* **Resource recovery** is the protection of z/OS resources for update consistency.
  - Resource recovery consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources.
  - z/OS, when requested, can coordinate changes to one or more protected resources, which can be accessed through different resource managers and reside on different systems.
* z/OS ensures that **all changes** are made or **no changes** are made for the following:
  1. Hierarchical databases 2. Relational databases 3. A product-specific resource (i.e. VSAM)
* RRS provides a global syncpoint manager that any ***resource manager (RM)*** on z/OS can exploit.
  - It enables transactions to update protected resources managed by many *resource managers*.

NOTE[1]: A ***resource manager (RM)*** is a subsystem or component such as CICS, IMS, or DB2 which manages resources that can be involved in ***transactions***. Resource managers can be categorized as *work managers*, *data resource manager*s, and *communication resource managers* [such as WAS on z/OS].

* Resource managers must provide the following functions to ensure that the resources they manage can be considered protected resources:
  - *Locking* functions, to ensure that changes to protected resources not yet committed by a transaction can only be seen by that transaction
  - *Logging* functions, to ensure that before and after images of changed resources are held in order to allow the changes to be backed out, or in order for restart processing to reapply changes in the event of system failure

NOTE[2] : A ***transaction*** as consisting of all activity from the start of a task until the initiating program requests that all changes are either committed or backed out.

⚠ This differs from the way many people use the term "**transaction**", where there may be a number of independent commits or backouts made within the one task. In publications you may hear the term "**work request**" when referring to an interaction that may consist of more than one unit of recovery.

* To qualify to be a transaction, a work request must have the following properties:

Atomic A transaction is known as *atomic* when an application changes data in multiple resource managers as a single transaction, and all of those changes are accomplished through a single commit request by a syncpoint manager. If the transaction is successful, all the changes are commited. If any piece of the transation is not successful, then all changes are backed out.

NOTE[3]: See ***atomic instant*** on right side of this page.

Consistent Applications involved in the transaction must be written to a standard that ensures consistency. If the application is correctly designed, then the transaction should maintain a consistent view of data.

Isolated The database managers involved in the transaction isolate the updates to their data so that only the application changing the data knows about the individual updates until the transaction is complete.

Durable Database managers involved in the transaction ensure the data is persistent, both before and after the transaction, regardless of success or failure.

Based on the first letter of each property, this type of transaction is often referred to as an ***ACID*** *transaction*.

* RRS is increasingly becoming a 'prerequisite' for new resource managers, or for new capabilities in existing resource managers rather than having to implement their own ***two-phase commit protocol*** (see right side) these products can use the support provided by RRS. Examples of recent exploiters of RRS are:
  - WLM-managed stored procedures in DB2 requires RRS. Similarly, if you want to have a single unit of recovery between a stored procedure and any resource manager *other* than DB2, you must use RRS.
  - WebSphere Application Server for z/OS: in this case, RRS is *always* required; however, its use is hidden from the application developer.
  - If you plan to use the Transactional EXCI interface provided by CICS Transaction Server , RRS *must* be started in the system.
  - APPC/MVS (appl. peer-to-peer communication / LU6.2) requires RRS when using protected conversations.
  - OTMA and ODBA support in IMS® requires RRS to be enabled when Synclevel Syncpt is specified for IMS Connect.
  - Transactional extensions to VSAM record level sharing (RLS) exploit RRS to allow batch jobs include VSAM files in the scope of an ACID transaction.
* To understand z/OS' use of resource recovery, you need to understand both the programs that work together and something of how they work together by firrst knowing what an ***exit manager*** is.
  - An **exit manager** is an authorized program that controls the flow of a predefined set of events. When a predefined event occurs, the exit manager gives control to an ***exit routine*** owned by a program interested in the event. In this exit routine, the program provides the processing for the event. z/OS provides *two exit managers*: **resource recovery services (RRS)** and **context services**.
  - The following three programs work together to protect resources:
    1. **Application program**: The application program accesses protected resources and requests changes to the resources.
    2. **Resource Manager**: A resource manager controls and manages access to a resource. A resource manager is an authorized program that provides an application programming interface (API) allowing the application program to read and change a protected resource. The resource manager, through exit routines that get control in response to events, takes actions that commit or back out changes to a resource it manages. Often an application changes more than one protected resource, so that more than one resource manager is involved. **A resource manager may be an IBM® product, part of an IBM product, or a product from another vendor.**

NOTE[4]: The resource manager in resource recovery is different from an RTM‡ resource manager, which is related to the operating system's recovery termination management (RTM) and runs during system termination processing.

