E21

z/OS RRS Exploitation for IMS and the IMS e-business Connectors

Jack Yuan

IMS Technical Conference

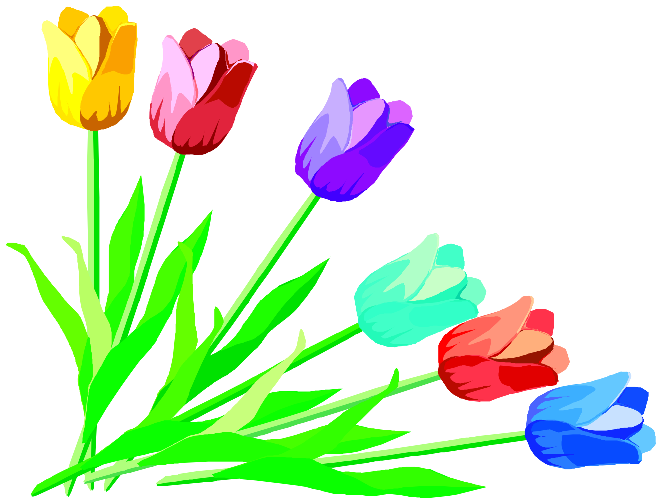
St. Louis, MO **Sept. 30 - Oct. 3, 2002**

RETURN TO INDEX

e-business

Special Thanks

To IMS restart/RRS expert, Jerry Zentner, who helped and encouraged me when it was really needed. I thank him from the bottom of my heart.

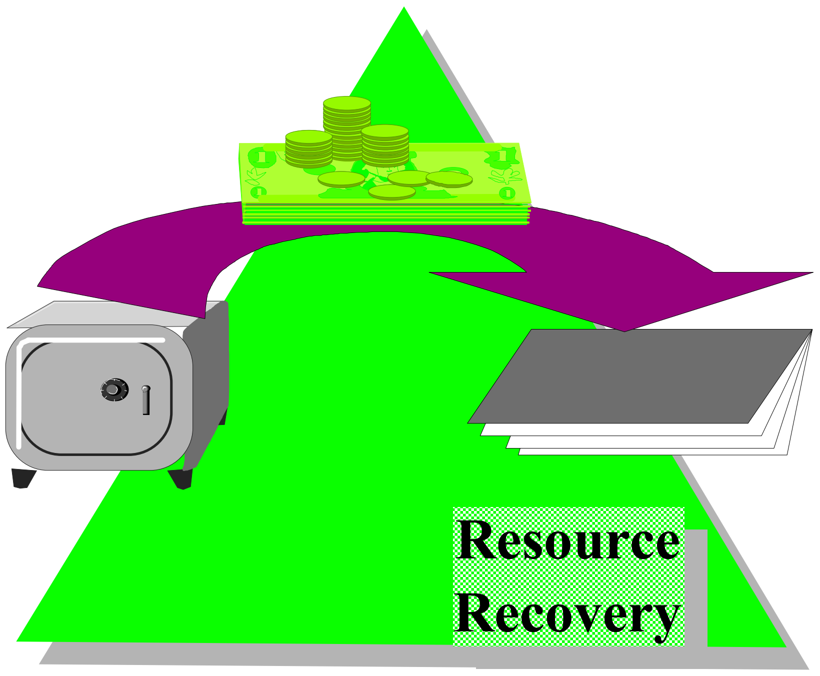


Agenda

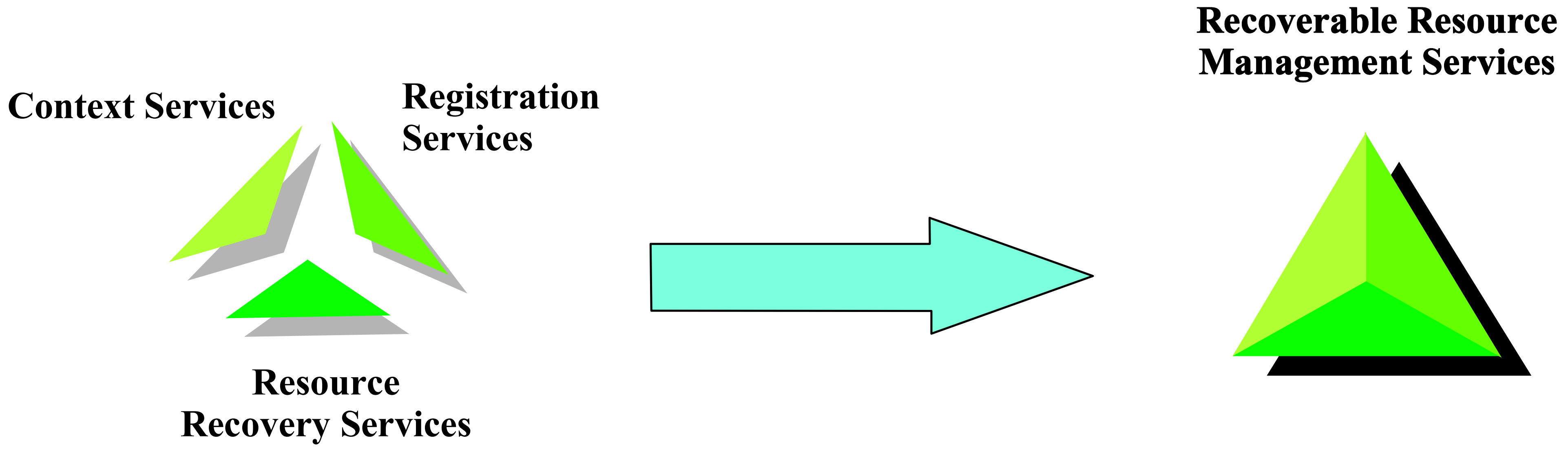
- Basic Concepts of z/OS Resource Recovery
- IMS OTMA and APPC Protected Transaction
- IMS Batch Support of RRS
- RRS Multisystem Cascaded Transactions
- IMS V8 OTMA/APPC Shared Queue exploitation
- IMS V8 RRS support for e-business connectors
- Considerations
- Summary

z/OS Resource Recovery

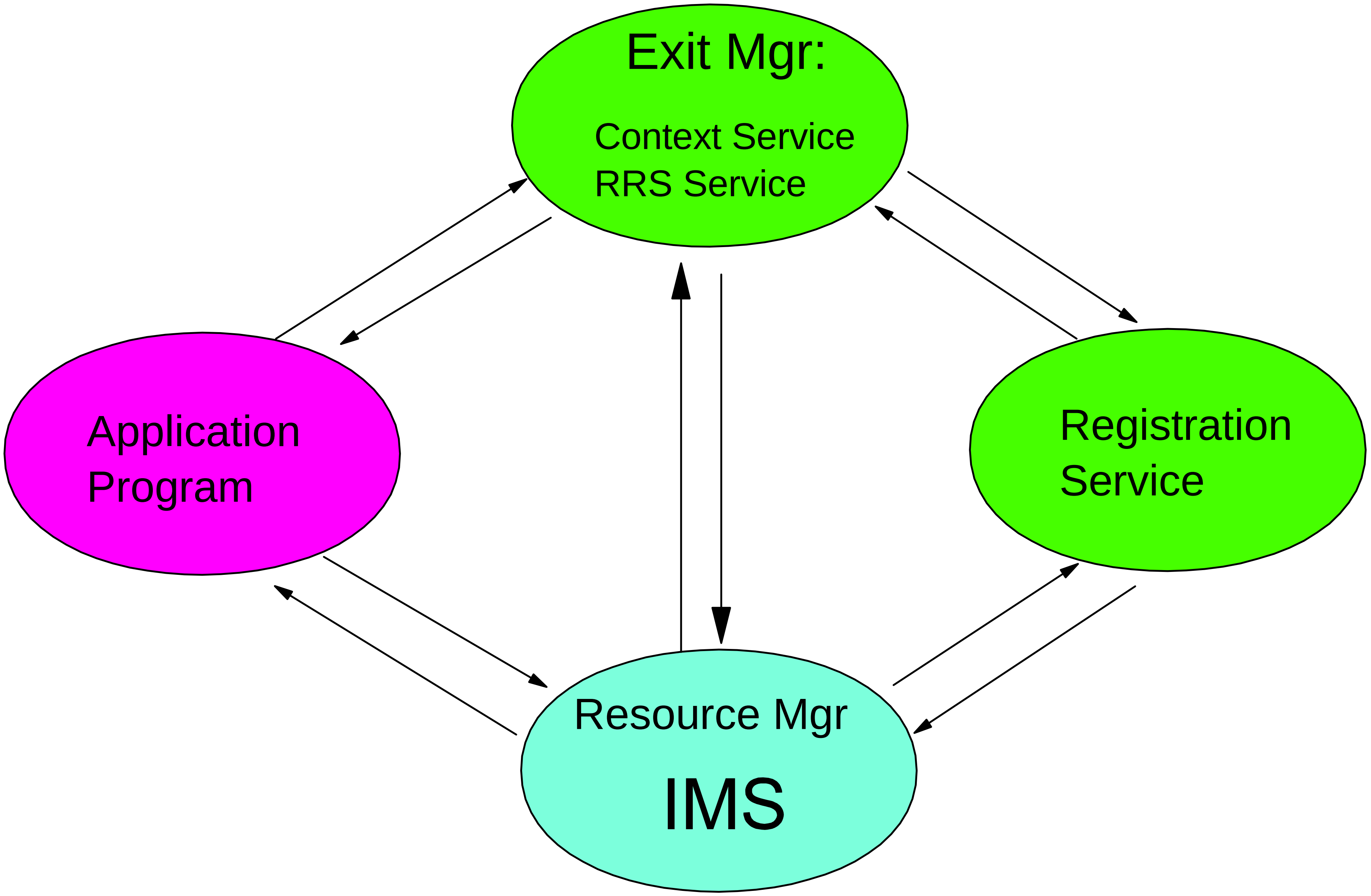
- The z/OS Resource Recovery consists of the protocols and program interfaces that allow an application program to make consistent changes to multiple protected resources. (Protected resources are sometimes called recoverable resources.)



