z/OS

MVS Programming: Resource Recovery

Version 2 Release 2

SA23-1395-02

Note
Before using this information and the product it supports, read the information in "Notices" on page 661.

This edition applies to Version 2 Release 2 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this document

This document describes RRS, the system-level resource recovery platform for z/OS. The document describes two major tasks:

1. How to use RRS services in authorized resource managers, such as database programs, and communications managers that manage distributed transactional communications.
2. How to manage RRS in an installation that runs resource managers that use RRS.

Who should use this document

This document is for:

1. Programmers who design and code resource managers. These programmers need to know how to work with MVS™ system interfaces and how to work with databases.
2. System programmers responsible for managing MVS, including such tasks as starting and stopping system functions and troubleshooting. Database administrators might also be involved in managing RRS.

How to use this document

If you are responsible for managing MVS or administering databases at your installation, you will find most of the information you need in Chapter 1, "Introducing resource recovery," on page 1 and Chapter 8, "RRS setup and control," on page 535.

If you design and code a resource manager, Chapter 1, "Introducing resource recovery," on page 1, which includes "Planning a resource manager" on page 18, provides concepts and a list of planning considerations for coding a resource manager that works with RRS. Detailed programming information appears in:

- Chapter 2, "Using registration services," on page 21
- Chapter 3, "Using context services," on page 31
- Chapter 4, "Using resource recovery services," on page 51
- Chapter 5, "Callable registration services," on page 137
- Chapter 7, "Callable resource recovery services," on page 227

Note: If you are an application programmer interested in using RRMS services, you will also find information in *z/OS MVS Programming: Callable Services for High-Level Languages*.

If you want to code a program that checks information about RRS or manipulates RRS information directly, then two macros might be useful:

- Chapter 11, "ATRQUERY — Obtain RRS Information," on page 595
- Chapter 12, "ATRSRV — Resolve Units of Recovery," on page 627

This document is one of the set of programming documents for MVS. This set describes how to write programs in assembler language or high-level languages, such as C, FORTRAN, and COBOL. For more information about the content of this set of documents, see *z/OS V2R2 Information Roadmap*.

Note: If you call the services described in this document from assembler language programs, you must use a high-level assembler.

Where to find more information

Where necessary, this document references information in other documents, using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see z/OS Internet Library (http://www.ibm.com/systems/z/os/zos/bkserv/).

You might also need the following information:

Short Title Used in This Document	Title	Order Number
SNA Sync Point Services Architecture	*Systems Network Architecture Sync Point Services Architecture Reference*	SC31-8134

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R2 MVS Programming: Resource Recovery
  SA23-1395-02
- The topic and page number that is related to your comment.
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS Support Portal (http://www-947.ibm.com/systems/support/z/zos/).

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for z/OS Version 2 Release 2 (V2R2) as updated March 2016

The following changes are made for z/OS® V2R2 as updated March 2016.

Changed

- Table 25 on page 153 in "Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF)" on page 149 has been updated.
- Programming requirements for all Chapter 5, "Callable registration services," on page 137, "Retrieve_Resource_Manager_Data (CRGRRMD, CRG4RRMD)" on page 144, and Chapter 7, "Callable resource recovery services," on page 227have been updated for clarity.

Summary of changes for z/OS MVS Programming: Resource Recovery for Version 2 Release 2

The following changes are made for z/OS Version 2 Release 2 (V2R2).

New

- Table 19 on page 102 in "Parameters" on page 101 has been updated with a new bit 14.

Changed

- Table 25 on page 153 in "Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF)" on page 149 has been updated.

z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Introduction and Release Guide*

Chapter 1. Introducing resource recovery

Many computer resources are so critical to a company's work that the integrity of these resources must be guaranteed. If changes to the data in the resources are corrupted by a hardware or software failure, human error, or a catastrophe, the computer must be able to restore the data. These critical resources are called **protected resources** or, sometimes, **recoverable resources**.

Resource recovery is the protection of the resources. 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. Resources that z/OS can protect include:

- A hierarchical database
- A relational database
- A product-specific resource

This section describes how to provide resource recovery for resources on a single system and for resources distributed across multiple systems. The topics are:

- "Resource recovery programs"
- "Two-phase commit protocol" on page 4
- "Resource recovery functions" on page 3
- "Distributed resource recovery" on page 8

"Planning a resource manager" on page 18 lists information you need if you are thinking of providing a resource manager to work with RRS.

Resource recovery programs

To understand how to use resource recovery on the z/OS platform, you need to understand both the programs that work together and something of how they work together. First, you need to know 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:

- **Application program**: The application program accesses protected resources and requests changes to the resources.
- **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) that allows 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. There are several types of resource managers; see "Types of resource managers" on page 3.

Note: 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 termination processing.

- **Syncpoint manager**: Resource recovery services (RRS) is the syncpoint manager. It uses a two-phase commit protocol, described in "Two-phase commit protocol" on page 4 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.

Two other operating system components also play key roles in resource recovery:

- **Registration services**: Registration services coordinates communication between the resource manager and the exit managers. A resource manager must register itself with the system as a resource manager. The resource manager must also set its exit routines with each exit manager; the resource manager identifies the exit manager and the exit routines it provides for resource recovery. Registration services is itself an exit manager, though it drives only one exit routine. There is more information in Chapter 2, "Using registration services," on page 21.
- **Context services**: Context services is an exit manager that provides the data constructs and primitives that resource managers can use as an anchor for a given work request to track specific events related to the work request. For example, when a given context ends, context services drives the end-context exit routines for each resource manager involved. There is more information in Chapter 3, "Using context services," on page 31.

Registration services, context services, and resource recovery services (RRS) are three separate MVS components, but it is sometimes useful to think of them as a single function called recoverable resource management services (RRMS), the z/OS syncpoint manager.

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

Because RRS provides much of the resource recovery function (syncpoint processing, in particular), technical information, like this book, often uses the term RRS unless specifically describing context services or registration services.

RRS can enable resource recovery on a single system or, with a communications manager such as APPC/MVS, on multiple systems. If the resources used by an application program are distributed, so that they are on multiple systems, a communications resource manager on each system works with the syncpoint

manager on that system. The communications resource managers, in cooperation with the syncpoint managers, work together to coordinate the entire set of changes.

Types of resource managers

There are three types of resource managers:

Data Resource Manager: A resource manager that allows the application to read and change data. Data resource managers include database managers, such as DB2®, and record file managers, such as VSAM. To process a syncpoint event, a data resource manager would take actions such as committing or backing out changes to the data it manages.

Communications Resource Manager: A resource manager that controls access to distributed resources and acts as an extension to the syncpoint manager. A communications resource manager provides access to distributed resources by allowing an application to communicate with other applications and resource managers, possibly located on different systems. It acts as an extension to the syncpoint manager by allowing the local syncpoint manager to communicate with other syncpoint managers as needed to ensure coordination of the distributed resources the application accesses. Communications resource managers include APPC/MVS and Transactional Remote Procedure Calls (TRPC). To process a syncpoint event, a communication resource manager communicates the event to the distributed syncpoint managers.

Work Manager: A work manager is a resource manager that controls application access to system resources by determining when and in what environment the application runs. Work managers include CICS® Transaction Server and IMS™ Transaction Monitor. To process a syncpoint event, a work manager might ensure that the application is in the correct environment to allow the syncpoint processing to continue.

Note that a single resource manager can be more than one type. For example, IMS is both a data resource manager and a work manager.

Resource recovery functions

Resource recovery based on the two-phase commit protocol has two functions:

- **Commit**: During the commit process, all changes to both local and distributed resources are made permanently.
- **Backout**: During the backout process, all pending changes to both local and distributed resources are not made.

The set of changes that are to be made or not made as a unit are called a **unit of recovery (UR)**. A UR represents an application program's changes to resources since the last commit or backout or, for the first UR, since the beginning of the application. Each UR is associated with a **context**, which consists of the UR, or more than one UR, with the associated application programs, resource managers, and protected resources.

A context, which is sometimes called a work context, represents a work request. The life of a context consists typically of a series of URs. Figure 1 on page 4 shows the relation of the context, application program, and URs.

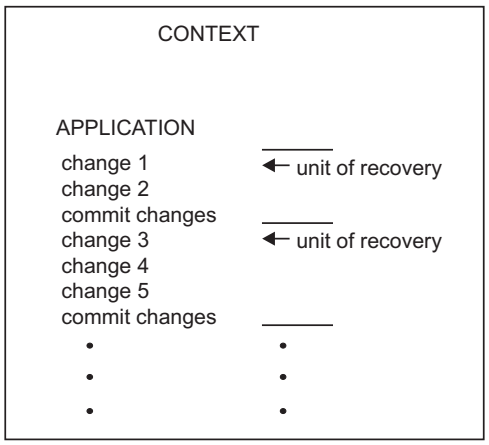

Figure 1. Context as a Series of URs

Two-phase commit protocol

When the application is ready to commit or back out its changes, the application invokes RRS to begin the two-phase commit protocol.

The two-phase commit protocol is a set of actions used to make sure that an application program makes all changes to the collection of resources represented by a UR or makes no changes to the collection. The protocol verifies the **all-or-nothing changes** even if the application program, the system, RRS, or a resource manager fails.