    3. **Syncpoint manager**: Resource recovery services (RRS) is the syncpoint manager. It uses a two-phase commit protocol, to coordinate changes to protected resources, so that all changes are made or no changes are made. During its processing, RRS drives resource manager exit routines. For example, if a commit event occurs (such as when an application requests changes be made to several resources), RRS drives the commit exit routine for each resource manager involved.
  - The responsibility for coding the exits that RRS requires lies with the developer of the resource manager (i.e. IBM).

NOTE[5]:This is not a concern for application programmers writing client applications for use with resource managers that implement RRS support, since RRS is largely "hidden" from the application programmer's view.

* Technically, by RRS we really mean Recoverable Resource Management Services (**RRMS**).
  - RRMS consists of **three services**:
    1. Registration services **2.** Context services **3.** Resource Recovery Services (RRS).
  - RRMS provides a systems programming interface (SPI) that enables a resource manager:
    > To *register* with the system *as* a resource manager; To *express interest* in work requests that access its resources; To *take part* in resource recovery for those work requests.

[The next section describes registration services and context services. It is RRS that provides the means to implement two-phase commit, but a resource manager must use registration services and context services in conjunction with Resource Recovery Services.]

In the past, a restriction prohibited a resource manager from restarting on a different system if the resource manager had outstanding transactions and RRS remained active through the failure. This restrictions impacted a system's **peer-level recovery** capability in the system because of an earlier inability of RRS to coordinate transactions across multiple systems internally. As a result, RRS required all of the resource managers involved in a common set of transactions to be restarted on a single system. These restrictions hampered recovery in certain types of failures, and prevented you from restoring a resource manager to the preferred system after recovering from a system outage. Starting with z/OS V1R6, IBM removed the restriction that required a failed resource manager to restart on the same system. The feature is called **RRS restart anytime/anywhere**. Resource managers can restart on any z/OS in the same Resource Recovery Services (RRS) **logging group**, regardless of whether RRS remains active across the failure.

See Slide #2 –bottom right

In **2PC** there is always a co-ordinator (C) RM and one or more participant (P) RMs, such as a UR that spans CICS (**C**) and DB2 (**P**), CICS acts as the **Syncpoint Mgr**.

* **Registration services:** The first thing a resource manager must do in order to start using RRS is to identify itself to RRS and supply RRS with its exit routines and to do so, a resource manager uses registration services.
  - Depending on what resources a resource manager protects and what its role is, a resource manager will call the Set_Exit_Information service and provides parms for each exit it wishes to enable.

NOTE[6]: A resource manager calls the Set_Exit_Information service to notify a specific exit manager of its intent to work with that exit manager, and to identify the entry points for the exit routines, if any, to be driven by that specific exit manager. Supported exit managers include: Context services, Resource recovery services/MVS (RRS) and in response to the call, the system returns a return-code.

