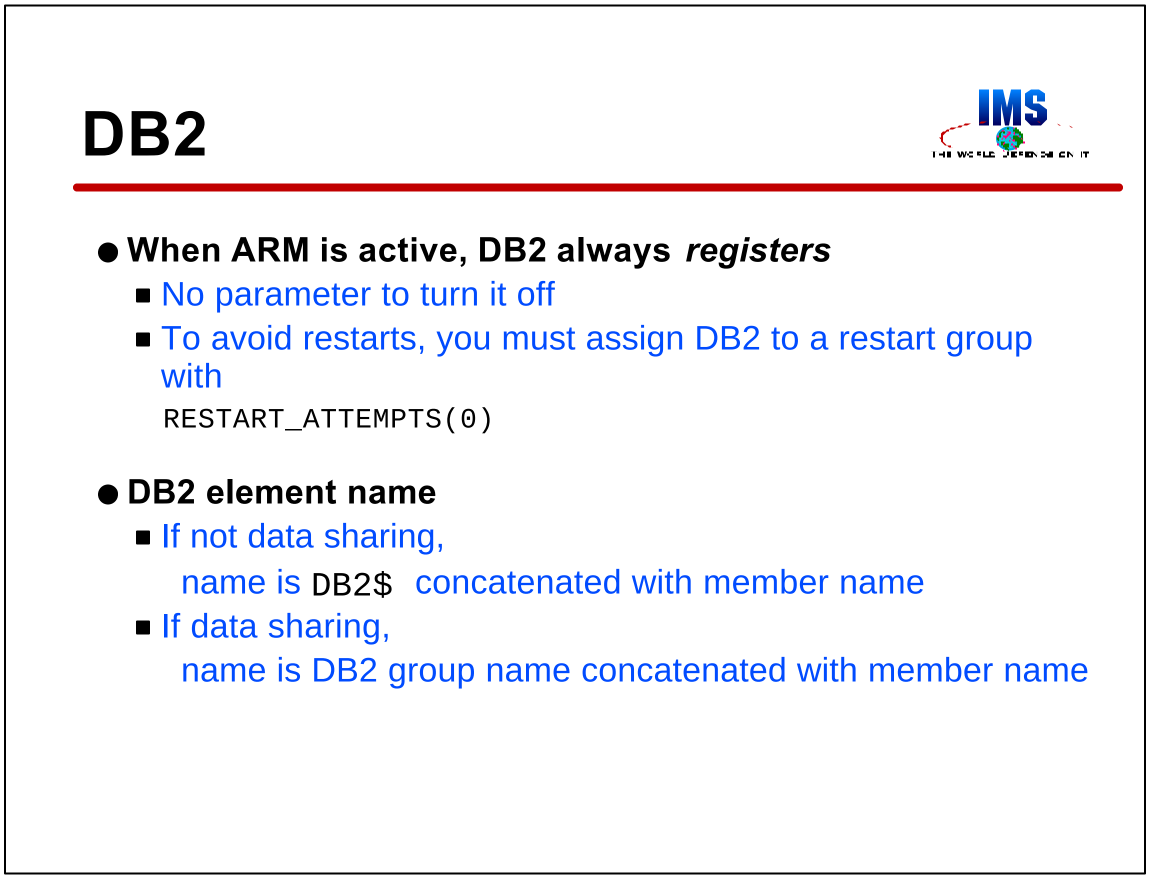
## IMS Connect

- IMS Connect has no explicit ARM support
  - IMS Connect provides TCP/IP connectivity for IMS
  - ARMWRAP may be used to provide ARM support

```
//PROCNAME PROC ...
//*  Register IMS Connect with ARM using element type of
//*  SYSIMS.  Restart on all types of terminations.
//REGSTEP  EXEC PGM=ARMWRAP,
//       PARM=(REQUEST=REGISTER,READYBYMSG=N,
//       TERMTYPE=ALLTERM,ELEMENT=IMSC1,ELEMTYPE=SYSIMS)
//IMSCONN  EXEC PGM=HWSHWS00,REGION=4M,TIME=1440,
//              PARM='BPECFG=BPECFGHT,HWSCFG=HWSCFG00'
//STEPLIB  DD   DSN=IMS.SDFSRESL,DISP=SHR
//         DD   DSN=IMS.HWS.RESLIB,DISP=SHR
//PROCLIB  DD   DSN=IMS.PROCLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=&SOUT
//SYSUDUMP DD   SYSOUT=&SOUT
//UNREGSTP EXEC PGM=ARMWRAP,PARM=(REQUEST=UNREGISTER)
```