The phases of the protocol are:

- **Phase 1**: In the first phase, each resource manager prepares to commit the changes. A resource manager typically prepares by writing the unchanged data image, often called **undo data**, and the changed data image, often called **redo** data, in a resource manager log that it can access during restart.

  If the resource manager can then commit the changes, it tells RRS that it agrees to allow the commit to continue. If the resource manager cannot commit the changes, it tells RRS to back out the changes.

  The decision to commit or back out the changes represented by a UR depends on responses from all of the resource managers. If the decision is to commit the changes, RRS hardens the decision, meaning that it stores the decision in an RRS log, and phase 2 begins. If the decision is to back out the changes, RRS generally does not harden the decision, and phase 2 begins as soon as the decision is made.

  Once a commit decision is hardened, the application changes are considered to be committed. If the application, the system, RRS, or a resource manager fails after the decision is hardened, the application changes will be made during restart. Before the decision is hardened, a failure of the application, the system, RRS, or a resource manager would cause the changes to be backed out during restart.
- **Phase 2**: In the second phase, the resource managers commit or back out the changes represented by a UR.

Figure 2 on page 5 shows the valid states that can occur as RRS processes a UR. Each arrow represents a valid state change. Figure 2 on page 5 also shows how

valid state changes correspond to the phases of the two-phase commit protocol. You can find definitions of each state in “UR states” on page 64.

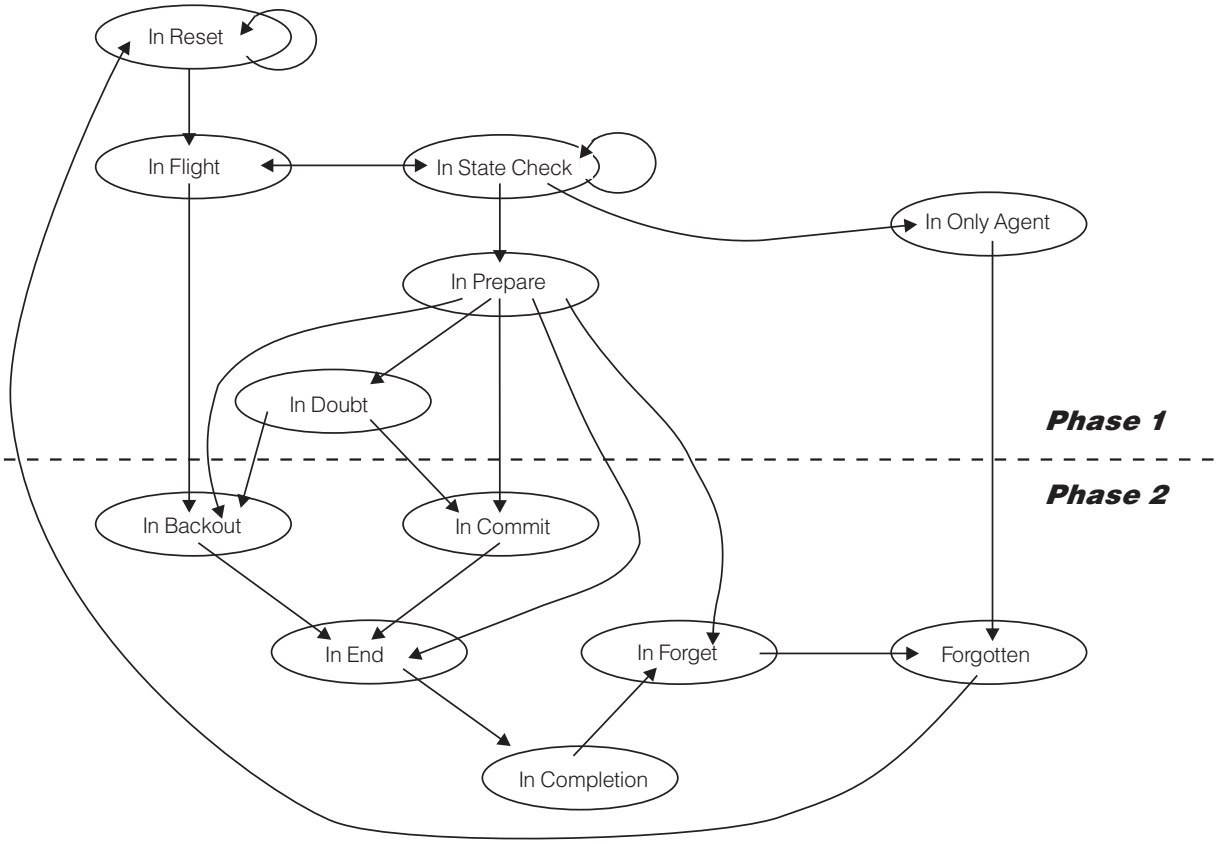

Figure 2. UR State Transitions

You can see examples of how the two-phase commit protocol works, including state changes, in “Commit” and “Backout” on page 7.

Commit

For a look at the commit function, think of a person who requests an automated teller machine (ATM) to transfer money from a savings account to a checking account. The application program receives the person's input from the ATM. Each account is in a different database. Each database has its own resource manager. The syncpoint manager is RRS. Figure 3 on page 6 shows how the ATM application, resource managers, and RRS work together

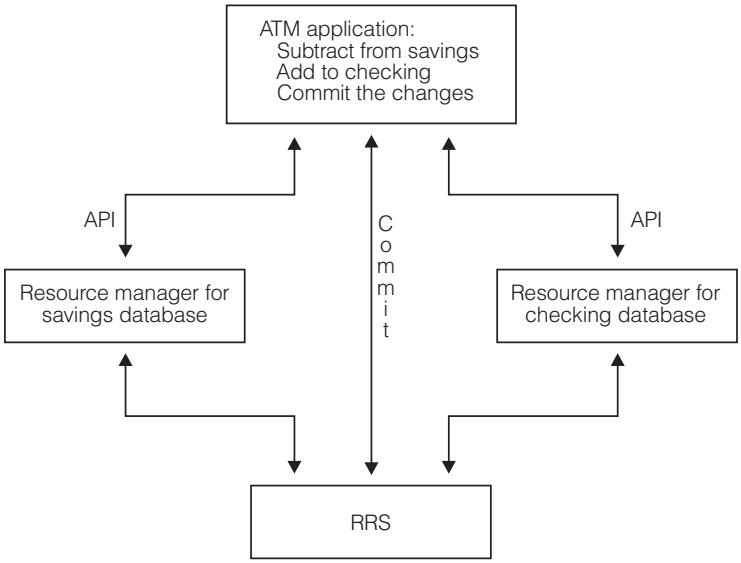

Figure 3. ATM Transaction

The actions required to process the ATM transaction are:

1. The ATM user requests transfer of money from a savings account to a checking account.
2. The ATM application program receives the ATM input.

   Figure 4 on page 7 shows, for the same transaction, the sequence of the following actions, with time moving from left to right, in the two-phase commit protocol RRS uses to commit the changes. The top lines in the figure show the state of the UR and the two phases of the two-phase commit protocol. For more information, see “UR states” on page 64 and “Two-phase commit protocol” on page 4.
3. The ATM application requests the savings resource manager to subtract the money from the savings database. For this step, the application uses the resource manager's application programming interface (API).
4. The ATM application requests the checking resource manager to add the money to the checking database. The application uses this resource manager's API.
5. The ATM application issues a call to RRS to commit the database changes.
6. RRS asks the resource managers to prepare for the changes.
7. The resource managers indicate whether or not they can make the changes, by voting YES or NO. In Figure 4 on page 7, both resource managers vote YES.
8. In response, RRS notifies the resource managers to commit the changes, that is, to make the changes permanently in the databases.
9. The resource managers complete the commit and return OK to RRS.
10. RRS gives a return code to the application program, indicating that all changes were made in the databases.

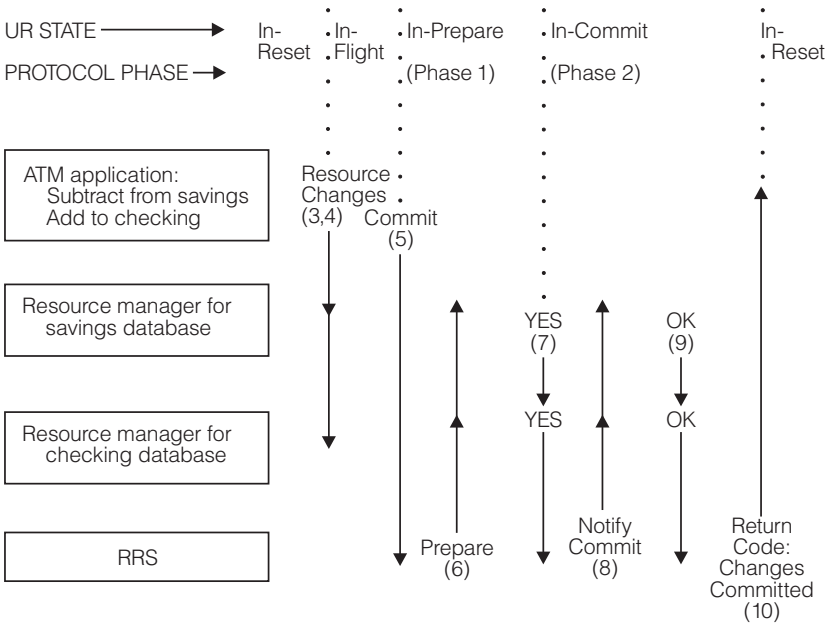

Figure 4. Two-Phase Commit Actions

Backout

If, for any reason, the ATM application cannot complete the transfer, the application requests backout in step 5, instead of commit. In this case, the changes are backed out and are not actually made in any database. See Figure 5.