RRMS and RRS

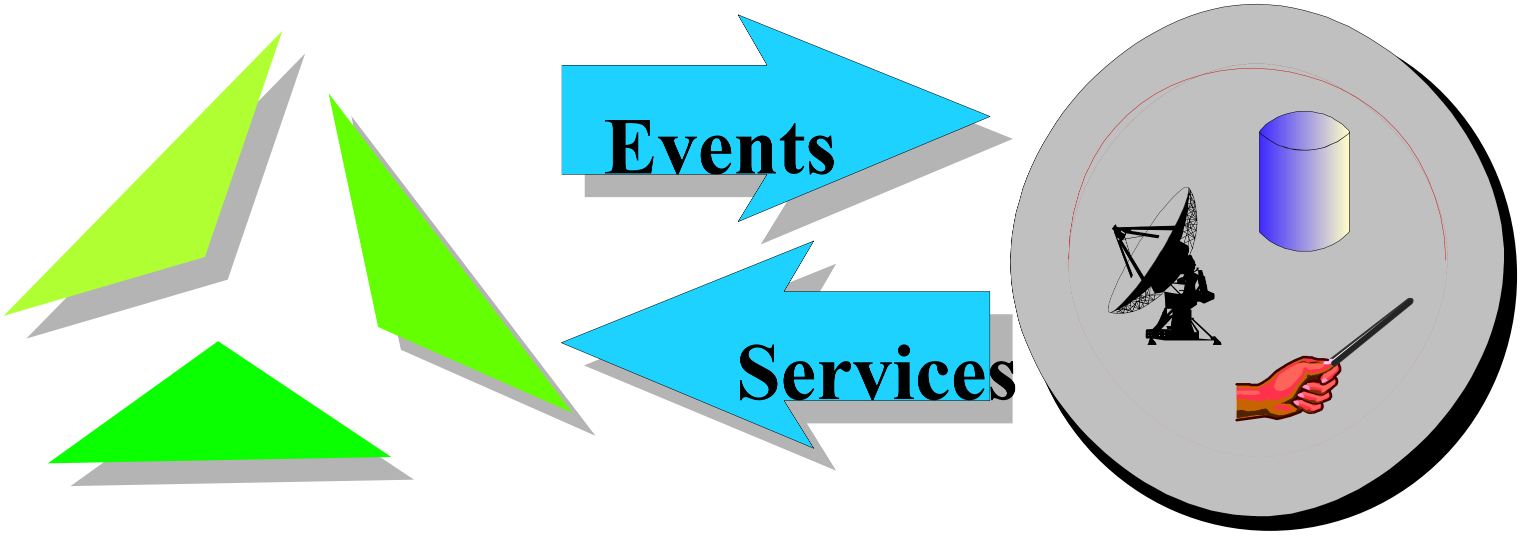

Context Services
Registration Services
Resource Recovery Services
Recoverable Resource Management Services

z/OS Resource Recovery



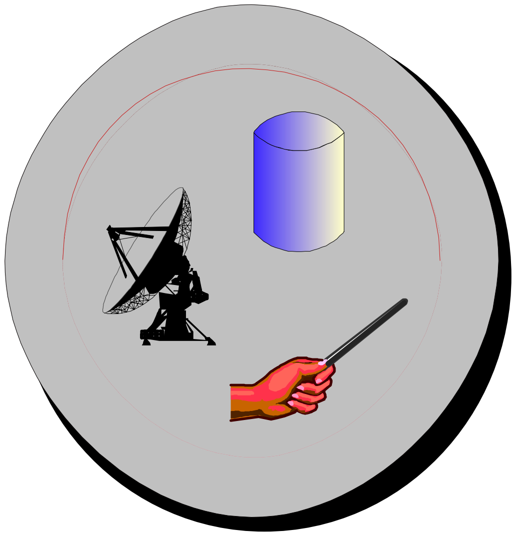
Exit Managers - Providing Services to Resource Managers

- **Exit Managers,**
  - Context Service and Resource Recovery Services
  - Provide **services** to RMs
  - Inform RMs of **events** via **exits**



Resource Managers

- **All resource managers (RM):**
  - Are **authorized**
  - **Register** with the system
  - Supply **exit routines**
  - Have a **system unique name**



- Additionally, RRS compliant resource managers:
  - Go through RRS **restart** processing
  - Have a **Sysplex unique name**
  - Respond to **syncpoint** events

Types of Resource Managers

There are 3 types of Resource Managers

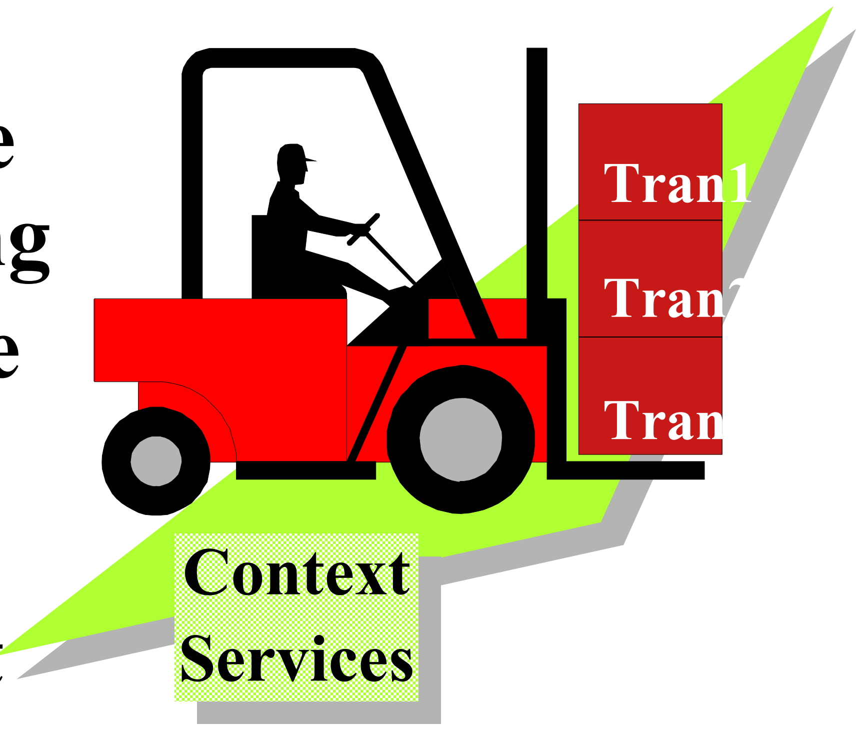
- Data Managers
  - DB2 DB, IMS DL1, VSAM
- Communications Managers
  - APPC/PC, TRPC, MQ
- Work Managers
  - IMS TM, CICS TS, DB2 Stored Procedures

Registration Services - Connecting RMs and EMs

- Always available
- Is informed by EMs and RMs when they are available
- Informs EMs when RMs come and go
- Informs RMs when EMs come and go
- Provides global anchors for each

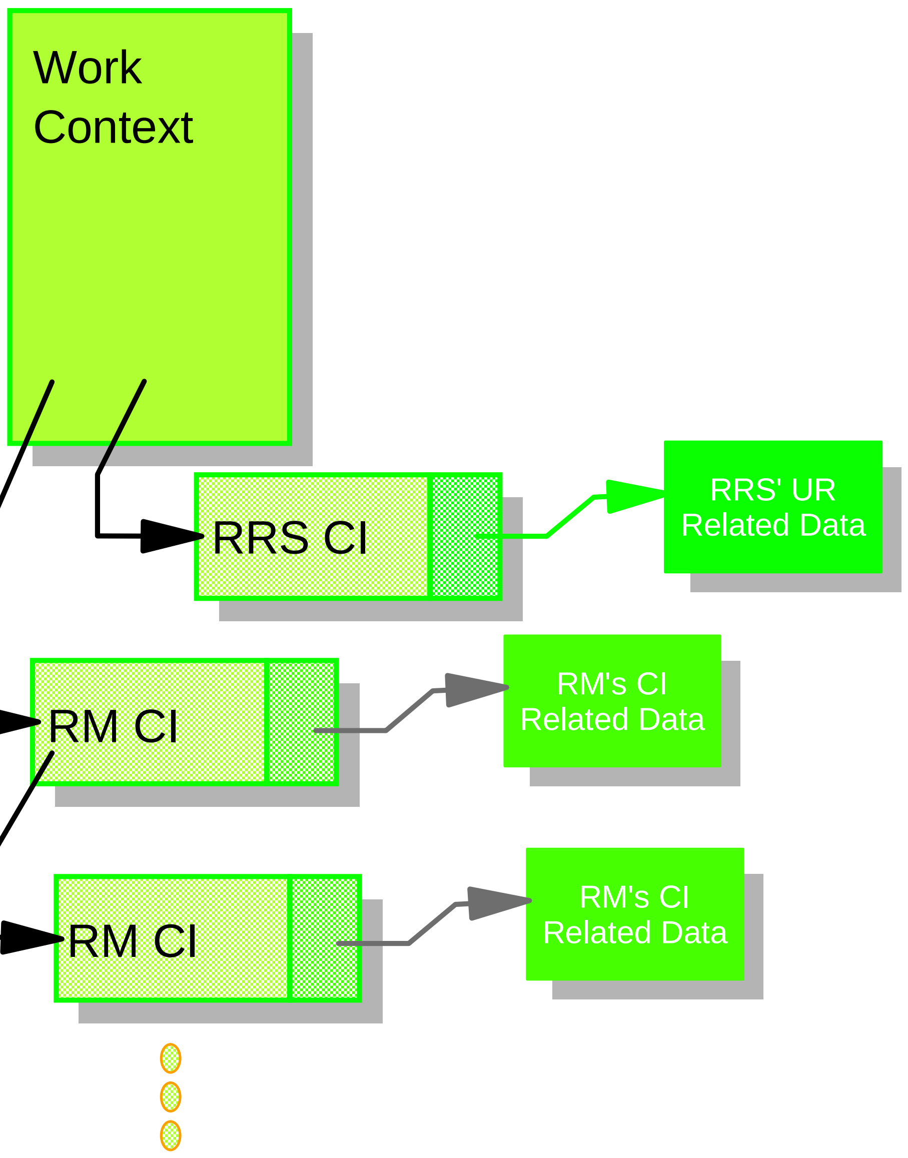
Context Services - Moving Transactions Around

- Used by a Work Manager (WM)
- WM tells CS where the work request is running
- WMs tells CS when the work request moves
- RMs are told by CS when the work request is completed



Context Services - Basic Structures

- The Context (or Work Context) is the base structure
- When a RM expresses interest in a Context, a CI is created for it.
- When RRS expresses interest in a context it gets a "special" CI
- Each CI has "user data" associated with it

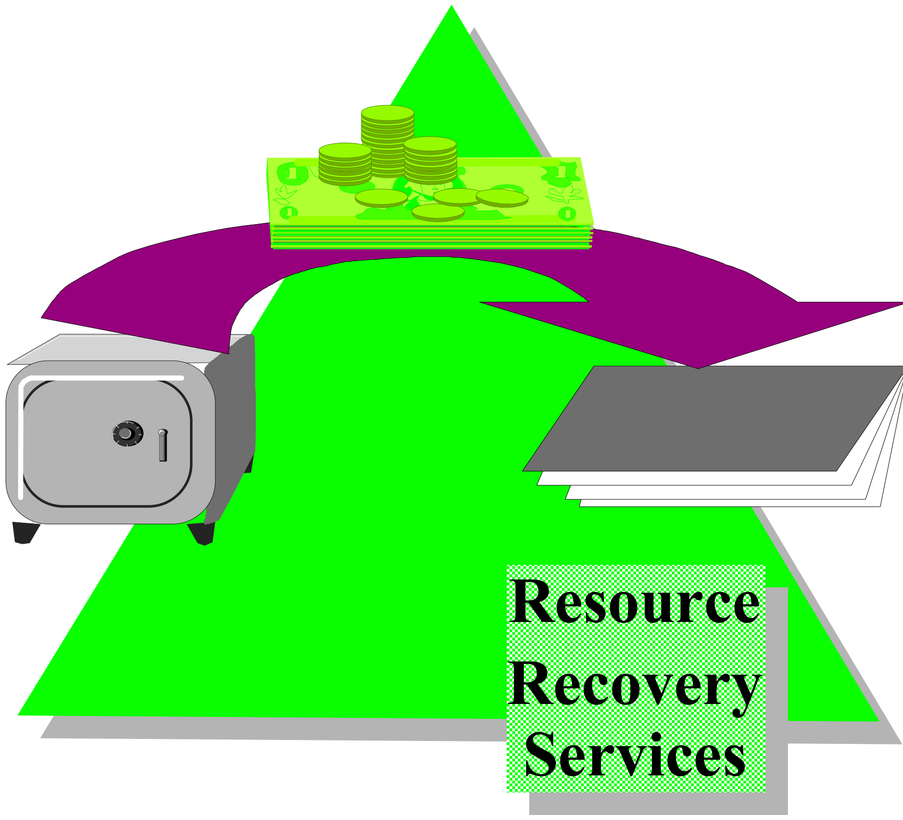


There are Two Types of Contexts

Du Native Context	**Private Context**
Always associated with a TCB	Doesn't have to be associated with any TCB
Can never move	Can move from TCB to TCB even across spaces
Created on first reference	Explicitly created by WM
Ends when address space or TCB ends or when WM says to end it	End as Native, also when WM ends or unregisters, if unassociated
Context end cannot be stopped	WM can stop Context end