  - A resource manager usually supplies exits for actions like PREPARE, COMMIT and BACKOUT, and would use the registration service to notify RRS what exits are to be driven in the event of certain conditions occurring.
* **Context services**: Is an exit manager that allows resource managers to "track" the changes made by a given unit of work.
  - It provides the data constructs and primitives that resource managers can use as an anchor to relate a unit of work to an application.
  - A *context* represents the environment that the application is running in and its work request.
  - A work manager can create a context and have its application run under it.
  - A work manager can actually create more than one context, but at any point of time there can only be one active context for a single task. If the work manager need to change the context, it has to explicitly switch to a different context
  - Associated to the context is the unit of recovery that is currently running with all the information about the resource managers involved in this unit of work.
  - A context may (and usually does) span several units of work.
    > In fact, after you commit a unit of work (UR), you can start another unit of work under the same context.
  - A context consists of the application program requesting the work, and the protected resources involved in the work.
  - When an application program requests access to a resource, the resource manager might express an interest in the context associated with the application program and its work request.
  - A resource manager that exploits RRS might need to express interest in a work context, not just a particular UR, because a context can persist over multiple URs.
* The ***two-phase commit (2PC)* protocol** is a set of actions used to ensure an application program either makes ***all*** changes to the resources represented by a single unit of recovery (UR), or makes ***no*** changes at all.
  - This protocol verifies that either all changes or no changes are applied even if one of the elements (such as the application, the system, or the resource manager) fails.
  - The protocol allows for restart and recovery processing to take place after system or subsystem failure.
  - The two-phase commit protocol is initiated when the application is ready to commit or back out its changes.
  - The program that initiates the commit or backout does not need to know who the syncpoint manager is or how two-phase commit works and is hidden from the application;
    > A program simply calls a commit or backout service and receives a return code indicating whether this has been successfully completed.
* When the application issues the commit request, the ***coordinating recovery manager***, also called the ***syncpoint manager***, gives each resource manager participating in the unit of recovery an opportunity to **"vote"** on whether its part of the UR is in a consistent state and can be committed.
  - If all participants vote **YES**, the syncpoint manager instructs all the resource managers to commit the changes.
  - If *any* vote **NO**, the syncpoint manager instructs them to back out the changes.
  - This process is usually represented as *two phases* - In ***phase 1***, the application program issues the syncpoint or backout request to the syncpoint manager.
  - The coordinator issues a ***PREPARE*** command to send the initial syncpoint flow to *all* UR agent resource managers.
  - In response to the PREPARE command, each resource manager involved in the transaction replies to the syncpoint manager, stating whether it is ready to commit.
  - After the syncpoint manager receives *all* the responses back from all of its agents, ***phase 2*** is initiated.
  - In this phase, the syncpoint manager issues the ***COMMIT*** or ***ROllBACK*** command based on the previous responses.
  - If any of the agents came back with a negative response, the syncpoint initiator tells *all* syncpoint agents to roll back their changes.
  - If a resource manager (RM) fails before the syncpoint manager issues the PREPARE command, the syncpoint manager assumes a backout is needed and the transaction will be backed out by all RMs; if an RM fails after responding to the prepare but before receiving the commit decision (during the in-doubt stage), then all other RMs commit and the failed RM must contact RRS on restart to determine the correct outcome for the UR.
* The moment when the coordinator records that it is going to tell all the resource managers to either commit or roll back is known as the ***atomic instant***.
  - Regardless of any failures after that time, the coordinator assumes that all changes are either committed or rolled back.
  - A syncpoint manager usually logs the decision at this point.
  - If any of the participants abend after the atomic instant, the abending resource manager must work with the syncpoint manager when it restarts to complete any commits or rollbacks that were in process at the time of the abend.
  - During the first phase of the protocol, the agents do not know whether the syncpoint manager will commit or roll back the changes until the syncpoint manager has collected all responses.
    > This time is known as the ***in-doubt period***.
* A context can be either a **native context** or a **privately-managed context**, as explained here:
  - **Native context:** The automatically occurring context of the application program and protected resources associated with a work request. A native context is associated with a single application task. This context always exists and cannot be detached from the task and associated with any other task. A native context is implicitly created by the operating system.
  - **Privately-managed context:** Created by a resource manager. The resource manager (for example CICS, IMS or WebSphere Application Server) owns the privately-managed context it creates, and the resource manager can switch a privately-managed context from one task to another. A privately-managed context is usually used by a resource manager like IMS/TM, that accepts and manages work like transactions from outside the system. The privately-managed context acts as an anchor for the inbound application request. It can associate a transactional scope for the context, such that any resources that are actively manipulated under that context are in the same transactional scope. A native context may never be switched to another task. A privately managed context can be switched to any number of tasks.

See RRS Panel examples on Slide #2

‡ RTM can be called to perform its recovery and termination services on behalf of the caller or on behalf of an assembler routine using two authorized macros -- CALLRTM and ABEND -- invoke RTM.

#57 zTidBits

RRS Panel Display and Examples

Slide 2 of 2

Session A - [24 x 80]

```
Display  Filter  View  Print  Options  Help
--------------------------------------------------------------------------------
SDSF DA MVSA     DEMOMVS  PAG  0  CPU/L/Z   6/  6/  0  LINE 261-278 (314)
COMMAND INPUT ===> _                                          SCROLL ===> PAGE
PREFIX=*  DEST=(ALL)  OWNER=*  SORT=JOBNAME/A  SYSNAME=
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO
     RCCRUNU2 RCCRUNU2 RCC7100 STC16283 SYSSTC     LO  FF  554   0.00   0.00
     RD4ZJMON RD4ZJMON JMON    STC16275 SYSSTC     LO  FF  811   0.00   0.00
     RD4ZLOCK RD4ZLOCK LOCKD   STC16273 SYSSTC     IN  F2 8310   0.00   0.00
     RD4ZRSE  RD4ZRSE  RSED    STC16272 SYSSTC     IN  F2 9117   0.00   0.00
     RD4ZRSE8 STEP1            STC16144 SYSSTC     IN  F4 111T   0.00   0.00
     REAGNT1  REAGNT1  ARYSPAGT STC16248 SYSSTC     LO  FF  14T   0.00   0.00
     RESERV1  RESERV1  ARYSPSRV STC16247 SYSSTC     IN  F2  11T   0.53   0.00
     RESOLVER RESOLVER IEFPROC                      NS  FE 4324   0.00   0.00
     RMF      RMF      IEFPROC  STC03389 SYSSTC     NS  FE  12T   0.00   0.00
     RMFGAT   RMFGAT   IEFPROC  STC03390 SYSSTC     NS  FE 4687   0.00   0.00
     RODM     RODM     RODM     STC16343 SYSSTC     NS  F2 4022   0.00   0.00
     RRS      RRS      RRS                          NS  F6 3158   0.00   0.00
     SCLMSRV  SCLMSRV  CQSCLM   STC16280 WEBSRV     IN  F2 3678   0.00   0.00
     SDSF     SDSF     SDSF     STC16060 SYSSTC     NS  F2 1153   0.00   0.00
     SMF      SMF      IEFPROC                      NS  FF  32T   0.00   0.00
     SMS      SMS      IEFPROC                      NS  FE  718   0.00   6.59
     SMSPDSE  SMSPDSE                               NS  FF 1142   0.00   0.00
     SMSPDSE1 SMSPDSE1                              NS  FF  26T   0.00   0.00
MA   a                                                          04/021
```