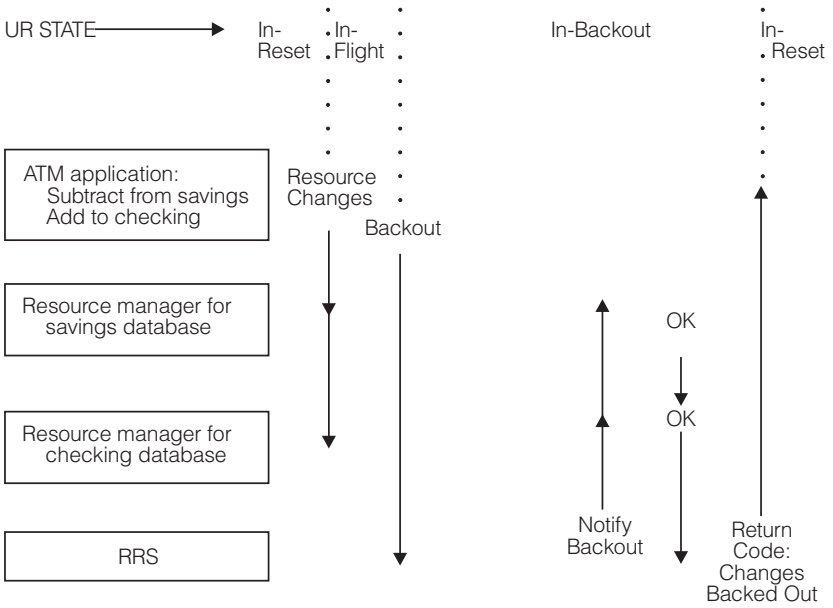

Figure 5. Backout — Application Request

A resource manager can also request backout. If a resource manager cannot make the change to its database, the resource manager votes NO during prepare. If **any** resource manager votes NO, all of the changes are backed out. See Figure 6 on page 8

page 8.

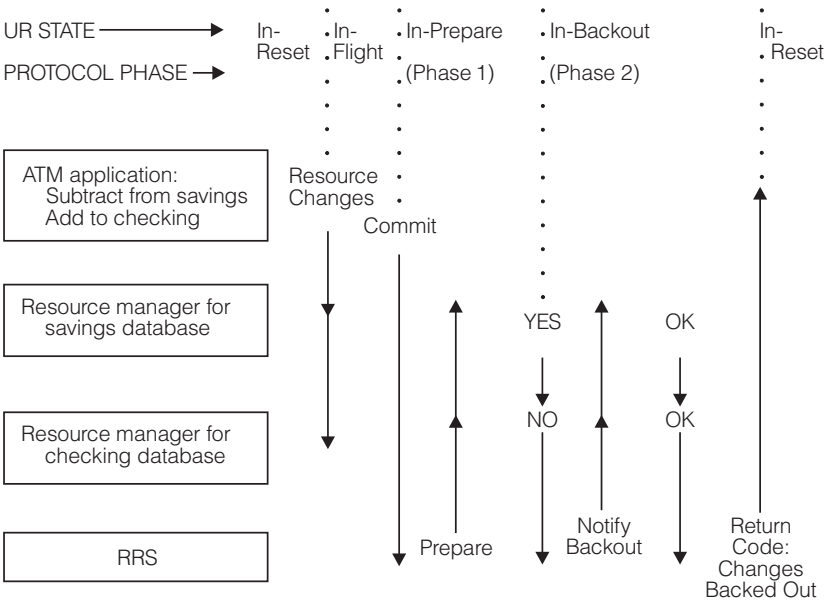

Figure 6. Backout — Resource Manager Votes NO

Distributed resource recovery

The resources that a work request updates can be distributed — reside on more than one system. Figure 7 on page 9 shows the programs that participate in distributed resource recovery, also called distributed syncpoint processing.

When resources reside on multiple systems, a part of the application must run on each system. Using the ATM example presented earlier. assume that the resource manager for the savings account database runs on system A, and the resource manager for the checking account database runs on system B. We know from the earlier example that the ATM application has three main responsibilities:

1. Communicate with the ATM user
2. Update the savings account database using the resource manager on system A
3. Update the checking account database using the resource manager on system B

Assume that the part of the application running on system A (APPL-A) always communicates with the ATM user.

When the ATM user requests the transfer of money from savings to checking, APPL-A sees the request, and uses the resource manager on system update the savings account database. Completing the transaction means that APPL-A must communicate with the resource manager on system B to update the checking account database. Thus, part of the application (APPL-B) must reside on system B.

APPL-B listens for a signal from APPL-A, and the signal tells APPL-B to use the resource manager on system B to update the checking account database. Another way to implement the application would be for the signal from APPL-A to actually initiate APPL-B.

In either case, APPL-A communicates with APPL-B through a communications resource manager (CRM), such as APPC/MVS. The main function of a CRM is to

allow applications to communicate across systems, but it also allows RRS on one system to communicate with RRS on another system.

Keep this brief description of distributed processing in mind. It applies to the two models for processing distributed resources: the peer-to-peer model and the client-server model. Which model is appropriate depends on the application and how its resources are distributed.

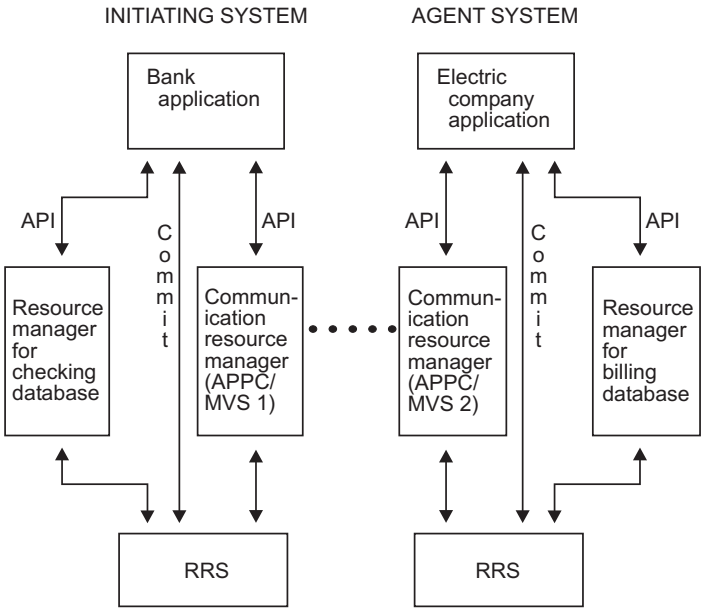

Figure 7. Transaction — Syncpoint Processing

Peer-to-peer model

Using the peer-to-peer model, all systems are equal until an application program running on a system issues the commit request that begins the syncpoint operation. The fact that any system can initiate the syncpoint operation is an important attribute of the peer-to-peer model.

A good place to begin a description of the peer-to-peer model is with a more detailed description of what happens when the ATM user mentioned earlier requests a transfer of money from a savings account to a checking account. Figure 8 on page 10 lists the processing steps.

1. The ATM user requests a transfer of money from a savings account to a checking account. The savings account database resides on one system (system A), while the checking account database resides on another system (system B).
2. The ATM application program (APPL-A) on system A receives the input from the ATM user.
3. APPL-A requests that the savings resource manager on system A subtract the money from the savings database.

   APPL-A requests the checking resource manager on system B to add the money to the checking database. To make this request, APPL-A communicates with APPL-B (the part of the ATM application that runs on system B). APPL-B tells the resource manager on system B to add the money to the checking account database.
4. APPL-A calls RRS to commit the database changes.
5. RRS on system A asks both resource managers (the one on system A and the one on system B) to prepare for the changes.
6. Both resource managers indicate to RRS on system A whether or not they can make the changes by voting YES or NO. For this example, assume that both vote YES.
7. In response, RRS on system A notifies both resource managers to make the changes permanent, which is called committing the changes.
8. The resource managers complete the commit and return OK to RRS on system A.
9. RRS on system A issues a return code to APPL-A to indicate that its commit request was successful; the databases have been changed.

Figure 8. Peer-to-Peer Processing

As Figure 8 describes, APPL-A must be able to communicate with APPL-B. It is also true, though less obvious in the example, that RRS on system A must be able to communicate with RRS on system B.

Note that the **initiating system** is the system on which the commit request is first issued. In the example, the initiating system is system A. Every other system (such as system B in the example) is an **agent system**. The communications resource manager (CRM) that runs on an agent system is called an **agent CRM**.

Figure 8 presents a high-level example of peer-to-peer processing. To help explain the role of the communications manager, this book uses APPC/MVS as the communications resource manager, but any communications resource manager that implements the peer-to-peer model would perform the same processing. Figure 9 on page 11 adds communication processing to the high-level example.

1. The ATM user requests a transfer of money from a savings account to a checking account. The savings account database resides on one system (system A), while the checking account database resides on another system (system B).
2. The ATM application program (APPL-A) on system A receives the input from the ATM user.
3. APPL-A requests that the savings resource manager on system A subtract the money from the savings database.