Resource Recovery Services - The Syncpoint Coordinator

- **Coordinates the two-phase commit process**
- **Creates an association between a Unit of Recovery (UR)** and a Work Context
- Preserves UR state across all failures

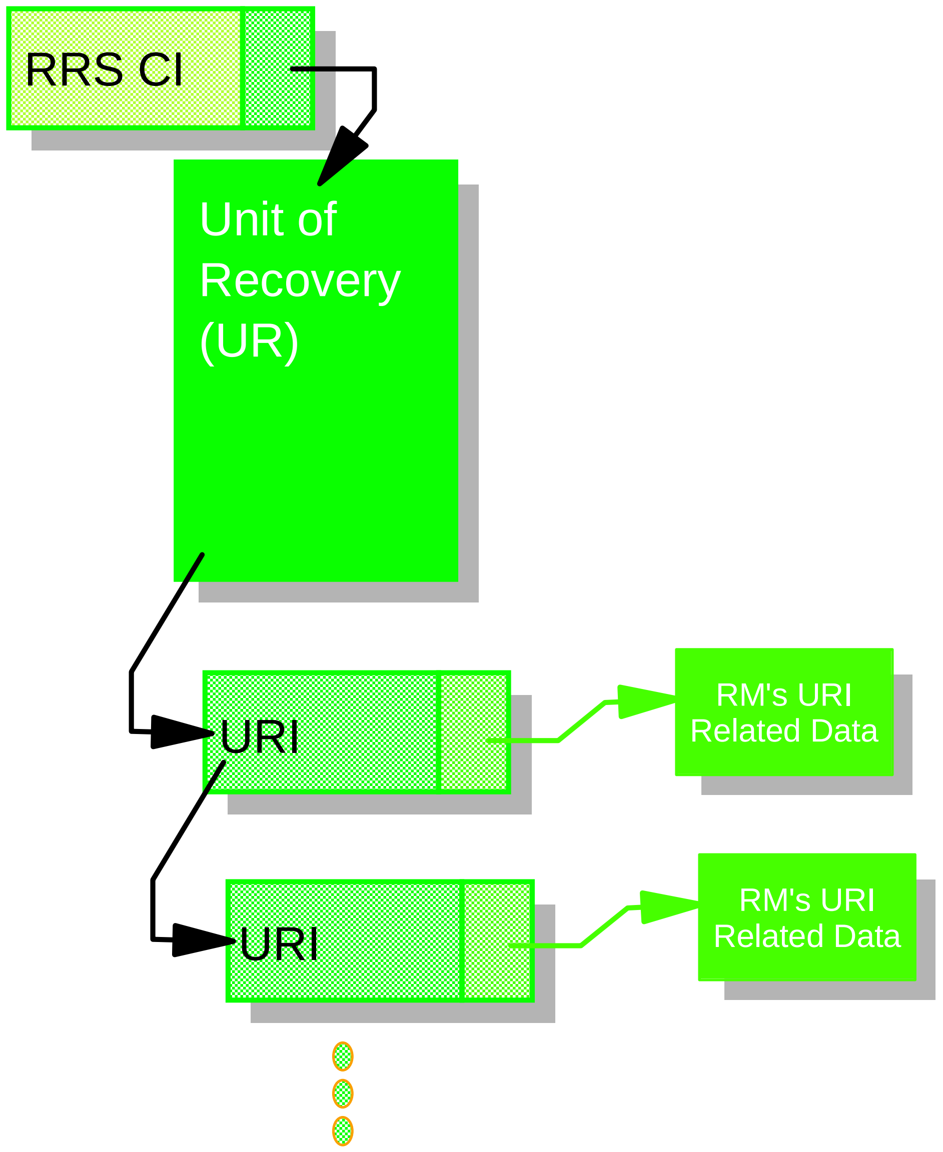


RRS - Basic Structures

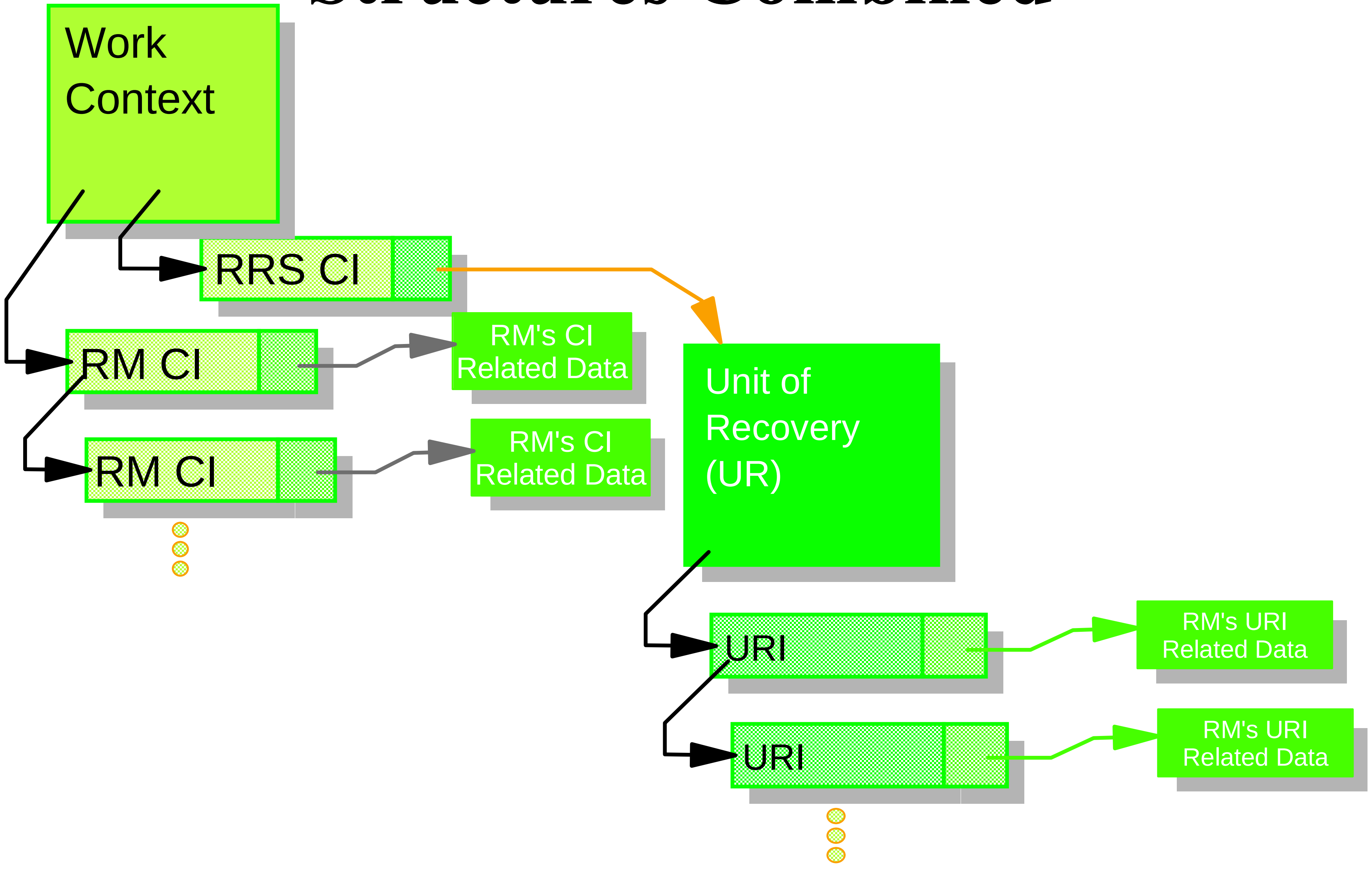
The Unit of Recovery (UR) is the base structure

Each time an RM expresses interest a new URI is created

Each URI has persistent and non-persistent "user data" associated with it

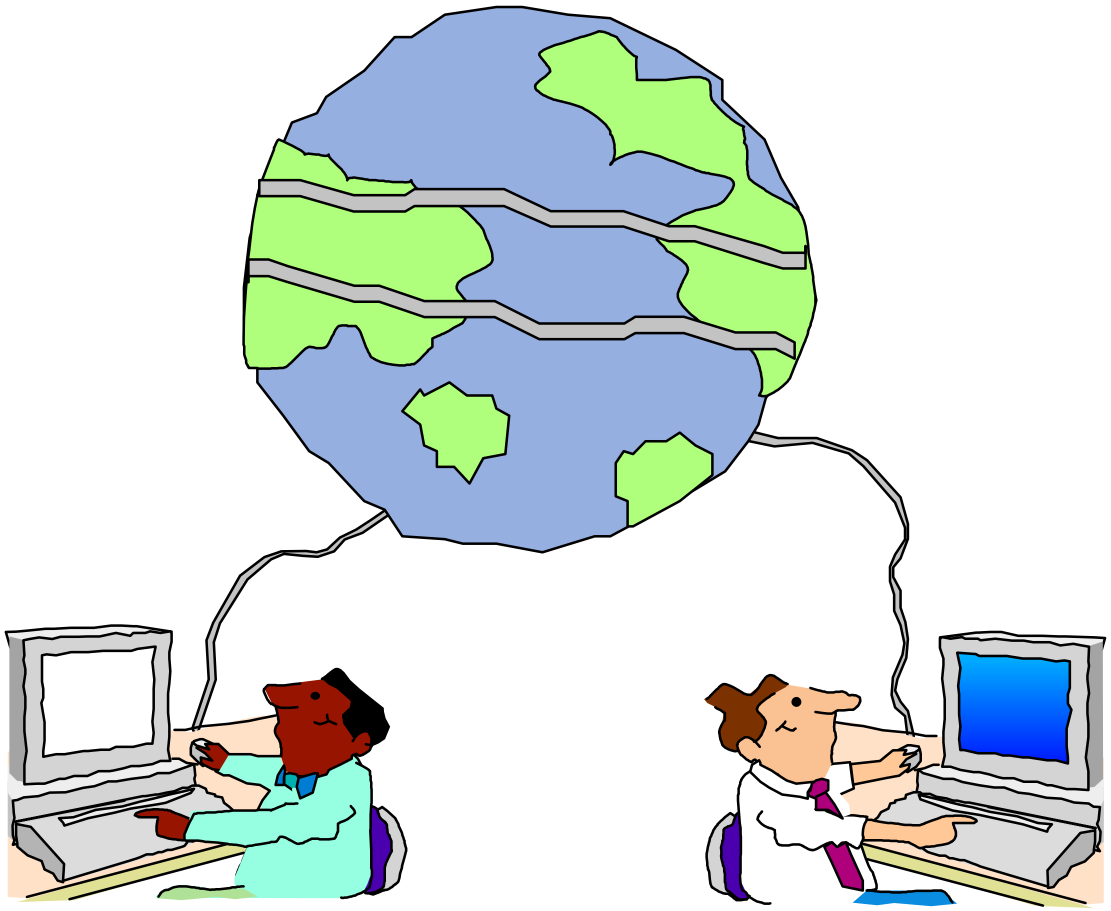


Context Services & RRS - Basic Structures Combined



Sync Point Processing:

Sync Point Phase 1 - "Prepare" : Each resourece manager votes
Sync Point Phase 2 - "Commit" : Changes are committed