IMS Connect (5655-E51) is an IBM product which provides TCP/IP connnectivity for IMS TM systems.  It runs in its own address space.  It connects to TCP/IP clients using sockets and communicates with IMS's  Open Transaction Manager Access (OTMA) interface.  IMS Connect has no explicit ARM support.

ARMWRAP may be used to provide ARM support for an address space where IMS Connect is being run.  This is an example of JCL that might be used.  Whether cross-system restarts (TERMTYPE=ALLTERM) are required, will be determined by the installation's configuration.  If there is already an IMS Connect region on each system, restarts only on ABENDs (TERMTYPE=ELEMTERM) is all that is required.



## DB2

- **When ARM is active, DB2 always *registers***
  - No parameter to turn it off
  - To avoid restarts, you must assign DB2 to a restart group with

    `RESTART_ATTEMPTS(0)`
- **DB2 element name**
  - If not data sharing,

    name is `DB2$` concatenated with member name
  - If data sharing,

    name is DB2 group name concatenated with member name

Unlike IMS, DB2 does not have an option not to register with ARM. DB2 Version 4 and later releases always register. To avoid restarts, a DB2 system must be assigned to a restart group with zero restart attempts.

DB2's element name depends on whether it is a member of a data sharing group. If not data sharing, it is "DB2$" concatenated with the subsystem name. If data sharing, it is the DB2 XCF group name concatenated with the XCF member name.

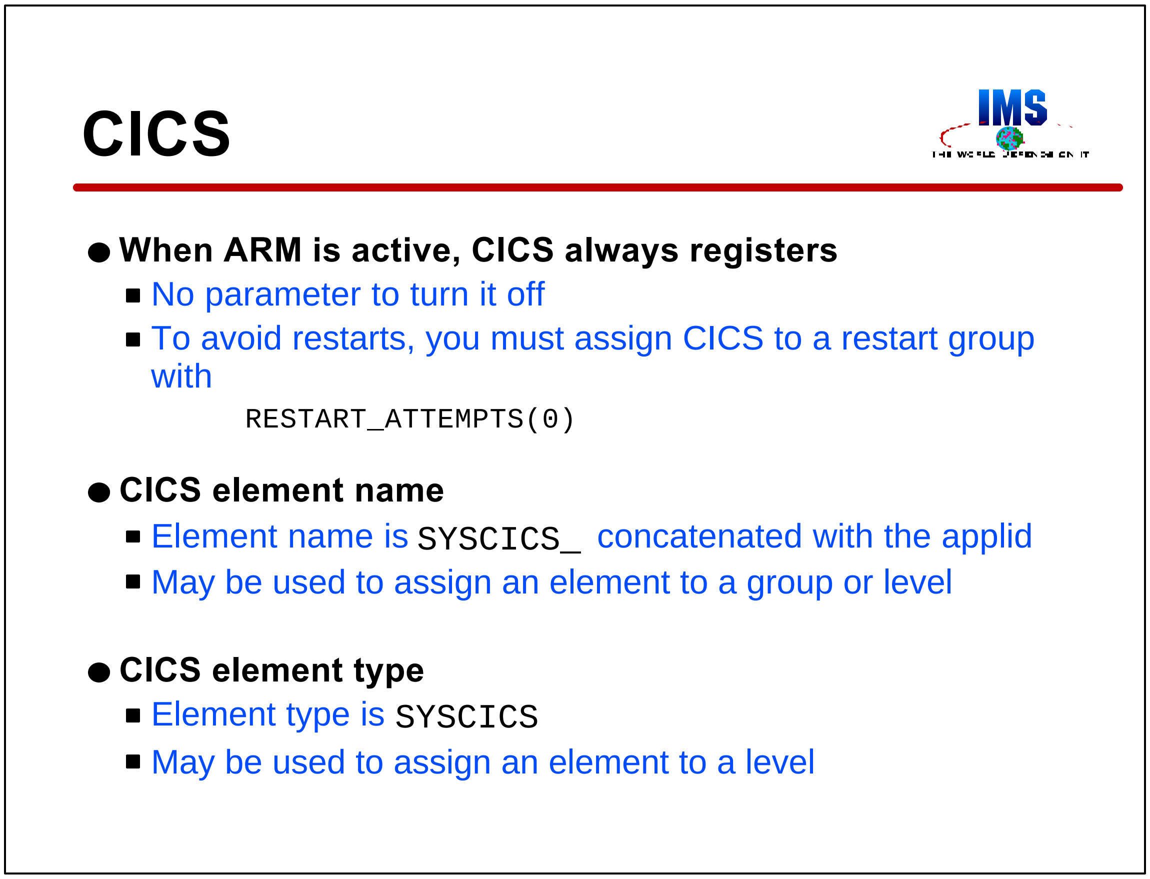
## DB2 (cont.)