   APPL-A requests the checking resource manager on system B to add the money to the checking database. To make this request, APPL-A communicates with APPL-B (the part of the ATM application that runs on system B). APPL-B tells the resource manager on system B to add the money to the checking account database.
4. APPL-A calls RRS to commit the database changes. ***Because the application on system A requests the commit, system A becomes the initiating system.***
5. RRS on system A asks both resource managers (the one on system A and the one on system B) to prepare for the changes:
   - ***RRS on system A collects prepare votes from resource managers on system A. These votes are called local prepare votes.***
   - ***RRS on system A tells RRS on system B (the agent system) to collect prepare votes from resource managers on system B. These s are vote called distributed prepare votes. To collect these distributed prepare votes:***
     - ***RRS on the initiating system (system A) tells APPC on A to system notify the application on the agent system (system B) of the need to commit the changes. APPC on system A contacts APPC onand APPC system B, on system B tells APPL-B that a commit is needed.***
     - ***APPL-B calls RRS on system B, requesting that the changes be committed.***
     - ***RRS on system B, the agent system, collects distributed prepare votes from the resource managers on system B.***
6. Both resource managers indicate to RRS on system A whether or not they can make the changes by voting YES or NO. For this example, assume that both vote YES:
   - ***RRS on system A, the initiating system, determines the outcome of the local prepare votes.***
   - ***RRS on system B, the agent system, tells RRS on system A the result of the distributed prepare votes, using APPC to communicate with RRS on system A.***

   RRS on system A, the initiating system, makes the final decision to commit or back out the resource changes, basing the decision on the local prepare votes and the result of the distributed votes. For this example, assume that the overall result is to commit the resource changes.
7. In response, RRS on system A notifies both resource managers to make the changes permanent, which is called committing the changes:
   - RRS on the initiating system (system A) tells the resource managers on system A to commit the changes.
   - RRS on system A uses APPC to tell RRS on system B, the agent system, that the final decision is to commit the resource changes.
8. The resource managers complete the commit and return OK to RRS on system A:
   - ***The resource managers on system A tell RRS on system A that the commit is complete.***
   - ***The resource managers on system B tell RRS on system B that the commit is complete, and RRS on system B uses APPC to tell RRS on system A that the commit is complete on the agent system.***
9. RRS on system A issues a return code to APPL-A to indicate that its commit request was successful; the databases have been changed.

Figure 9. Communication Processing

The initiator is responsible for the overall decision. The agents are responsible for collecting local votes and distributing the decision to the local coordinator (RRS). When APPC/MVS becomes an agent, it informs RRS by taking the **distributed syncpoint resource manager** (DSRM) role, which makes it responsible for informing RRS of the commit decision.

RRS on each MVS system provides the two-phase commit protocol for the resource managers on its system. Each RRS collects the prepare votes from the local resource managers, then returns the collective vote to the DSRM.

When a work request is distributed rather than local, the two-phase commit processing is slightly different: on the agent system, the UR state becomes **in-doubt** at the end of phase 1. The state remains **in-doubt** until the initiator collects all the votes and returns a commit or backout signal to the DSRM on the agent system.

Remember that RRS is an exit manager; it drives resource manager exit routines in response to resource recovery events. For example, when an application requests commit, RRS drives the PREPARE and COMMIT exit routines for each resource manager involved.

To put all the pieces together, assume a transaction where a user requests the transfer of money from a checking account to the electric company to pay an electric bill. The actions required to process this sample transaction are:

1. The user of a computer connected by a modem to the bank's computer requests transfer of money from a checking account to the electric company to pay an electric bill.
2. The bank application program receives the user's input.

   Figure 10 on page 14 shows, for this transaction, the sequence of the following actions, with time moving from left to right, in the two-phase commit protocol with distributed resource recovery. The top lines of the figure show the states for each UR as it moves through the two-phase commit protocol. For more information, see "UR states" on page 64 and "Two-phase commit protocol" on page 4.
3. The bank application:
   - Requests the checking resource manager to subtract the money from the checking database
   - Allocates a conversation to the electric company application to receive the money as payment for the electric bill
4. The electric company application requests the billing database resource manager to add the money to the user's account.
5. The bank application issues a call to RRS 1 to commit the checking database changes.
6. RRS 1 asks the checking resource manager to prepare for the changes. The PREPARE exit routine votes YES.

   If any vote is NO, all changes are backed out on all systems.
7. RRS 1 also sends a PREPARE signal through APPC/MVS 1 to APPC/MVS 2. In response to the PREPARE signal:
   a. APPC/MVS 2 informs RRS 2 that it is assuming the role of distributed syncpoint resource manager (DSRM). RRS 2 does not complete the commit; that is, RRS 2 does not drive COMMIT exit routines until told to do so by the DSRM. (This action is related to the processing described for step 10).
   b. APPC/MVS 2 informs the electric company application that a commit has been requested.
8. The electric company application issues a call to RRS 2 to commit the billing database changes.
9. RRS 2 asks the billing resource manager to prepare for the changes. The PREPARE exit routine votes YES.

   If any vote is NO, all changes are backed out on all systems.
10. The resources are distributed, so RRS 2 cannot make a unilateral commit; it must synchronize its commit with RRS 1 to ensure that all changes are made or no changes are made.

    Because APPC/MVS 2 took the DSRM role earlier, RRS 2 does not call COMMIT exits. Instead, RRS 2 collects the votes on system 2. If all resource managers vote yes, then the local collective vote is to commit the changes, and RRS 2 tells APPC/MVS 2 to send a REQUEST_COMMIT signal through APPC/MVS 1 to RRS 1.

    If any resource manager on the electric company's system (the agent system) votes not to commit the resource, then the local collective vote is NO, and RRS 2 sends a REQUEST_BACKOUT signal.
11. RRS 1 notifies the checking resource manager to commit the changes. The checking resource manager completes the commit and returns OK to RRS 1.

For a REQUEST_BACKOUT signal, RRS 1 notifies the checking resource manager to backout the changes. The changes are not made.

12. RRS 1 notifies APPC/MVS 1 to commit the changes. APPC/MVS 1 sends a COMMITTED signal through APPC/MVS 2 to RRS 2.
13. RRS 2 notifies the billing resource manager to commit the changes. The billing resource manager completes the commit and returns OK to RRS 2.
14. RRS 2 informs APPC/MVS 2 that RRS 2 has finished its processing for the UR. APPC/MVS 2 sends a FORGET signal through APPC/MVS 1 to RRS 1; the signal means that APPC/MVS 1 can do clean up for the UR.
15. RRS 1 has finished its processing for the UR and informs APPC/MVS 1.
16. RRS 1 and RRS 2 give return codes to the application programs, indicating that all changes were made in the databases.

At this point, you might want to know more about how RRS uses APPC to communicate across systems. The basic mechanism is the exit routines APPC sets with RRS.

APPC registers with RRS as a resource manager, just like any resource manager. Thus, RRS will drive APPC's PREPARE and COMMIT exit routines when the bank application and the billing application issue their commit requests.

Unlike the other resource managers, however, APPC has no database to update. Instead, APPC is responsible for communication. Thus, APPC uses its COMMIT and PREPARE exit routines to kick off communications, not to update databases:

- The PREPARE signal described in step 7 on page 12, for example, is sent by APPC's PREPARE exit, which then waits for a response from the other system.
- The REQUEST_COMMIT signal described in step 10 on page 12 is sent by APPC's DISTRIBUTED_SYNCPOINT exit routine, which then waits for a response.

After APPC's PREPARE exit receives the response, the COMMITTED signal described in step 12 is sent by APPC's COMMIT exit routine, which then waits for a response.

After APPC's waiting DISTRIBUTED_SYNCPOINT exit routine receives the response, the FORGET signal described in step 14 is sent by APPC's END_UR exit routine and received by its waiting COMMIT exit.