DA "display active"

Session B - [32 x 80]

RRS Options Panel

```
                                  RRS
Option ===> _

Select an option and press ENTER:

1 Browse an RRS log stream
2 Display/Update RRS related Resource Manager information
3 Display/Update RRS Unit of Recovery information
4 Display/Update RRS related Work Manager information
5 Display/Update RRS UR selection criteria profiles
6 Display RRS-related system information

F1=Help    F2=Split   F3=Exit    F9=Swap   F12=Cancel
MA   b                                                          02/014
```

Connected to remote server/host 9.82.24.70 using lu/pool TCPB0001 and port 23

```
                    RRS Log Stream Browse Selection
Command ===>

Provide selection criteria and press Enter:

Select a log stream to view:                Level of report detail:
2  1.  RRS Archive log                      1  1.  Summary
   2.  RRS Unit of Recovery State logs         2.  Detailed
   3.  RRS Restart log
   4.  RRS Resource Manager Data log
   5.  RRS Resource Manager MetaData log

RRS Group Name . . . .          Default Group Name:  WSCPLEX
Output data set  . . ATR.REPORT

Optional filtering:
 Entries from . . . .              local date in yyyy/mm/dd format
                                  local time in hh:mm:ss format
 through . . . . . .               local date in yyyy/mm/dd format
                                  local time in hh:mm:ss format
 UR identifier . . .                                  (Options 1,2,3)
 RM name . . . . . .                                  (Option 4,5)
 SURID                                                (Options 1,2,3)
```

```
                    RRS Resource Manager List                Row 1 to 23 of 139
Command ===> _                                               Scroll ===> PAGE

Commands: v-View Details u-View URs r-Remove Interest d-Delete RM
          n-Unregister RM
S  RM Name                          State      System   Logging Group
   BBO.AWBASEB.AWSR1.AWSR1B.IBM     Reset      SYSB     WSCPLEX
   BBO.AWCELL.AWSR2.AWSR2B.IBM      Reset      SYSB     WSCPLEX
   BBO.AWCELL.BBOCOD1.BBOSOD1.IBM   Reset      SYSB     WSCPLEX
   BBO.BTBASE.BTSRV2.BTSRV2B.IBM    Reset      SYSB     WSCPLEX
   BBO.BTCELL.BTSRV2.BTSRV2B.IBM    Reset      SYSB     WSCPLEX
   BBO.B1BASEB.B1SR01.B1SR01B.IBM   Reset      SYSB     WSCPLEX
   BBO.B6BASEB.B6SR01.B6SR01B.IBM   Reset      SYSB     WSCPLEX
   BBO.B6CELL.B6AGNTB.B6AGNTB.IBM   Reset      SYSB     WSCPLEX
   BBO.B6CELL.B6SR01.B6SR01B.IBM    Reset      SYSB     WSCPLEX
   BBO.B6CELL.B6SR02.B6SR02B.IBM    Reset      SYSB     WSCPLEX
   BBO.B6CELLNDB6AGNTB.B6AGNTB.IBM  Reset      SYSB     WSCPLEX
   BBO.B6CELLNDB6DMGR.B6DMGR.IBM    Reset      SYSB     WSCPLEX
   BBO.B6CELLNDB6SR01.B6SR01B.IBM   Reset      SYSB     WSCPLEX
   BBO.B6CELLNDB6SR02.BBOSOD1.IBM   Reset      SYSB     WSCPLEX
   BBO.B6CELLNDB6SR02.B6SR02B.IBM   Reset      SYSB     WSCPLEX
   BBO.C6CELL.C6SR01.C6SR01B.IBM    Reset      SYSB     WSCPLEX
   BBO.C6CELLB.C6SR01.C6SR01B.IBM   Reset      SYSB     WSCPLEX
   BBO.F6CELL.F6AGNTB.F6AGNTB.IBM   Reset      SYSB     WSCPLEX
   BBO.F6CELL.F6SR01.F6SR01B.IBM    Reset      SYSB     WSCPLEX
   BBO.F6CELLB.F6SR01.F6SR01B.IBM   Reset      SYSB     WSCPLEX
   BBO.G6CELL.BTSRV2.BTSRV2B.IBM    Reset      SYSB     WSCPLEX
   BBO.G6CELL.G6AGNTB.G6AGNTB.IBM   Reset      SYSB     WSCPLEX
   BBO.G6CELL.G6SR01C.BBOSOD1.IBM   Reset      SYSB     WSCPLEX
F1=Help    F2=Split   F3=Exit    F5=ListErr  F6=Refresh  F7=Up
F8=Down    F9=Swap    F12=Cancel
```