- **DB2 element type**
  - Element type is `SYSDB2`
  - Element type may be used to assign an element to a level
- **IRLM with DB2**
  - May use ARM or AUTO START to restart IRLM after system failure
  - Use ARM to restart IRLM after IRLM ABEND

The element type for DB2 is always SYSDB2.

Each DB2 needs its own IRLM. For system failures, either DB2 and its IRLM may be placed in the same restart group, or the AUTO START option of DB2 may be used. This will cause DB2 to start its IRLM. ARM should be used to restart IRLM after IRLM ABENDs.

DB2 provides information on its use of ARM in its *Administration Guide* and its *Data Sharing: Planning and Administration* manual.



## CICS

- **When ARM is active, CICS always registers**
  - No parameter to turn it off
  - To avoid restarts, you must assign CICS to a restart group with

```
        RESTART_ATTEMPTS(0)
```

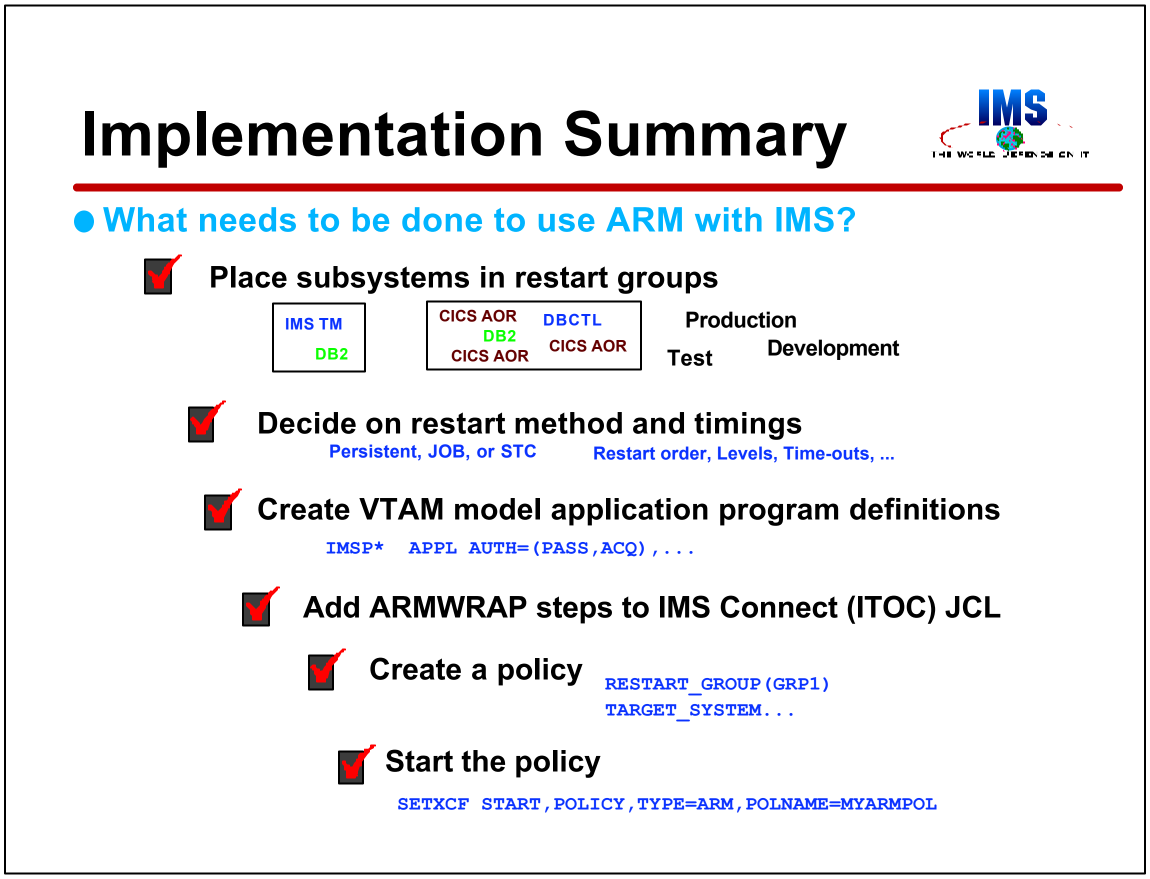
- **CICS element name**
  - Element name is `SYSCICS_` concatenated with the applid
  - May be used to assign an element to a group or level
- **CICS element type**
  - Element type is `SYSCICS`
  - May be used to assign an element to a level