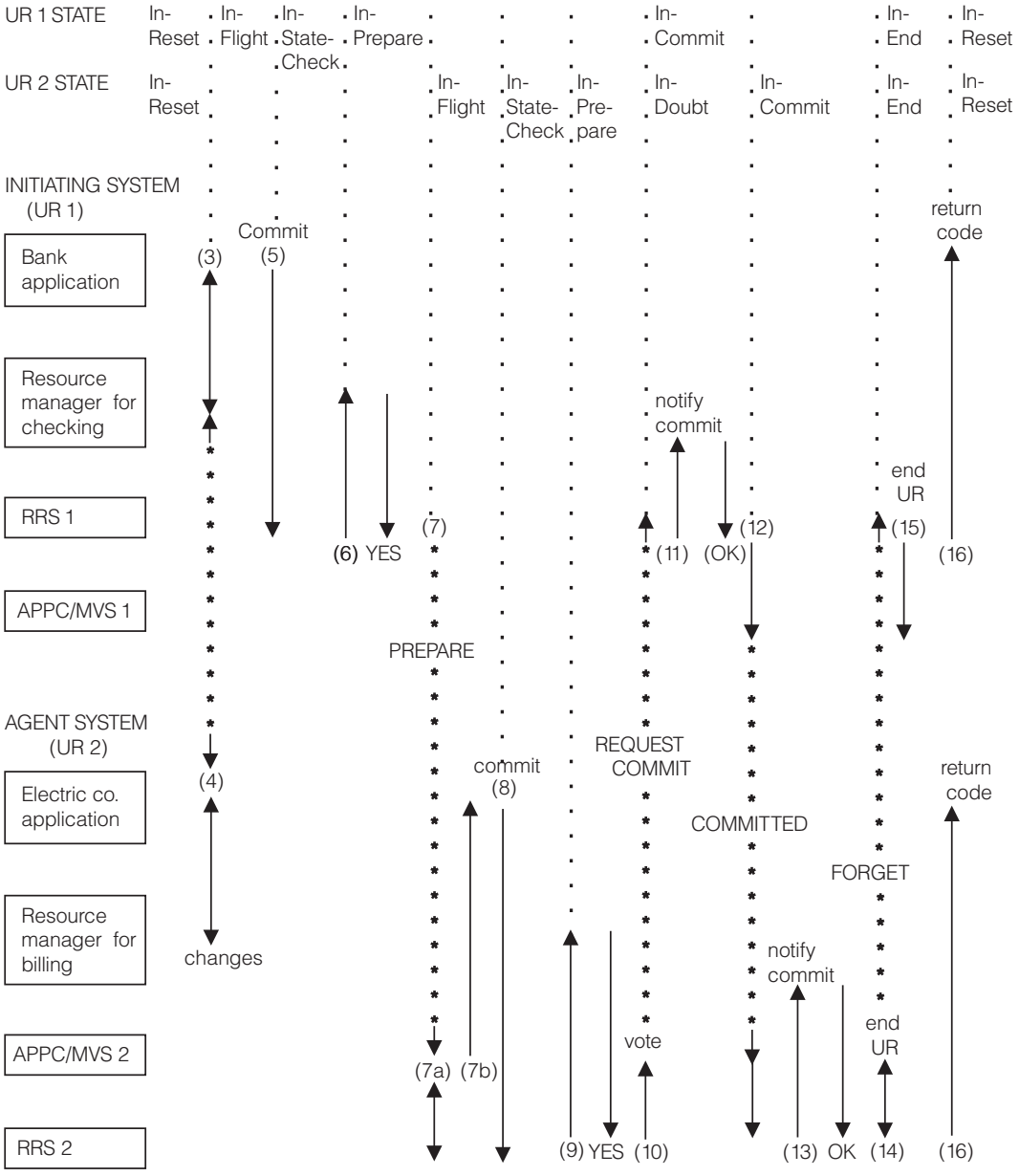

Figure 10. Syncpoint Processing — Peer-to-Peer

Client-server model

With the client-server model, there can be one client system, which is always the initiating system, and many server systems, which are always agent systems.

The client-server model provides a generic, or flexible, approach to distributed syncpoint processing. It is used, for example, by Transactional Remote Procedure Call (TRPC), a communications protocol, and might be suited for many other uses.

Figure 11 on page 15 shows the high-level flow of syncpoint processing using the client-server model. The dotted lines link the functions provided on each system.

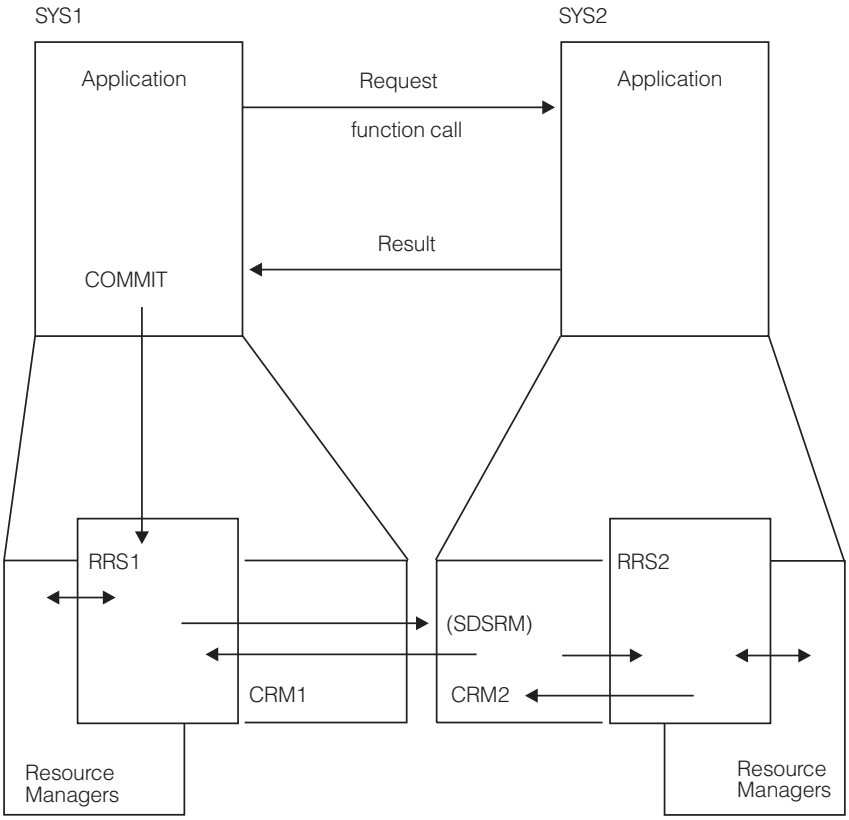

Figure 11. Client-Server — High-Level Flow

In Figure 11, note that the application on SYS1, the client system, initiates the syncpoint by sending a function call to SYS2 to request resource changes. When SYS2 sends a successful result back, the application on SYS1 calls RRS on SYS1 to commit the changes. RRS1 sends the request through the communications resource manager (CRM1) to SYS2. CRM2 takes the **server distributed syncpoint resource manager** (SDSRM) role and, acting as an agent, calls RRS2 on SYS2 to collect the votes on SYS2, then returns the result to RRS1 on SYS1.

To understand client-server syncpoint processing in more detail, assume a transaction where a user requests the transfer of money from a checking account to a savings account. The transaction requires updates to protected resources on two systems:

1. UR 1 represents the changes on the first system, the initiating system, where RM 1 manages the checking account database, and CRM 1 manages the communications protocols.
2. UR 2 represents the changes on the second system, the agent system, where RM 2 manages the savings account database. CRM 2 on the second system takes the role of the server distributed syncpoint resource manager (SDSRM) and manages the communications protocols on the agent system.

The actions required to process this sample transaction follow. Figure 12 on page 17 shows, for this transaction, the sequence of these actions, with time moving from left to right. The top lines of the figure show the states for each UR as it moves through the two-phase commit protocol. For more information, see "UR states" on page 64 and "Two-phase commit protocol" on page 4.

1. After making update requests to both RM 1, the resource manager for checking on the initiating system, and RM 2, the resource manager for savings on the agent (server) system, the bank application issues a call to RRS 1 to commit the distributed transaction.
2. RRS 1 tells RM 1, the resource manager for checking, to prepare its changes. RM 1 prepares the resources for commit, then indicates to RRS 1 that it is ready to commit the changes.
3. RRS 1 tells CRM 1 to prepare its resources for the commit. That is, RRS 1 drives the CRM 1 prepare exit routine.
4. From its prepare exit routine, CRM 1, the communications resource manager on the initiating system, sends a PREPARE signal to CRM 2, the communications resource manager on the agent, or server, system.
5. CRM 2, on the agent, or server, system, calls the Prepare_Agent_UR service to tell RRS 2 to initiate the prepare phase for UR 2, which represents the requested change to the savings account.
6. RRS 2 then tells RM 2, the resource manager for savings, to prepare its resources for commit. RM 2 prepares its resources and tells RRS 2 that it is ready to commit the changes.
7. RRS 2 returns the local collective vote results to CRM 2. For this example, assume that the local collective vote is to commit the changes.
8. CRM 2 sends a REQUEST_COMMIT signal to CRM 1 to request that the initiating system commit the changes.
9. Because CRM 1 received a REQUEST_COMMIT signal, it tells RRS 1 that the resources on the agent system can be committed. RRS 1 determines the overall results. For this example, the overall collective vote is to commit the changes; RRS 1 hardens the commit decision. That is, RRS 1 writes a record to its log to indicate that the state of UR1 is now **in_commit**.
10. RRS 1 tells RM 1, the resource manager for checking, to commit its changes. RM 1 commits its changes.
11. RRS 1 drives the CRM 1 commit exit routine to tell CRM 1 to commit its resources.
12. From its commit exit routine, CRM 1 sends a COMMIT signal to CRM 2 to indicate that the initiating system has committed the changes.
13. CRM 2, acting as the SDSRM, calls the Commit_Agent_UR service to tell RRS 2 that the overall decision is to commit all resources. RRS 2 hardens the commit decision. That is, RRS 2 writes a record to its log to indicate that the state of UR 2 is now **in_commit**.
14. RRS 2 tells RM 2, the resource manager for savings, to commit its changes. RM 2 commits the changes.
15. RRS 2 then returns to CRM 2 with the information that the local resources, the resources on the agent, or server, system, are committed.
16. CRM 2 sends a FORGET signal to CRM 1 to indicate that the initiating system can forget the UR, then calls the Forget_Agent_UR service to tell RRS 2 to delete its log record.
17. Because CRM 1 received a FORGET signal, it tells RRS 1 that its processing is complete. CRM 1 receives the FORGET signal and returns to RRS 1 from its commit exit routine.
18. RRS 1 then returns the results of the commit request to the bank application and deletes the log record for UR1. The state of UR 1 is now **forgotten**.