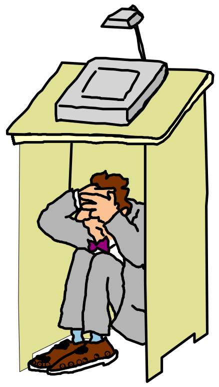


Vote

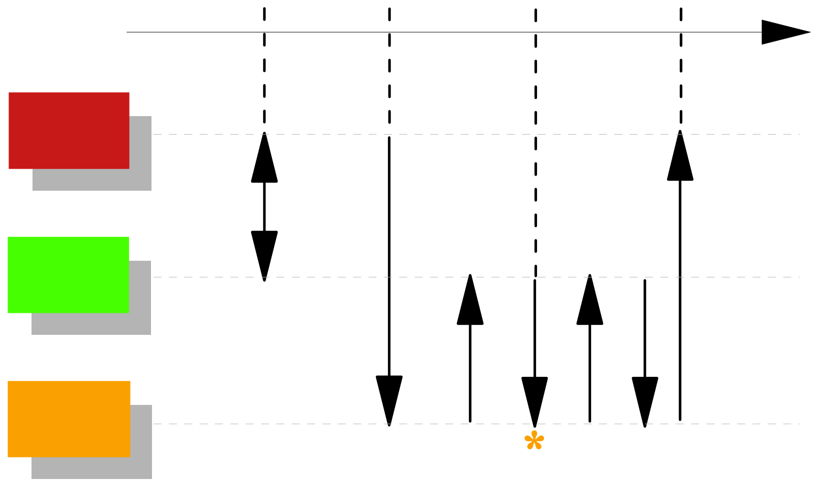
Vote

RRS Unit of Work States

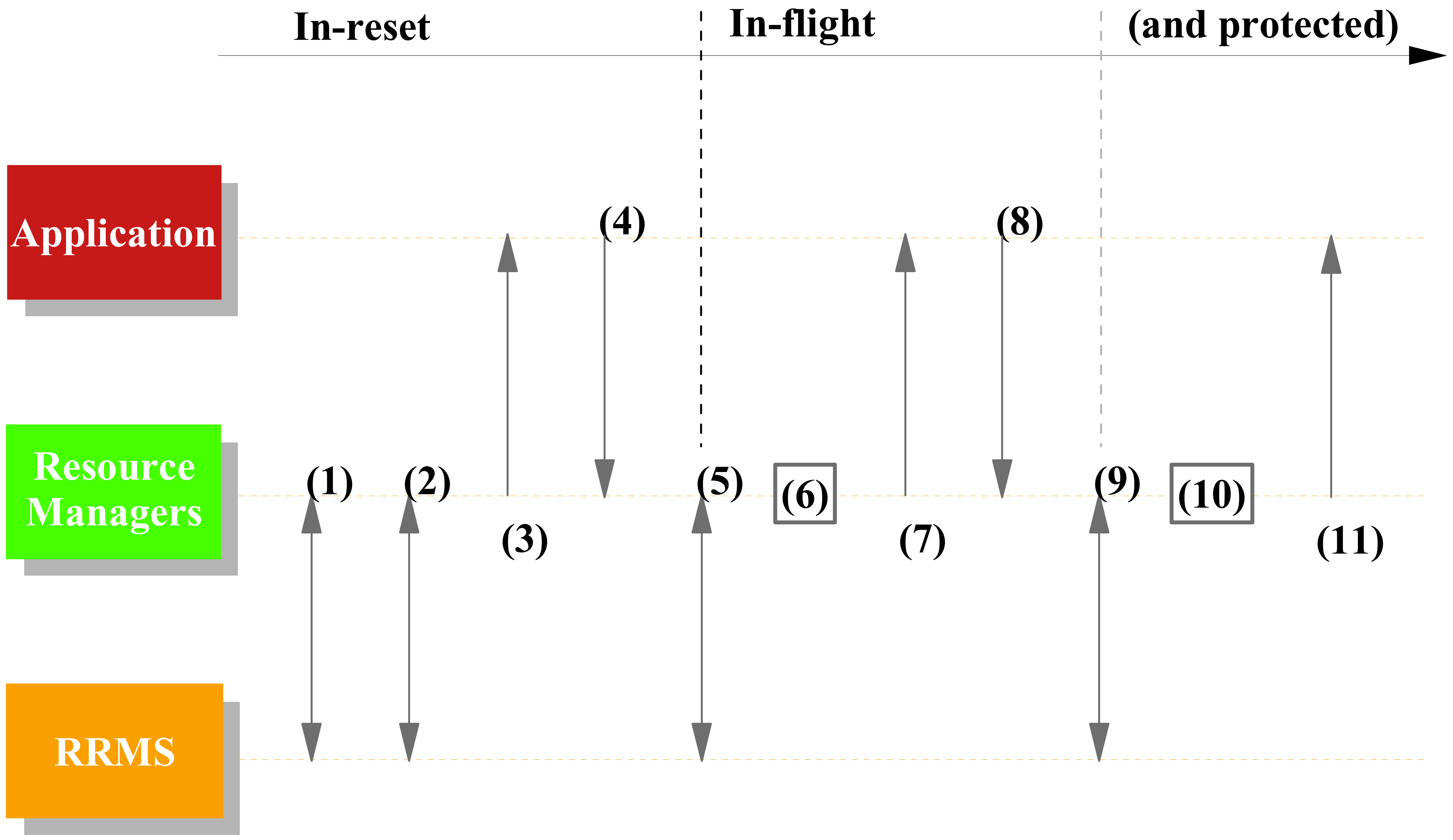
- **IN-FLIGHT :**
  - Work changes in process
- **IN-DOUBT**
  - Work changes between phase 1 and phase 2
- **IN-COMMIT**
  - Work changes are committed
- **IN-BCKOUT**
  - Work changes are backed out



A Distributed Syncpoint Example



Part 1 - The Application Makes Changes



art 1 - The Application Makes Changes

, Work Manager gets a private Context from Context
 Services.

. Work Manager associates Context with a task (via CS).

. Work Manager dispatches application.

. Application makes read request to Data RM1.

. RM1 expresses unprotected interest in UR, UR and URI1
 for RM1 are created.

. RM1 locks and reads data

. RM1 returns to application.

. Application makes write request to Data RM2.

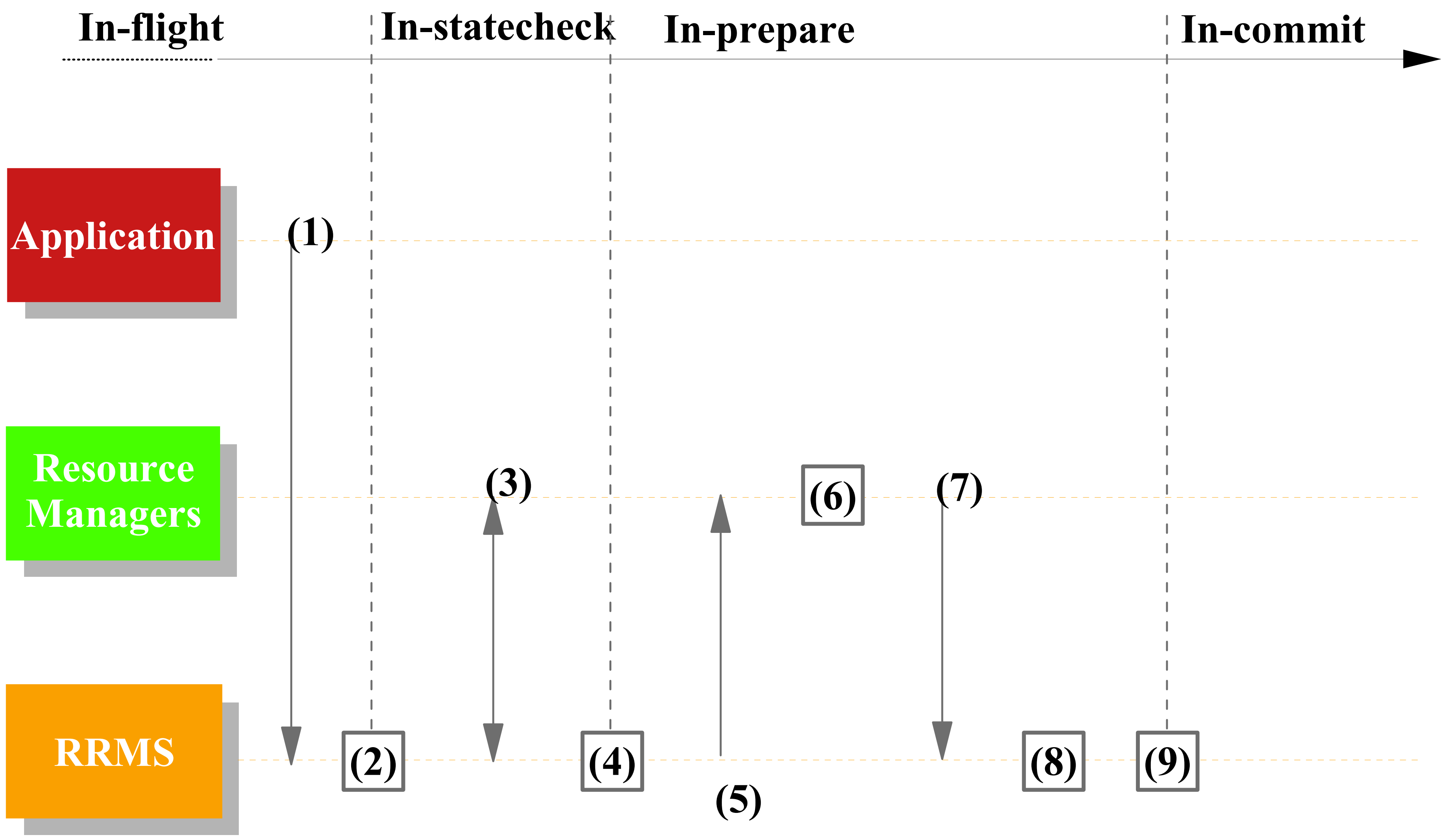
. RM2 expresses protected interest in UR, URI2 for RM2 is
 created.

0. RM2 locks data then makes and logs changes.

1. RM2 returns to application

Note: Steps 4,6-7 and 8,10-11 can be repeated.