Unlike IMS, CICS does not have an option not to register with ARM. CICS Version 4 and later releases always register. To avoid restarts, a CICS region must be assigned to a restart group with zero restart attempts.

The CICS element name is "SYSCICS_" concatenated with the CICS applid.

The CICS element type is always SYSCICS.

CICS provides information on its use of ARM in the *CICS Recovery and Restart Guide*.



## Implementation Summary

- What needs to be done to use ARM with IMS?
  - ✓ Place subsystems in restart groups

    IMS TM DB2 | CICS AOR DBCTL DB2 CICS AOR CICS AOR | Production Test Development

  - ✓ Decide on restart method and timings

    Persistent, JOB, or STC  Restart order, Levels, Time-outs, ...

  - ✓ Create VTAM model application program definitions

```
IMSP*   APPL  AUTH=(PASS,ACQ),...
```

  - ✓ Add ARMWRAP steps to IMS Connect (ITOC) JCL
  - ✓ Create a policy

```
RESTART_GROUP(GRP1)
TARGET_SYSTEM...
```

  - ✓ Start the policy

```
SETXCF  START,POLICY,TYPE=ARM,POLNAME=MYARMPOL
```

To implement IMS use of ARM, an installation would do the following.

First, restart groups must be selected.  Typically, this would place, at most, one IMS system, one DB2 system, and a collection of CICS AORs in each group.  Of course, only elements that would require cross system restarts would be place in groups.