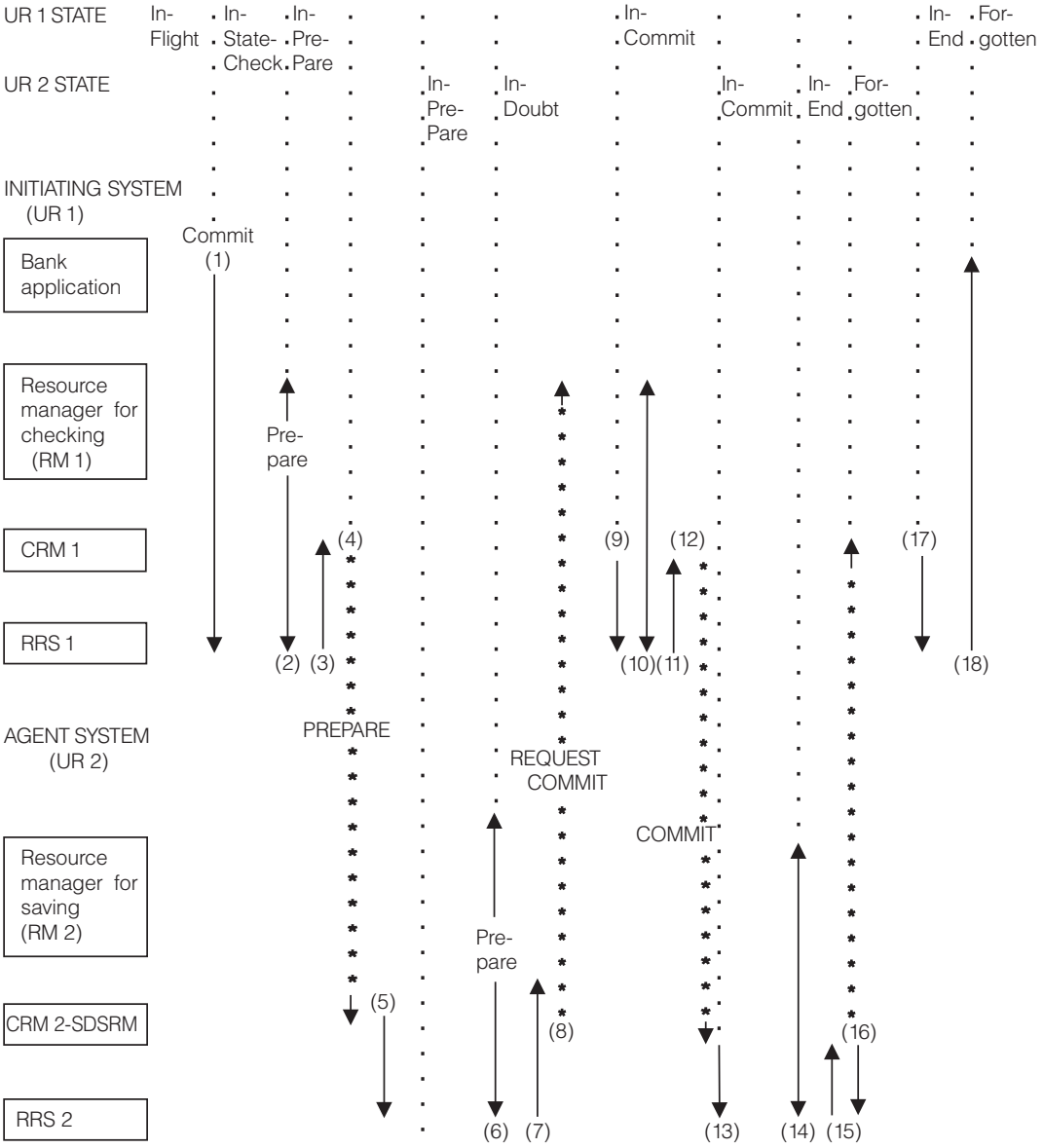

Figure 12. Syncpoint Processing — Client-Server

Heuristic decisions

Whether a syncpoint is local or distributed, following the two-phase commit protocol means that a decision to commit or back out a set of resource changes is, on one level, a holistic decision. That is, the decision to commit or back out is made by the participant designated to make the decision, based on input from the whole set of participants.

There are, however, occasions when a local or distributed resource manager might make a commit or backout decision on its own. This decision is called a **heuristic decision** because it is made by a resource manager that is not designated to make the final decision for the UR as a whole. Installation personnel usually are involved in making a heuristic decision. For example:

- Suppose a system involved in distributed resource processing is hung. Installation personnel use RRS panels to resolve any URs in an **in-doubt** state. The installation might commit the UR on one system, but the resource managers or installation personnel might back out the UR on another system.
- Database locks are being held too long because one or more resource managers or applications involved in a UR failed in a way that is not detectable by RRS. A heuristic decision must be made to free locked resources. Since RRS does not limit transaction duration, properly designed resource managers should track resource lock hold durations and either automatically make a heuristic decision after an excessive time has passed, or warn installation personnel that they should make a heuristic decision.

There are three possible heuristic conditions

- **Heuristic commit (HC)**: A heuristic decision to commit some of the protected resources associated with a UR.
- **Heuristic reset (HR)**: A heuristic decision to back out some of the protected resources associated with a UR.
- **Heuristic mixed (HM)**: Inconsistent commit or backout decisions for a UR. One of the decisions is heuristic.

Any heuristic condition is a problem because it often means that there have been inconsistent changes to protected resources. Resolving the problem might require manual intervention.

Planning a resource manager

A resource manager can control:

- Protected resources, which can be recovered if a failure occurs
- Unprotected resources

To control protected data resources, a resource manager must provide the following:

- An application programming interface (API) that an application program can use to read, write, and change resources
- A log of changes to the resource's data before the changes are permanent
- A log of the state of the work
- A commit action to make permanent changes to the resource manager's data
- A backout action to restore the resource manager's data to its previous contents

Resource managers for unprotected resources do not log data changes or provide actions for commit and backout. This information deals with the actions resource managers take to work with RRS to control protected resources.

While each resource manager that works with RRS to control protected resources will be different, all must consider a common set of decisions and actions, including the following:

1. Your resource manager must register (make itself known to the operating system) before it can work with exit managers, such as RRS, to protect resources. To register, the resource manager issues a call to the Register_Resource_Manager service. When registering, the resource manager can specify global data, which is passed to all of the resource manager's exit routines when they are invoked.

More information appears in Chapter 2, "Using registration services," on page 21, and you can find descriptions of each service in Chapter 5, "Callable registration services," on page 137.

2. Decide whether or not to set the notification exit routine with registration services. This exit routine, though optional, is important because it keeps your resource manager informed of events related to registration. It is driven when an exit manager becomes available or unavailable, and it is also driven when your resource manager exit routines have become unset. For information on the exit routine, see Chapter 2, "Using registration services," on page 21.
3. After registering, your resource manager must establish connections to each exit manager it needs. To connect to an exit manager, the resource manager issues one or more calls to the Set_Exit_Information service. Each call specifies one exit manager and identifies the exit routines that the specified exit manager is to invoke, or drive. Setting an exit routine means that the exit manager will drive the exit routine when the event occurs.

   The exit manager drives the exit routines in response to predefined events. For RRS, these predefined events are events that occur during resource recovery. For example, when an application commits the changes for a UR, RRS invokes the COMMIT exit routine for each affected resource manager. The exit routine sets a return code that determines how the exit manager continues to process the event.

   A resource manager must set its exit routines before an exit manager can invoke any of the routines.

   For information about each exit manager and the exit routines it can drive, see:

   - Chapter 2, "Using registration services," on page 21
   - Chapter 3, "Using context services," on page 31
   - Chapter 4, "Using resource recovery services," on page 51
4. Decide whether or not your resource manager needs to express interest in a work context. A *context* represents a business unit of work: one or more units of recovery with the associated application programs, resource managers, and protected resources. The context should be the anchor for the resource manager's control structures related to the work request.

   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 might need to express interest in a work context because a context can persist over multiple URs.

   A work context can be either a native context, which is associated with a single application task, or a privately managed context, which can be switched from one task to another. A privately managed context is usually used by a work manager, such as IMS.

   More information appears in Chapter 3, "Using context services," on page 31.
5. Decide how to use resource recovery services to implement the two-phase commit protocol in your resource manager, described in Chapter 4, "Using resource recovery services," on page 51.
6. You need to plan the actions your resource manager will take if a failure occurs.

   The effect of a resource manager failure depends on the scope of the failure, which can be either:

   - Exit manager scope
   - System scope

If a resource manager has registered, the state of the resource manager is **registered**. It is known to the system, and it can then set exits with one or more exit managers. For example, one resource manager might set exits with RRS while another might set exits with both RRS and context services. Once a resource manager sets exits with an exit manager, its state with that exit manager is **set**.

Exit Manager Scope: A failure that has exit manager scope occurs when the failure causes the resource manager's exits to be unset with a specific exit manager. The exit manager changes the resource manager state from **set** to **unset** and drives the resource manager's NOTIFICATION exit routine to inform the resource manager of the failure.