Part 2 - Syncpoint Processing Begins



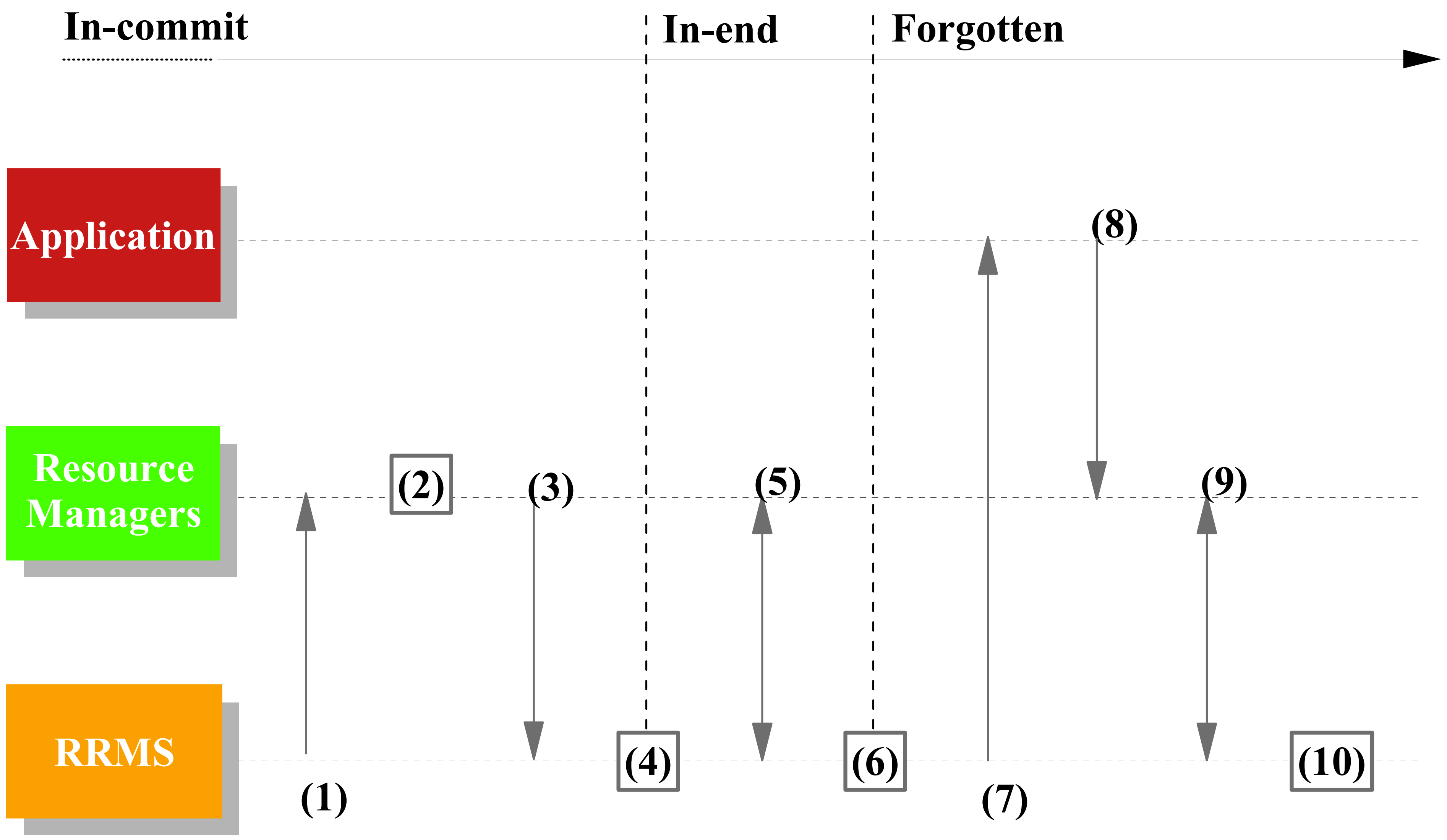
art 2 - Syncpoint Processing Begins

. Application requests all changes be committed.
. RRS starts syncpoint process [In-statecheck begins].
. RRS drives all RM's (optional) statecheck exits and they return.
. [In-prepare begins] If any RM requested presume nothing protocols, RRS logs a PRP (prepare) record.
. RRS drives All RM's (required) prepare exits.
. RMs take prepare actions
a. RM1 (read only) releases data locks.
b. RM2 forces undo/redo records.

. RMs return from prepare exits
a. RM1 (read-only) votes forget. It is no longer involved in the UR.
b. RM2 votes OK.
. RRS determines overall result (in this case it is OK).

. RRS logs a CMT (commit) record. This is the "Atomic Instant" [in-commit begins].

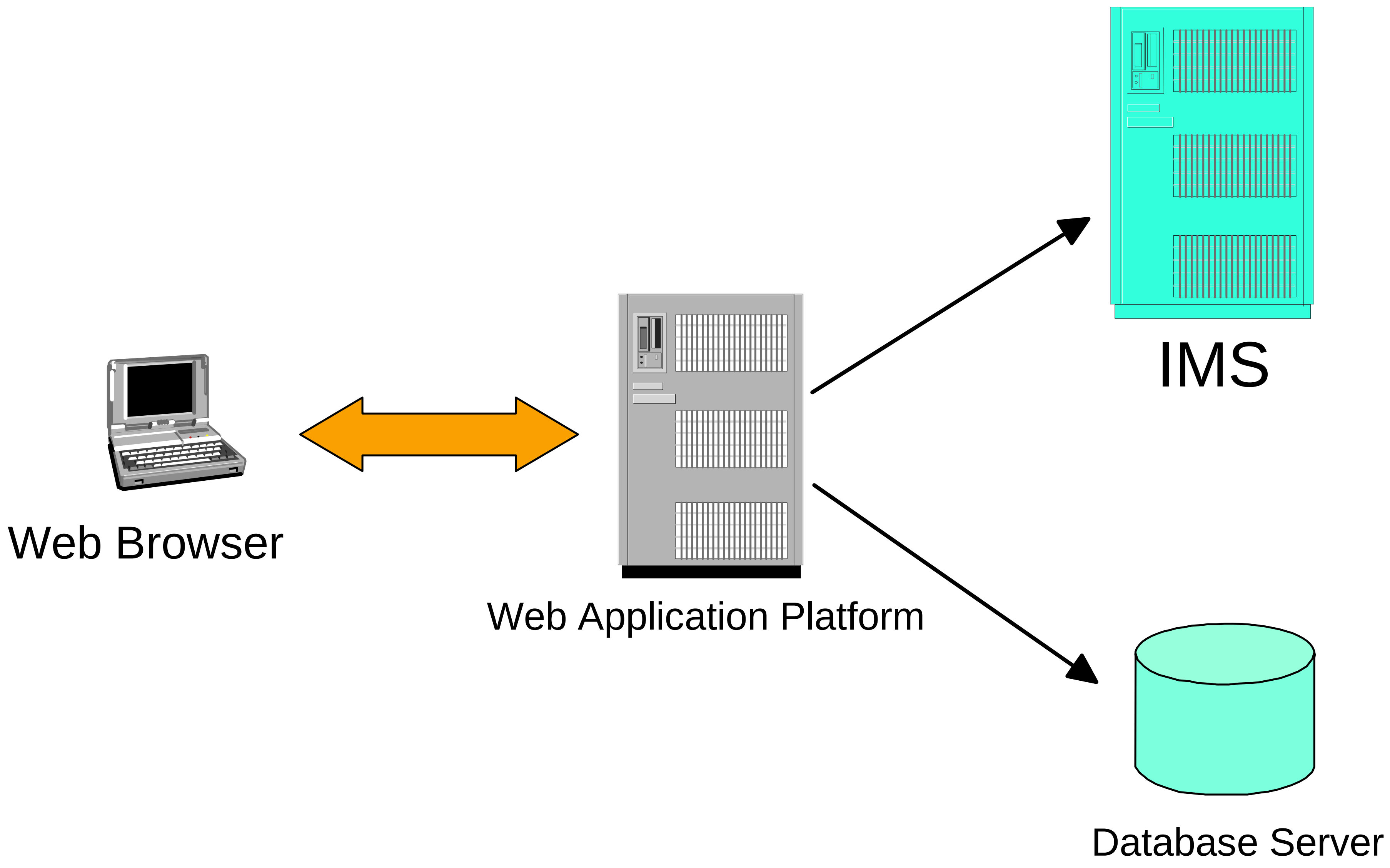
Part 3 - Committed and Forgotten



art 3 -Committed and Forgotten

. RRS drives RM2's (required) commit exit. (RM1 has forgotten and so its exits are no longer driven).
. RM2 takes commit action by releasing data locks. The changes are now visible to other work requests.
. RM2 returns to RRS.
. RRS begins end processing [in-end begins].
. RRS drives RM2's (optional) end-UR exit and it returns.
. UR processing is completed. RRS logically deletes the CMT record from the log.
. RRS returns to the application informing it that the commit request completed successfully.
. Application completes and returns to its Work Manager.
. Work Manager ends the private Context.
0. Sometime later as part of log cleanup, RRS physically deletes the CMT record from the log