Note: WAS on z/OS

```
                    RRS System Information List              Row 1 to 4 of 4
Command ===> _                                               Scroll ===> PAGE

        System Name          Logging Group Name
        SYSA                 WSCPLEX
        SYSB                 WSCPLEX
        SYSC                 WSCPLEX
        SYSD                 WSCPLEX
********************************* Bottom of data *********************************
```

```
Menu  Utilities  Compilers  Help

BROWSE    KETTNER.ATR.REPORT                        Line 00000000 Col 001 080
Command ===> _                                                Scroll ===> PAGE
********************************* Top of Data ************************************
RRS/MVS LOG STREAM BROWSE SUMMARY REPORT

READING ATR.WSCPLEX.RESTART        LOG STREAM
ATR053I ATR.WSCPLEX.RESTART        IS EMPTY,
        RC=00000008 RSN=00000846
********************************* Bottom of Data *********************************
```

Session B - [32 x 80]

```
Menu  Utilities  Compilers  Help

BROWSE    KETTNER.ATR.REPORT                        Line 00000000 Col 001 080
Command ===> _                                                Scroll ===> PAGE
********************************* Top of Data ************************************
RRS/MVS LOG STREAM BROWSE DETAIL  REPORT

READING ATR.WSCPLEX.RM.DATA        LOG STREAM

SYSB     2009/12/26 12:30:34.942402 BLOCKID=00000000563942E9
 RESOURCE MANAGER=BBO.H2CELL.H2DMGR.H2DMGR.IBM     LOGGING SYSTEM=SYSB
   RESOURCE MANAGER MUST RESTART ON SYSTEM SYSB
   RESOURCE MANAGER WAS LAST ACTIVE WITH RRS ON SYSTEM SYSB
   LOG NAME IS BBOTEN0.0000003C.IBM
   RESTART ANYTIME SUPPORTED
   LOG INSTANCE NUMBER: 2009/12/21 16:42:19.609668

SYSC     2009/12/26 12:30:56.989400 BLOCKID=00000000563943C1
 RESOURCE MANAGER=BBO.H7CELL.H7AGNTA.H7AGNTA.IBM   LOGGING SYSTEM=SYSC
   RESOURCE MANAGER MUST RESTART ON SYSTEM SYSA
   RESOURCE MANAGER WAS LAST ACTIVE WITH RRS ON SYSTEM SYSA
   LOG NAME IS BBOTEN0.0000000D.IBM
   RESTART ANYTIME SUPPORTED
   LOG INSTANCE NUMBER: 2009/12/21 16:42:23.632855

SYSC     2009/12/26 12:30:56.991465 BLOCKID=0000000056394499
 RESOURCE MANAGER=BBO.H2CELL.H2SR02.H2SR02B.IBM    LOGGING SYSTEM=SYSC
   RESOURCE MANAGER MAY RESTART ON ANY SYSTEM
   RESOURCE MANAGER WAS LAST ACTIVE WITH RRS ON SYSTEM SYSB
   LOG NAME IS BBOTEN0.00000015.IBM
   F1=Help     F2=Split    F3=Exit     F4=Return   F5=Rfind    F7=Up
   F8=Down     F9=Swap     F10=Left    F11=Right   F12=CRetriev
MA   b                                                          04/015
```

Note: Restart Capability