For example, assume that a resource manager has set its exits with RRS. During processing, RRS drives the resource manager's EXIT_FAILED exit routine, but the routine returns an unexpected return code. This unexpected return code causes RRS to unset the resource manager's exits and drive its NOTIFICATION exit routine. The resource manager state with the system is still **registered**, but its state with RRS is **unset** (though it continues to run and might remain set with another exit manager).

System Scope: In contrast, a failure that has system scope changes the resource manager state from **registered** to **unregistered**, either because the resource manager task or address space has failed, or because the resource manager has itself requested the state change. As a result, the resource manager state with all interested exit managers changes from **set** to **unset**. Before it can again participate in resource recovery, the resource manager must restart.

Chapter 2. Using registration services

Registration services allow a resource manager to define itself to the operating system. A **resource manager** is a program that controls and manages access to a resource.

Examples of resource managers are a database manager that works with RRS to protect resources, a work manager that uses the context services component, and a communications resource manager that handles protected communications, such as Advanced Program-to-Program Communication/MVS (APPC/MVS).

Note: The resource manager here is different from an RTM resource manager, which is related to the operating system's recovery termination management (RTM) and runs during termination processing.

As part of registration, a resource manager identifies itself and its exit routines, if any, to **exit managers**, which have registered with the system. An exit manager invokes an exit routine when a specific event occurs. Examples of exit managers are context services and resource recovery services.

The registration services are:

Callable service	Description
Register_Resource_Manager	Register a resource manager.
Retrieve_Resource_Manager_Data	Retrieve resource manager global data.
Set_Exit_Information	Identify resource manager and its exit routines, if any, to an exit manager.
Unregister_Resource_Manager	Unregister a resource manager.

For information on the calls, see Chapter 5, “Callable registration services,” on page 137.

An authorized resource manager can provide an exit routine to be invoked to notify the resource manager about events related to registration, such as when an exit manager becomes registered or unregistered, or when your resource manager exit routines have become unset. Although the notification exit routine is optional, it is a good idea to provide one; it can report events that are important to your resource manager. The registration services exit routine is:

Exit routine	Exit Number in: Hexadecimal (Decimal) Equate Symbol	Event
NOTIFICATION	1 (1) CRG_NOTIFICATION_EXIT	An exit manager has become registered or unregistered.

“NOTIFICATION exit routine” on page 23 describes the routine.

Registration

A resource manager must register every time it is started, regardless of whether the start is caused by failure of the system or the resource manager itself. If an exit manager fails, however, the resource manager does not need to register again, though it does have to call Set_Exit_Information to reidentify itself and to reset the exits for the exit manager that failed.

Registration is the same when starting for the first time, restarting after a normal shut down, or restarting after a failure. The sequence is:

1. Call the Register_Resource_Manager service.
2. Call the Set_Exit_Information service one or more times to identify the resource manager to appropriate exit managers and to identify all of the resource manager's exit routines.

In the call to the Register_Resource_Manager service, the resource manager identifies itself with a unique 32-byte name. Registration services checks whether or not the name is already registered on the current system; the check does not include other systems in the sysplex. If the name is not registered, registration services registers it and returns a 16-byte resource manager token. This token represents the resource manager and is required on many calls. The resource manager token is a random value that is not preserved across a restart of the system, exit manager, or resource manager. Thus:

- Do not use the resource manager token as an identifier in log records.
- Do not try to discern the contents of the token or create any dependencies on the contents.

Setting exit routines

A resource manager should call the Set_Exit_Information service one or more times to identify itself and the entry points for its exit routines for each exit manager. Each call specifies one exit manager and its exit routines.

Resource manager global data

During registration, the resource manager can provide global data. When an exit manager invokes an exit routine, the exit manager passes this global data to the routine. The global data should provide the exit routine with an anchor or anchors to the resource manager's data structures.

The global data is not preserved across a restart of the system, exit manager, or resource manager.

The resource manager can call the Retrieve_Resource_Manager_Data service to retrieve the global data.

Unregistration

A registered resource manager becomes unregistered as follows:

- The resource manager explicitly unregisters itself by a call to the Unregister_Resource_Manager service.
- Registration services implicitly unregisters a resource manager if:
  - The resource manager's task ends.
  - The cross memory resource-owning task of the resource manager ends.
  - The resource manager's address space ends.

When it registers, the resource manager chooses which of the preceding events applies.

- The system can implicitly unregister a resource manager because of errors, such as consecutive exit errors.

When the resource manager is unregistered, exit managers do not invoke its exit routines.

NOTIFICATION exit routine

If the exit manager specified in a call to the Set_Exit_Information service is not available, the system returns an error code. If and when the exit manager becomes available, the system gives control to the NOTIFICATION exit routine, if provided; the routine can reissue the call to the Set_Exit_Information service.

The NOTIFICATION exit routine has a second use. If an exit manager becomes unavailable, the system gives control to the NOTIFICATION exit routine, if provided.

Programming considerations

The following topics describe installing, invoking, processing, and returning for the exit routine and the action taken on an exit routine failure.

Installing an exit routine: To install the registration services exit routine, the resource manager must:

- Call the Register_Resource_Manager service.
- Set the NOTIFICATION exit routine or routines for a resource manager through one or more calls to the Set_Exit_Information service.

Invoking an exit routine: The system invokes a NOTIFICATION exit routine for the following events related to an exit manager:

- An exit manager that was not available when the Set_Exit_Information service was called becomes available
- A running exit manager becomes unavailable
- A running exit manager has changed the resource manager state to **unset**.

Note that some exit managers, such as context services, are always available.

The *notification_exit_type* parameter in the call to the Set_Exit_Information service specifies how registration services is to invoke the exit routine:

- **SRB routine**: The system schedules a service request block (SRB) at local priority in the resource manager's address space to give control to the exit routine.

  The exit routine may run synchronously or asynchronously. In either case, it will be nonpreemptable.

  A resource manager in a swappable address space must use SRB exit routines.
- **PC routine**: The system issues a stacking Program Call (PC) instruction to give control to the exit routine. The stacking PC must use a system LX so that the routine is available from all address spaces.

  Note: Consider carefully before deciding to use a system LX. Using a system LX improperly can prevent ASIDs from being reused, which can in turn cause unscheduled IPLs. To avoid unnecessary loss of ASIDs, IBM recommends that a resource manager use a system LX only when the resource manager is a

long-running address space. See "Reusing ASIDs" in *z/OS MVS Programming: Extended Addressability Guide* for more detail.

The exit routine will run synchronously; therefore, the resource manager should not suspend processing of the work unit. The system cannot invoke any other exit routines until the PC routine completes.

The resource manager must be in a nonswappable address space to use PC exit routines. A PC exit routine must remain available to the system until the resource manager ends processing, unregisters, or issues a call to the Set_Exit_Information service to change the exit routine.

A PC exit routine and any routine that it invokes cannot issue an SVC instruction.

The advantage of the PC routine over an SRB routine is a shorter path length to invoke it. Invocation of an SRB routine has the overhead of scheduling and dispatching an SRB.

Processing by an exit routine: A resource manager can have a NOTIFICATION exit routine for each exit manager or a single routine for all exit managers. At invocation, the exit routine receives a parameter list, which names the exit manager. If a resource manager uses a single exit routine, the routine can identify the processing needed based on the *exit_manager_name* parameter.

If the exit manager was unavailable when the resource manager called the Set_Exit_Information service, the exit routine can now set its exits with the exit manager.

If the resource manager previously set its exit routines with the exit manager, the exit routine was invoked because the exit manager became unavailable. In this case, the routine can do processing needed because the exit manager is no longer available.

Returning from an exit routine: An exit routine returns to its exit manager as follows:

- An SRB routine must return to the address that was in register 14 on entry to the routine.
- A PC routine must return with a Program Return (PR) instruction.

Action on exit routine failure: If the exit routine fails, it returns a nonzero return code or abnormally ends with an abend code. In response, the system unsets the resource manager's exit routines and unregisters the resource manager.

Environment

Before the exit routine receives control, registration services establishes a function recovery routine (FRR) for error recovery.

An SRB exit routine receives control in the following environment:

Minimum authorization:	Key of the resource manager when it registered, supervisor state
Dispatchable unit mode:	SRB
Cross memory mode:	PASN = HASN = SASN, home address space of the resource manager when it registered
AMODE:	31-bit
ASC mode:	Primary

Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

A PC exit routine receives control in the following environment:

Minimum authorization:	Determined by the PC instruction characteristics, supervisor state
Dispatchable unit mode:	SRB or task
Cross memory mode:	Determined by the PC instruction characteristics, home address space unpredictable
AMODE:	Determined by the PC instruction characteristics
ASC mode:	Determined by the PC instruction characteristics
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming requirements

The high level language (HLL) definitions for the exit routine parameter list are:

HLL Definition	Description
CRGASM	390 Assembler declarations
CRGC	C/390 declarations

Entry to an exit routine

The exit routine receives information in the registers and a parameter list.