Extending the Enterprise with RRS



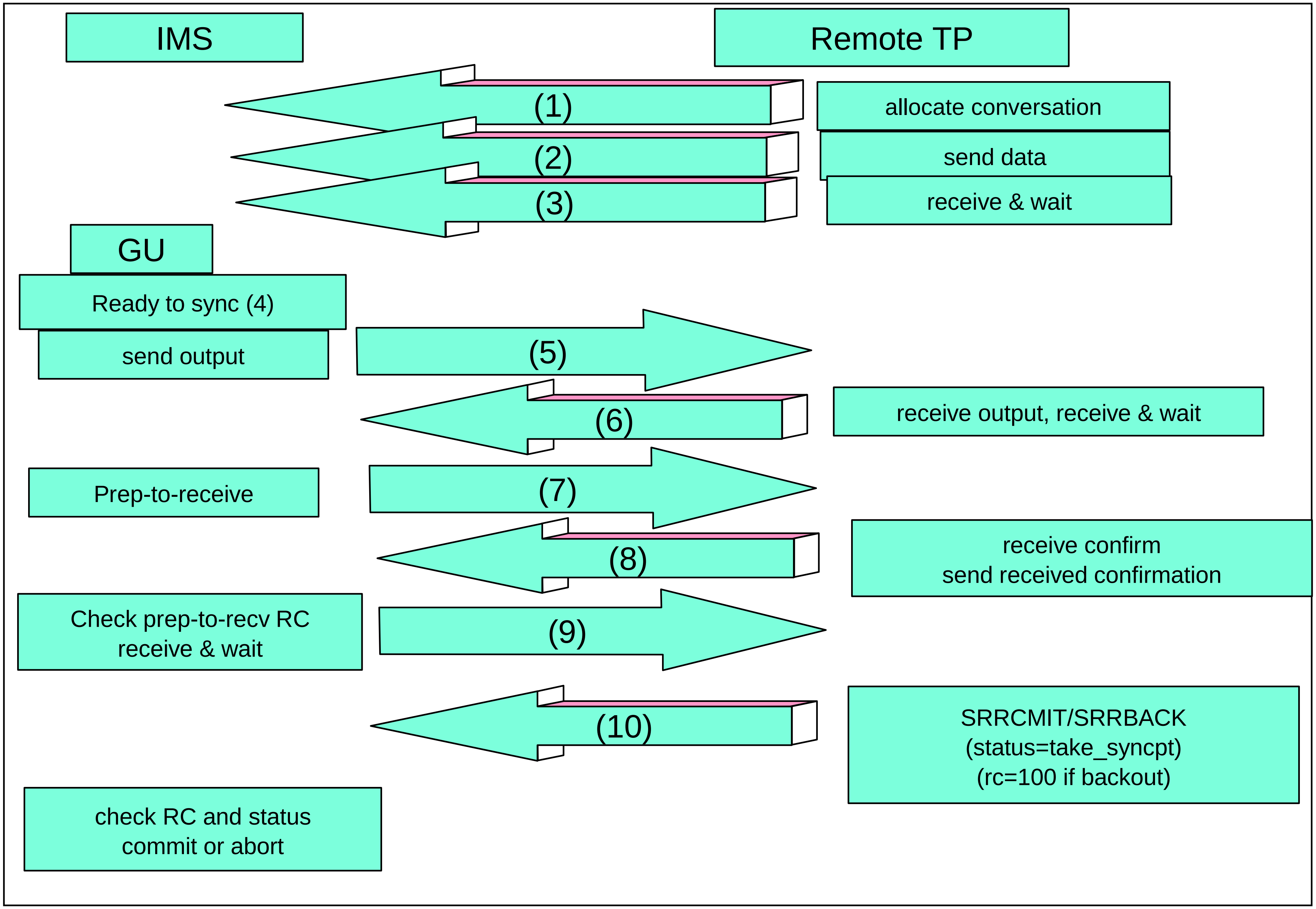
e-business

IMS Connecting to RRS

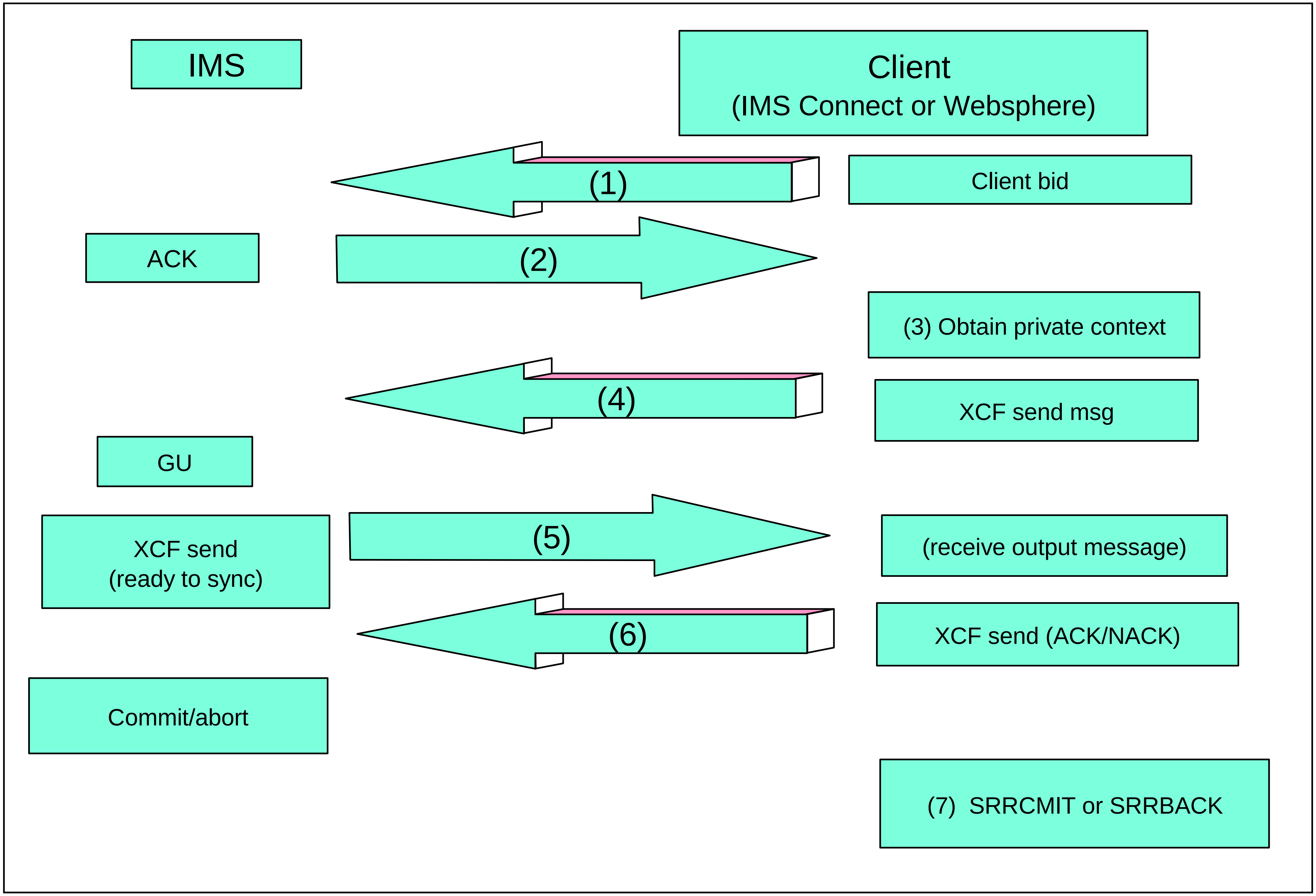
OTMA/APPC Synchronization Level (Sync_Level)

- None : synchronization not required
- Confirm : confirmation is required before the syncpoint
- SYNCPT: RRS coordinated two-phase commit flows

IMS-APPC/PC Conversation Flow

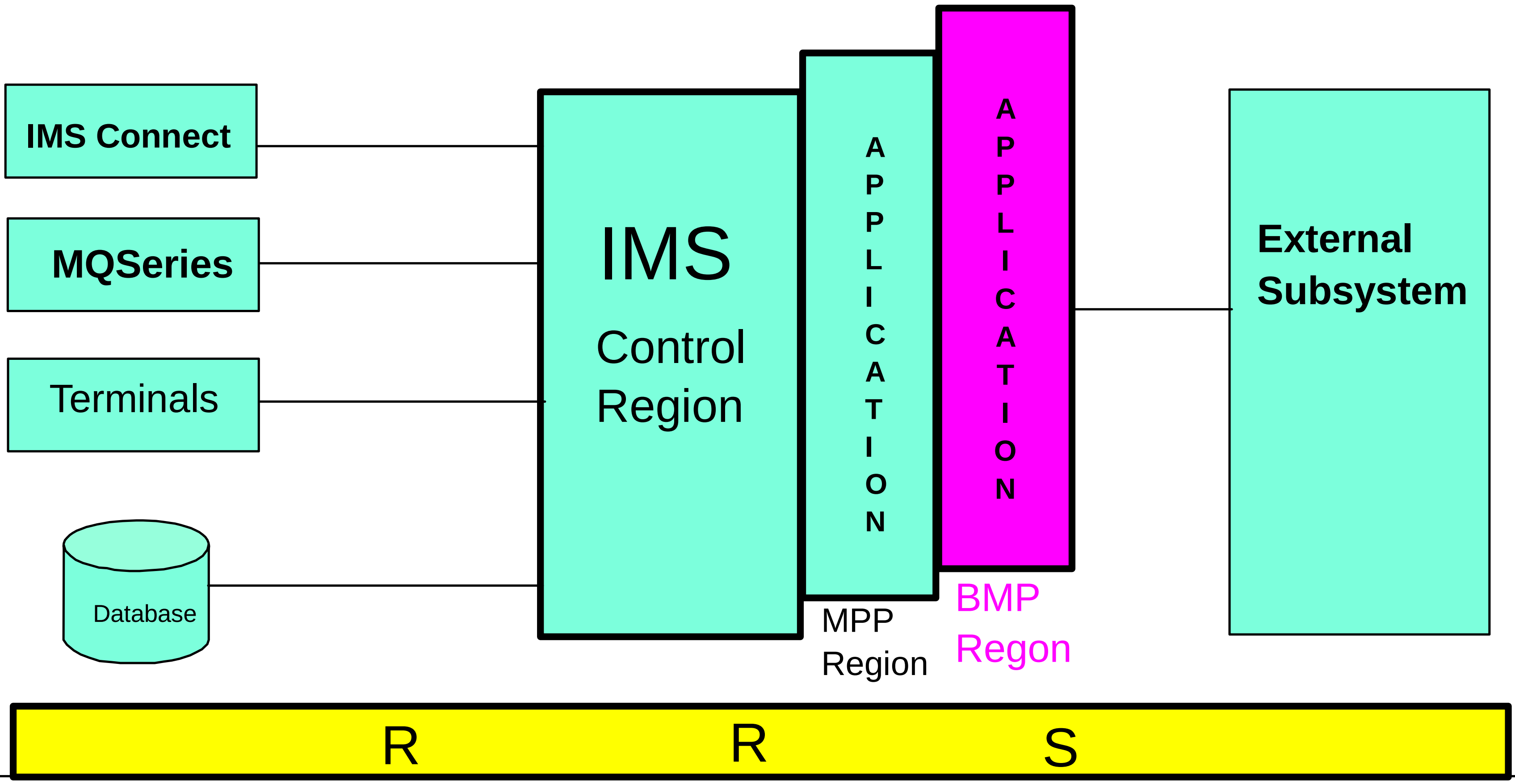


OTMA Conversation Flow



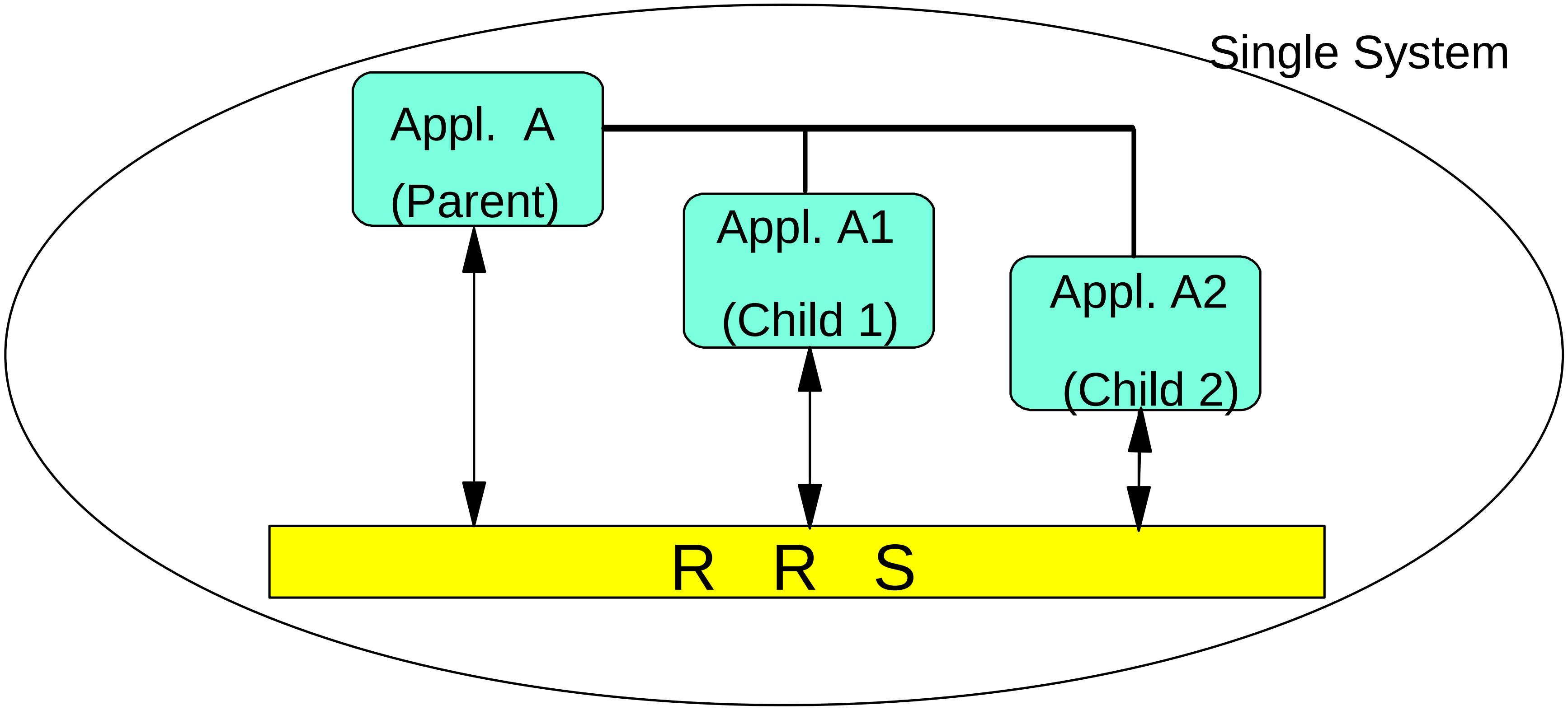
IMS Batch Support of RRS

- V7 APAR PQ51895 supports the 2-phase commit process for batch application programs.
- Users have batch applications which can access and update the resources of more than one resourece managers, like IMS or DB2.
- The "RRS=" in DBBBATCH and DLIBATCH can be used to activate or deactivate the function.