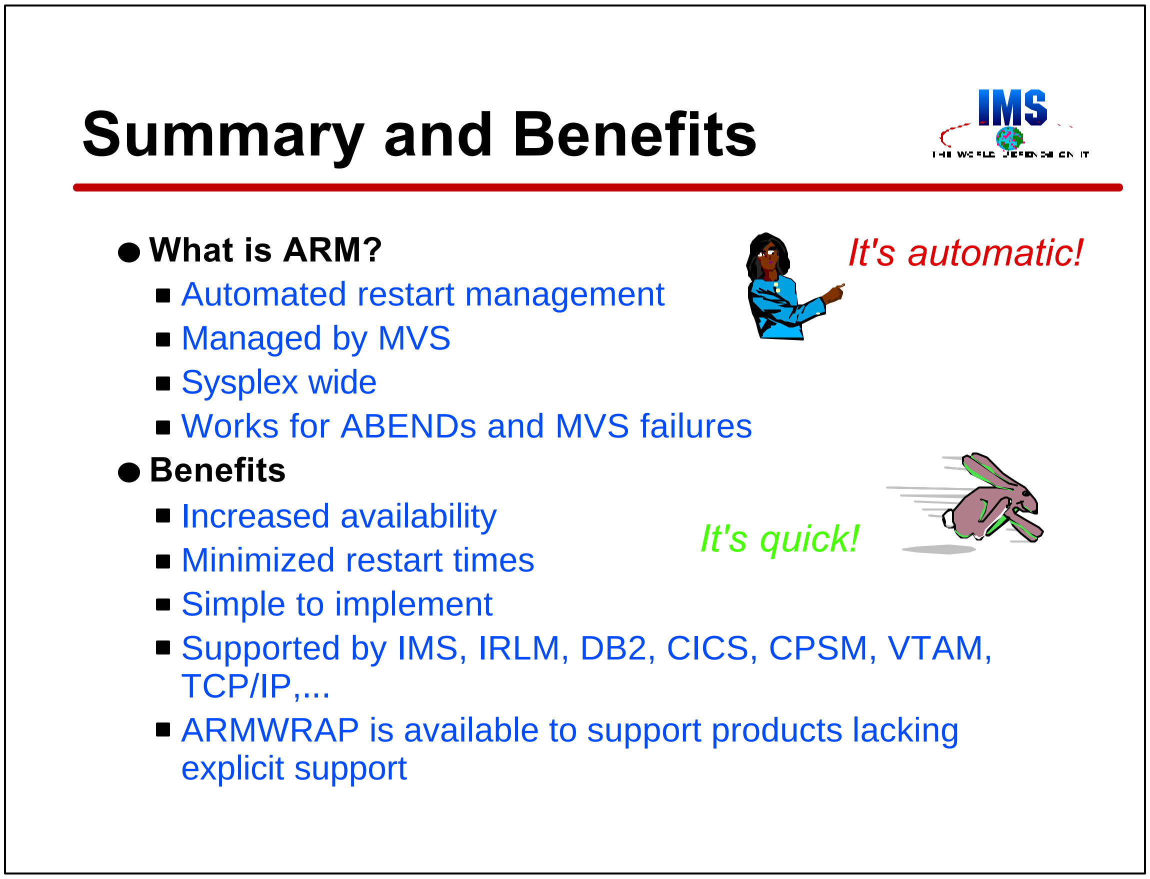
The restart method, restart order, time-out values, and candidate systems must be determined.  If exit routines will be required, they must be designed, coded, and implemented.

VTAM model application program definitions must be created for all  systems using VTAM and which are subject to cross system restarts

If the IMS Connect (IMS TCP/IP OTMA Connector) is used with IMS, restarts of it by ARM may be handled by adding ARMWRAP register and unregister steps to the JCL.

An ARM policy with the appropriate parameters must be defined using the Administrative Data Utility.

Finally, the ARM policy must be started by issuing a SETXCF START,POLICY command

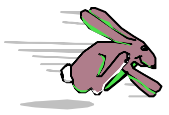


## Summary and Benefits

- **What is ARM?**
  - Automated restart management
  - Managed by MVS
  - Sysplex wide
  - Works for ABENDs and MVS failures
- **Benefits**
  - Increased availability
  - Minimized restart times
  - Simple to implement
  - Supported by IMS, IRLM, DB2, CICS, CPSM, VTAM, TCP/IP,...
  - ARMWRAP is available to support products lacking explicit support

*It's automatic!*

*It's quick!*



The first question we answered was, "What is ARM?"  It is an MVS managed restart capability for started tasks and authorized jobs.  When registered elements ABEND, they are restarted on the same MVS system.  When a system fails, ARM takes advantage of Parallel Sysplex facilities to restart elements on other MVS systems.

The benefits are obvious.  ARM provides increased availability by minimizing restart times.  It is simple to implement.  It typically requires only the writing of simple policy statements.  Support is provided by IMS, IRLM, DB2, CICS, CICSPlex System Manager, VTAM, TCP/IP and many others.  ARMWRAP may be used with those programs which do have explicit ARM support.