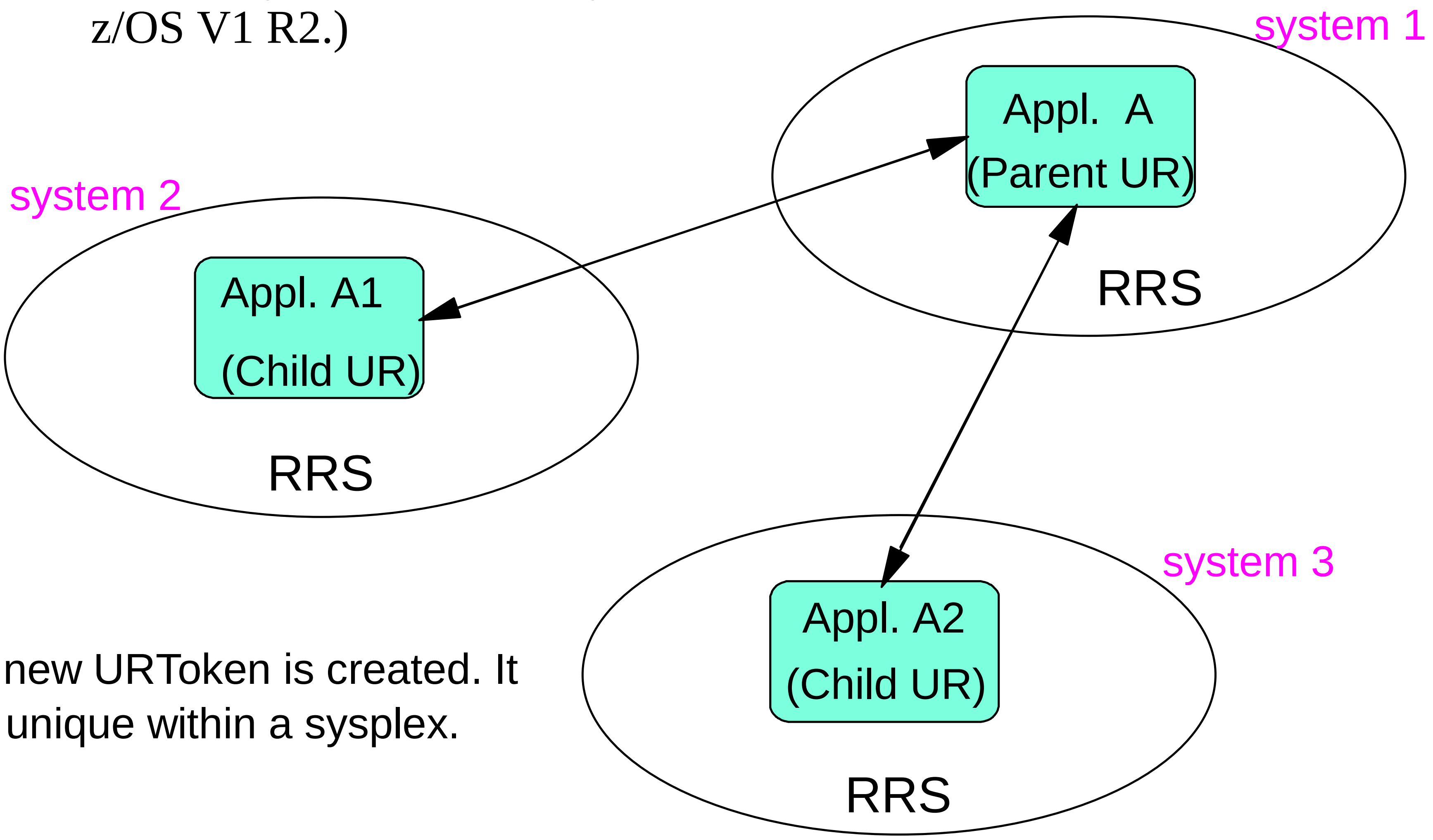
RRS Single-System Cascaded Transaction

- When a single work request needs to run multiple application parts in parallel, the application can create multiple threads each executing part of a cascaded transaction. All of the changes made by the separate applications can then be part of a single commit scope managed localy by RRS. (The feature was introduced in OS/390 V2 R9.)



RRS Multisystem Cascaded Transaction

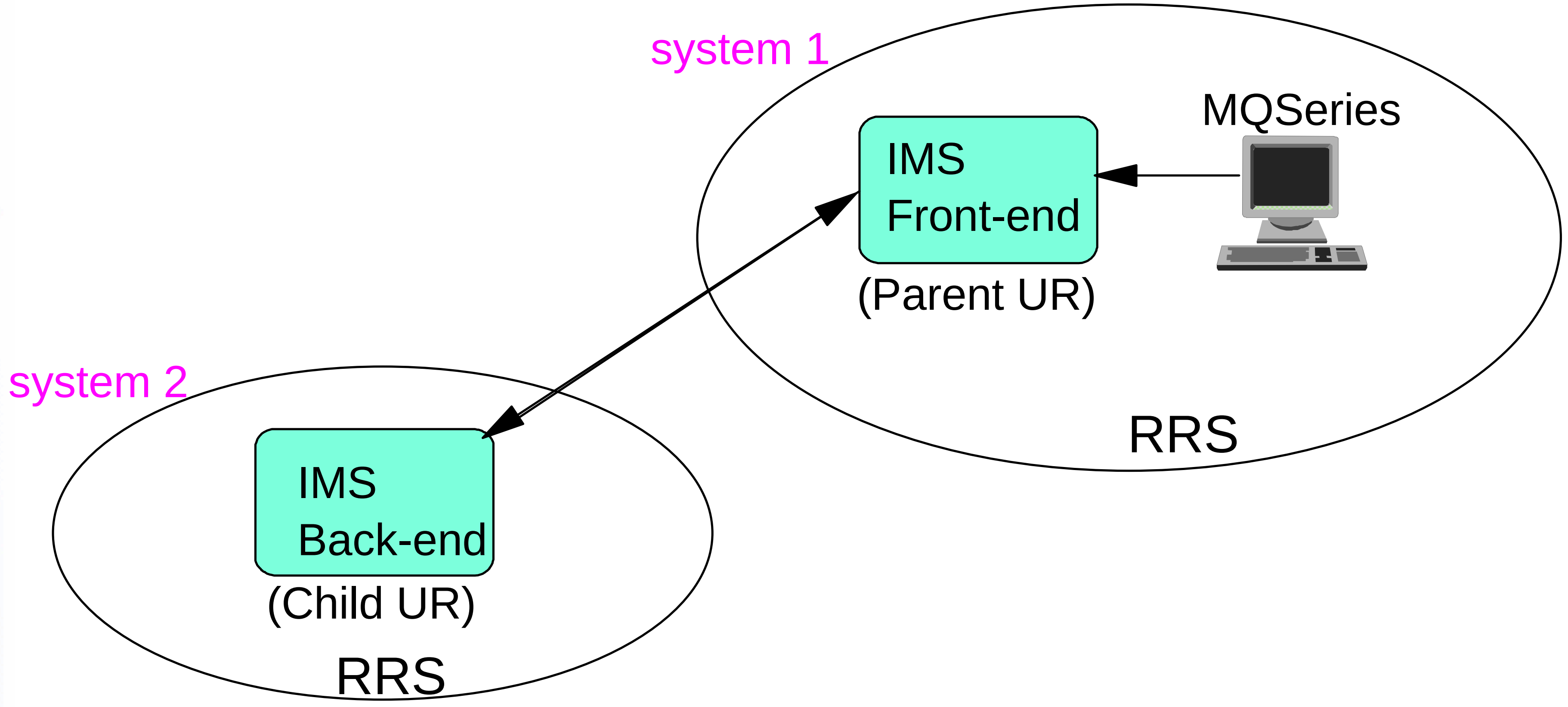
- An application executing on one system in a sysplex can have transactional access to data or applications available on another system in a sysplex. RRS coordinates a transaction across multipe systems within a sysplex. (The feature was introduced in z/OS V1 R2.)



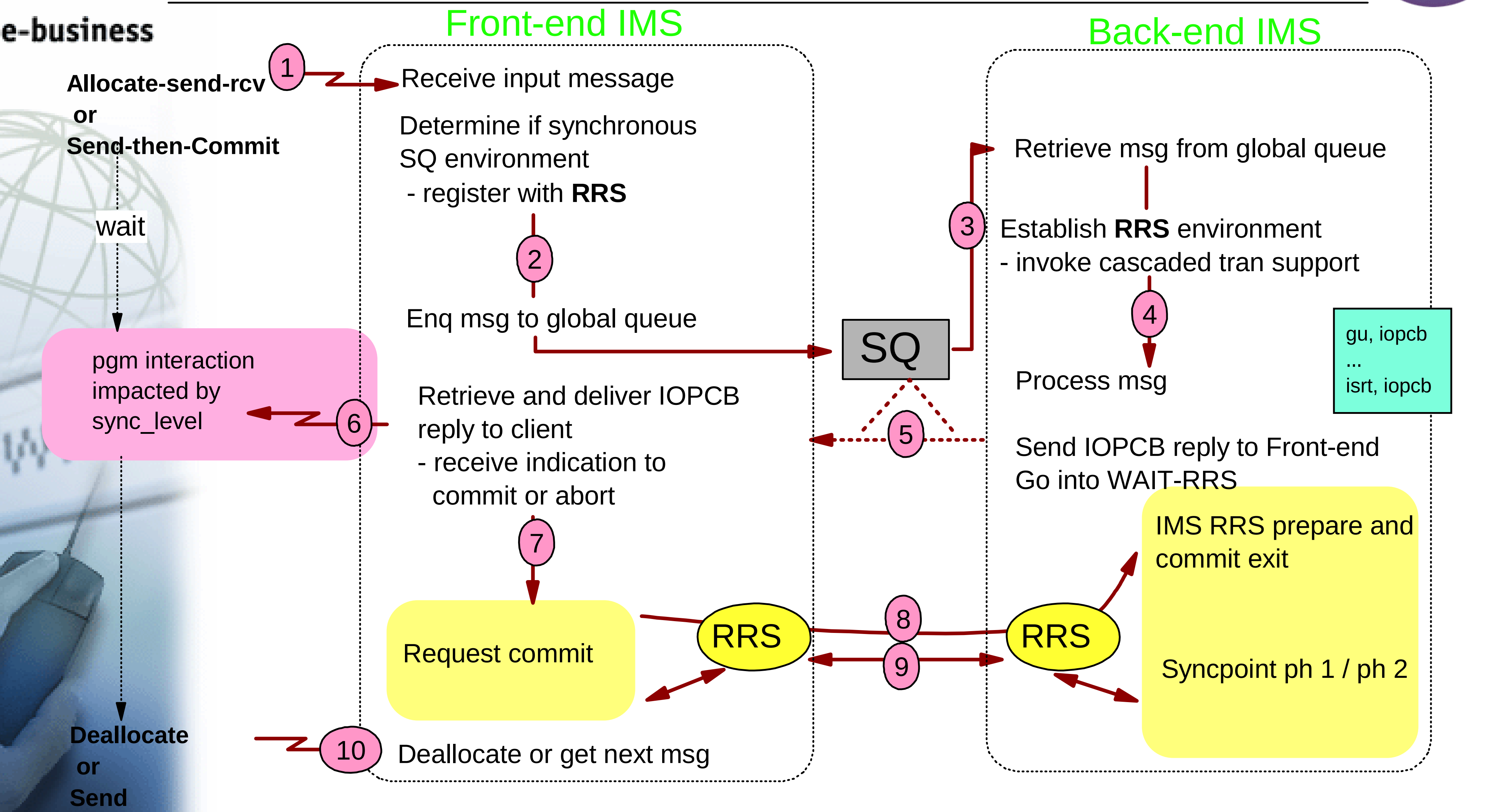
A new URToken is created. It is unique within a sysplex.

RRS Multisystem Cascaded Transaction application in IMS V8

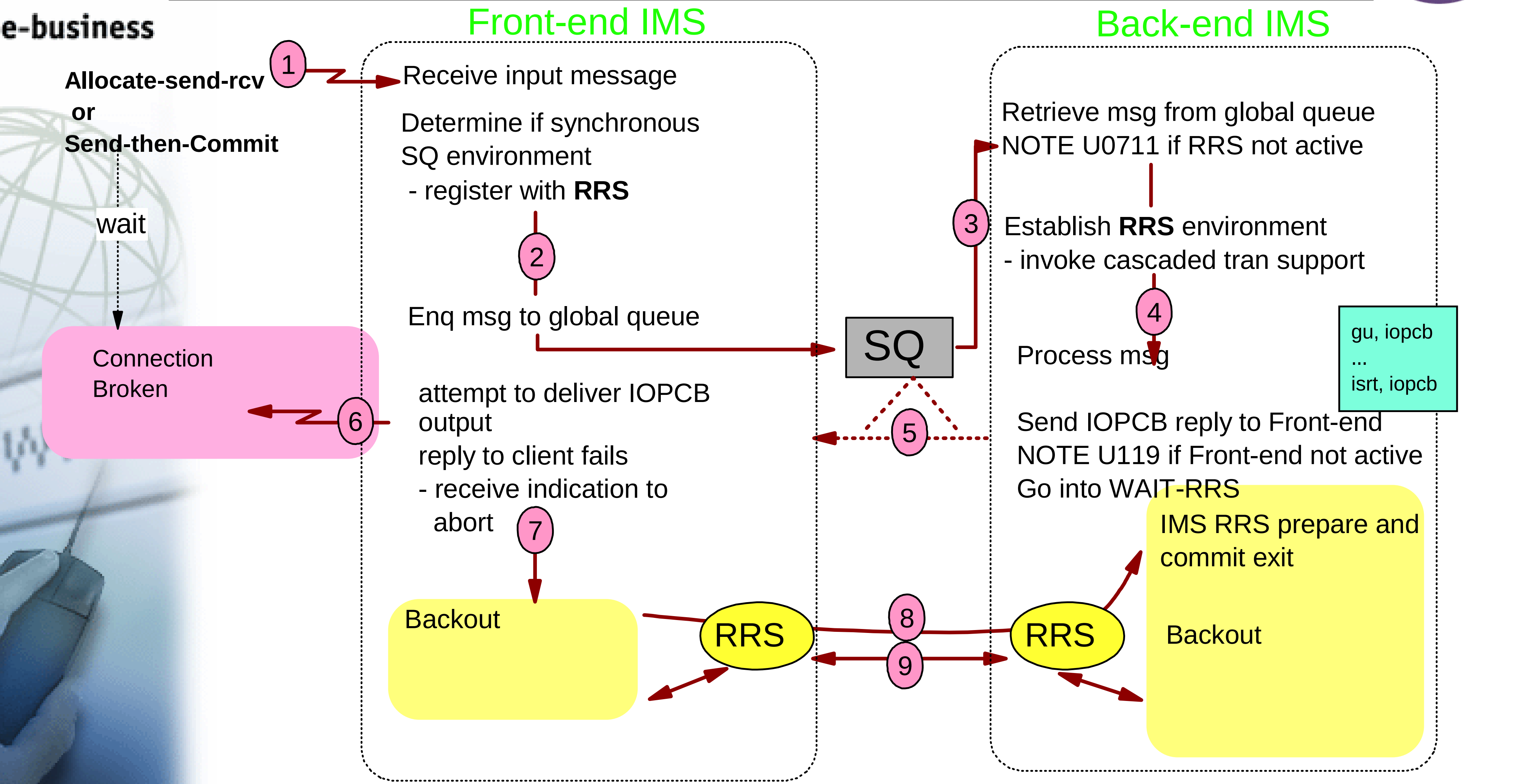
- IMS V8 Synchronous APPC and OTMA Shared Queues Enablement is one of RRS applications using RRS multisystem cascaded transaction feature. The new V8 APPC/OTMA function allows Synchronous APPC/OTMA workload to be distributed and executed on any of the IMS systems in the Shared Queues group.



IMS OTMA/APPC Shared Queue with RRS cascaded transaction (commit flow)

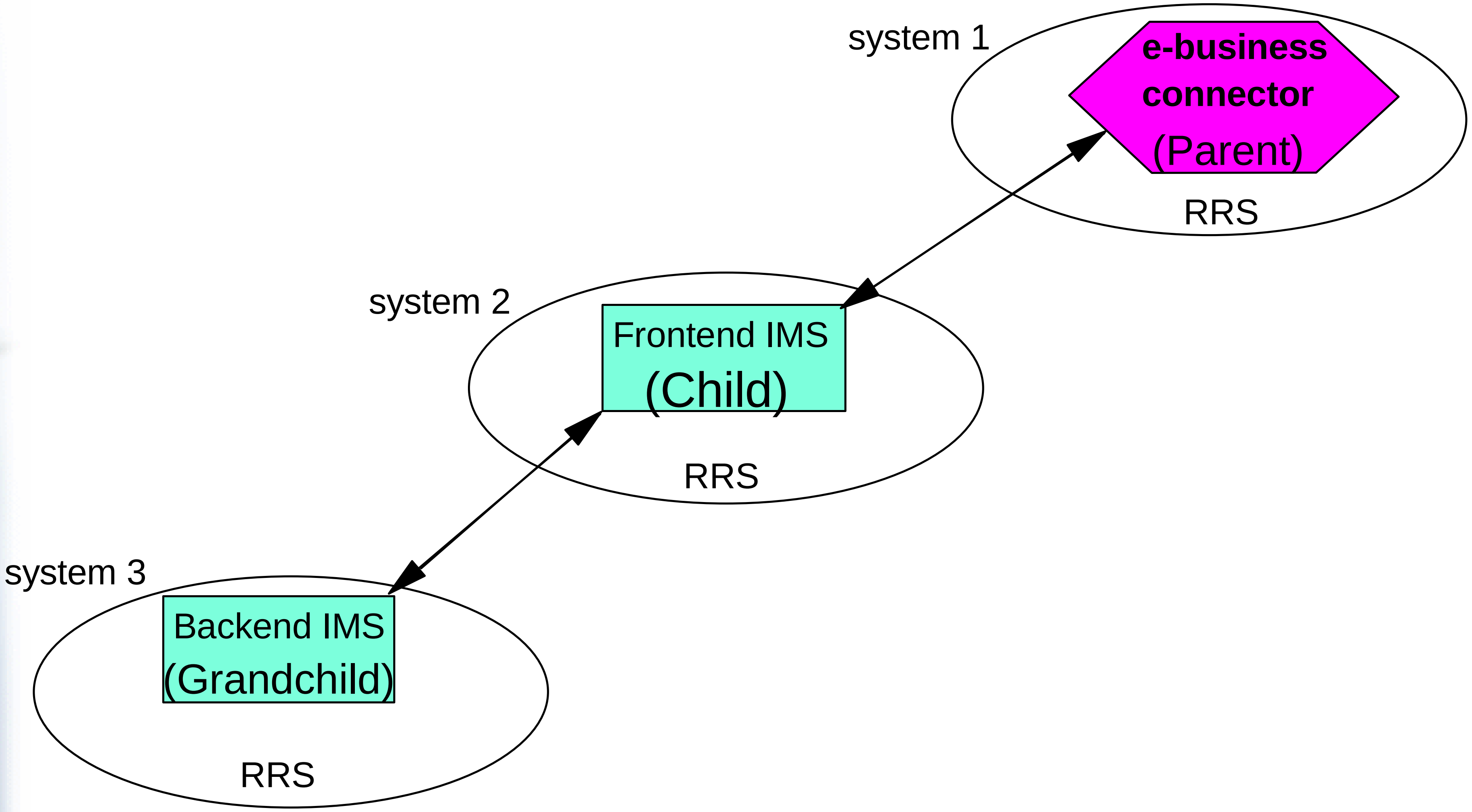

Front-end IMS
Back-end IMS
1
Allocate-send-rcv
or
Send-then-Commit
wait
Receive input message
Determine if synchronous SQ environment
- register with RRS
2
Enq msg to global queue
SQ
3
Retrieve msg from global queue
Establish RRS environment
- invoke cascaded tran support
4
Process msg
gu, iopcb
...
isrt, iopcb
pgm interaction impacted by sync_level
6
Retrieve and deliver IOPCB reply to client
- receive indication to commit or abort
5
Send IOPCB reply to Front-end
Go into WAIT-RRS
7
IMS RRS prepare and commit exit
Request commit
RRS
8
9
RRS
Syncpoint ph 1 / ph 2
Deallocate
or
Send
10
Deallocate or get next msg

IMS OTMA/APPC Shared Queue with RRS cascaded transaction (backout flow)


Front-end IMS
Back-end IMS
Allocate-send-rcv
or
Send-then-Commit
wait
Connection Broken
1
Receive input message
Determine if synchronous SQ environment
- register with RRS
2
Enq msg to global queue
SQ
attempt to deliver IOPCB output
reply to client fails
- receive indication to abort
6
7
5
Backout
RRS
8
9
RRS
3
Retrieve msg from global queue
NOTE U0711 if RRS not active
Establish RRS environment
- invoke cascaded tran support
4
Process msg
gu, iopcb
...
isrt, iopcb
Send IOPCB reply to Front-end
NOTE U119 if Front-end not active
Go into WAIT-RRS
IMS RRS prepare and commit exit
Backout

RRS Multisystem Cascaded Transaction application for IMS e-business connectors

- IMS V8 OTMA can process protected transaction (synclevel=syncpt) from different systems.



RRS Considerations for IMS Customers

- Performance will be impacted by using RRS
  - RRS logs many activities related to the RRS events and services.
  - RRS ARCHIVE logging is the major one. Customer can delete the ARCHIVE logstream.
- Waits due to RRS processing can occur in dependent regions on the IMS control region.
  - Dump the RRS asid when they have a wait/hang
- ABENDU0711 could occur when a system error related to RRS is detected, even when the RRS is not used for the 2-phase commit.

RRS Diagnostic Info

- RRS CTRACE
  - Start by issuing 'TRACE CT,100M,ON,COMP=SYSRRS' or larger
    - Rxx,OPTIONS=(EVENT(ALL))
  - Stop by issuing 'TRACE CT,OFF,COMP=SYSRRS'
- RRS Panel
  - look at the status of URs and RMs in ISPF
- IMS Logrecords (4098, 5615, 5616, and 67D0)
- IMS /DIS UOR command
  - Shows outstanding UORs that IMS knows about
- /TRA SET ON TABLE RRST command in a V8 SPE APAR
  - Trace abnormal RRS events in IMS
- Reference
  - MVS Programming: Resource Recovery GC28-1739
  - IMS FAST manual for ABENDU0711 return codes

e-business

RRS Diagnostic Info ...

- ABENDU0711:
  - When U0711 is a pseuoabend
    - A type 67D0 logrecord will be cut
    - DFS554A will be issued
    - Application terminates, dependent region does not
    - In most cases, the trans and program will need to be restarted
  - When U0711 is a stardard abend
    - A dump of dependent region or control region will be produced

IMS RRS= parm

- **To disable/enable the RRS connection via a CTL region RRS= parm**
  - PQ62874 for V7
  - PQ62873 for V8
- **To disable/enable the batch support for RRS**
  - PQ59157 for V7
  - PQ51895 for V8

Summary

- z/OS Resource Recovery allows an application program to make consistent changes to multiple protected resources.
- OTMA/APPC synchronous synclevel = syncpt message, "protected transaction", is supported by using RRS.
- IMS RRS= parm were introduced to disable/enable the RRS function for IMS.
- IMS V8 OTMA/APPC synchronous messages (CM1) are allowed to be processed by any backend IMS using RRS.
- IMS V8 OTMA will support protected transactions from different systems with RRS multisystem cascaded feature.