Registers at entry

When an SRB exit routine receives control, the GPRs contain:

Register	Contents
0	Not applicable
1	Address of the parameter list for the exit routine
2-12	Not applicable
13	Address of a 72-byte save area
14	Return address
15	Address of the exit routine's entry point

When an SRB exit routine receives control, the ARs contain:

Register	Contents
0-15	Not applicable

When a PC exit routine receives control, the GPRs contain:

Register	Contents
0	Not applicable

1 Address of the parameter list for the exit routine

2-15 Not applicable

When a PC exit routine receives control, the ARs contain:

Register
Contents

0-15 Not applicable

Parameter list syntax

The parameter list consists of pointers to fields containing the values. If a parameter is not meaningful for the exit routine being invoked, the field contains binary zeros. All parameters, except *return_code*, are input to the exit routine. Access to the parameters is controlled by storage protect key:

- **Input parameters**: For the parameters received by the exit routine, the resource manager and exit routine have READ access, but might not have WRITE access.
- **Output parameters**: For the parameters returned by the exit routine, the resource manager and exit routine have READ and WRITE access.

```
(return_code
,version
,exit_number
,resource_manager_token
,reg_exit_manager_name
,resource_manager_global_data
,exit_manager_name
,value1
,value2
,value3
,value4
,value5)
```

Parameters

`return_code`
Points to a field that, upon return from the exit routine, is to contain a hexadecimal return code. Define the field as a 4-byte integer.

`version`
Points to a field that contains the version of the registration services interface. The current version is 1. Define the field as a 4-byte integer.

`exit_number`
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
1

Decimal
1

Equate symbol
CRG_NOTIFICATION_EXIT

resource_manager_token
Points to a field that contains the resource manager token. Define the field as a 16-byte character string. Your resource manager received the token from the Register_Resource_Manager service.

reg_exit_manager_name
Points to a field that contains the name of the exit manager that is driving the exit. Define the field as a 16-byte character string. The exit manager for this exit routine is registration services; its exit manager name is:

```
CRG.REGSERV.IBM
```

resource_manager_global_data
Points to a field that contains the resource manager's global data. Define the field as a 16-byte character string. Your resource manager provided this data in the call to the Register_Resource_Manager service.

For the exit routine, this data should be an anchor or anchors for data structures in the resource manager.

exit_manager_name
Points to a field that contains the name of the exit manager for which this exit is being driven. Define the field as a 16-byte character string. The name of the RRS exit manager is:

```
ATR.EXITMGR.IBM
```

value1
Points to a field that contains the reason the exit routine is being invoked. Define the field as a 4-byte hexadecimal constant.

Value in: **Hexadecimal** **(Decimal)** **Equate Symbol**	**Event**
1 (1) CRG_EM_AVAILABLE	**Exit manager registered**: The exit manager that was previously not available during a call to the Set_Exit_Information service is now available.
2 (2) CRG_EM_UNAVAILABLE	**Exit manager unregistered**: The exit manager that was previously available during a call to the Set_Exit_Information service is now unavailable.
3 (3) CRG_RM_EXITS_UNSET	**Resource manager exits unset**: The exit manager has unset the exits for the resource manager.

value2
Points to a field that, when *value1* is CRG_RM_EXITS_UNSET, contains the reason the exits were unset. Define the field as a 4-byte hexadecimal constant.

Value in: Hexadecimal (Decimal) Equate Symbol	Meaning
0 (0) CRG_UNSET_EFE_REQUESTED	The resource manager's EXIT_FAILED exit routine has requested that the exit manager unset the resource manager's exit routines.
1 (1) CRG_UNSET_EFE_FAILED	The resource manager's EXIT_FAILED exit routine has failed, and the exit manager has unset the resource manager's exit routines.
2 (2) CRG_UNSET_EFE_BAD_RETCODE	The resource manager's EXIT_FAILED exit routine has returned a bad return code, and the exit manager has unset the resource manager's exit routines.
8000–8008 (32768–32776) EXIT_MANAGER_SPECIFIC	The exit manager has unset the resource manager's exit routines for a reason that is specific to the resource manager. See the information about the resource manager for information about the specific code.
8009 (32777) ATR_EM_UNAVAILABLE	The resource manager has been unset because RRS was terminated through the SETRRS SHUTDOWN command.
800A–FFFF (32778–65535) EXIT_MANAGER_SPECIFIC	The exit manager has unset the resource manager's exit routines for a reason that is specific to the resource manager. See the information about the resource manager for information about the specific code.

value3
value4
value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Exit from an exit routine

The exit routine provides information to the system in the return code in the parameter list.

Registers at Exit: When a SRB exit routine returns control, the GPRs must contain:

Register
Contents

0-1 Not applicable

2-13 Restored to contents upon entry

14-15 Not applicable

When an SRB exit routine returns control, the ARs must contain:

Register
Contents

0-1 Not applicable

2-13 Restored to contents upon entry

14-15 Not applicable

When a PC exit routine returns control, the GPRs contain:

Register
Contents

0-15 Not applicable

When a PC exit routine returns control, the ARs contain:

Register
Contents

0-15 Not applicable

Return codes

When the NOTIFICATION exit routine returns control to the system, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	**Meaning and Action**
0 (0) CRG_OK	**Meaning**: The NOTIFICATION exit routine completed processing. **Action**: None.
hhh (dddd) ccccccccccc	**Meaning**: The NOTIFICATION exit routine failed. **Action**: Check the exit routine for a probable coding error. Correct the exit routine and rerun it.

Chapter 3. Using context services

Context services allow a resource manager to indicate interest in a work context.

A **context** represents the resources for a work request; a context consists of the application program requesting the work and the protected resources involved in the work. A *context* represents a business unit of work: one or more units of recovery with the associated application programs, resource managers, and protected resources. The context should be the anchor for the resource manager's control structures related to the work request.

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 is using RRS might need to express interest in a work context, not just a particular UR, because a context can persist over multiple URs (as shown earlier in Figure 1 on page 4).

Contexts

A context can be either:

- A native context, which is 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.
- A privately-managed context, which a resource manager creates. The resource manager *owns* a 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 work manager, which is a resource manager, like IMS, that accepts and manages work, such as transactions, from outside the system.

Table 1 summarizes the differences between the two types of contexts.

Table 1. Context Type Differences

Difference	Native Context	Privately-Managed Context
Creation of context	Implicitly by the operating system	Explicitly by a resource manager
Association of context	Always with an application's task	Temporarily may not be associated with any task
Association change	Cannot change association from one application's task to another	Can change association from one application's task to another at any time. The change can even be to a work unit in a different home address space

Table 1. Context Type Differences (continued)

Difference	Native Context	Privately-Managed Context
Context end	• When the application's task ends • When a resource manager running under the application's task explicitly ends the context (though a new native context automatically begins) • When the home address space of the application's task abnormally ends	• When a resource manager explicitly ends a privately-managed context that it owned. If the context is associated with an application's task, the resource manager must be running under that task • If the owning resource manager ends or unregisters, a context disassociated from an application's task ends immediately • If the owning resource manager ends or unregisters, a context associated with an application's task continues until the task ends • If the application's task associated with a privately-managed context ends, the system invokes the PVT_CONTEXT_OWNER exit routine, if provided. The routine indicates if the privately-managed context is to be ended or disassociated from the task

Expressing Interest in a Context: A resource manager expresses interest in a context to cause the system to invoke the resource manager's exit routine when:

- The context ends
- The context switches from one application's task to another

When expressing interest in a context, a resource manager can provide context interest data. This data can contain an anchor for the resource manager's data structures for the context.

When a resource manager expresses interest in a context, the system provides a context token that represents the context. The context token is unique within an MVS system or sysplex but is not guaranteed to be unique across a network of MVS systems.

Privately-Managed Contexts: When a resource manager that is processing transactions creates a new work request, the resource manager should create a new privately-managed context for the request. The resource manager can associate the context with the application's task that will run for the work request.

If needed, the resource manager can disassociate the privately-managed context from a task and later reassociate it with the same task or another task. By changing the associations, the resource manager can have one task that runs for many work requests, many tasks that run in series for a single work request, or both. Note that a context cannot be associated at the same time with multiple tasks.

When a task changes from processing for one work request to processing for another work request, the resource manager should switch the privately-managed contexts associated with the task.

Current Context: Every task in the system has an associated context; thus, there is always a context for a given task. When a task is created, context services provides the original (native) context for the task. A call to the Begin_Context service creates a privately-managed context, and a call to the Switch_Context service changes the current context to the privately-managed context. The native context still exists, but is not current. If a later call to the Switch_Context service disassociates the privately-managed context, the native context again becomes the current context.

If a privately-managed context associated with a task ends, the native context becomes the current context. If a task ends while there is a privately-managed context associated with it, the privately-managed context ends, followed immediately by the end of the task's native context.

Callable services for contexts

Resource managers that use context services might use any of the following services, particularly Express_Context_Interest:

Callable Service	Description
Delete_Context_Interest	Delete interest in a context
Express_Context_Interest	Express® interest in a context
Retrieve_Context_Interest_Data	Retrieve context interest data
Retrieve_Current_Context_Token	Retrieve the context token for the currently active context
Set_Context_Interest_Data	Set context interest data

Resource managers that are also work managers use the following context services:

Callable Service	Description
Begin_Context	Begin a privately-managed context
End_Context	End a privately-managed context
Switch_Context	Switch a context

Unauthorized resource managers

Resource managers which run in PKM 8–15 and problem state are considered to be unauthorized. Unauthorized resource managers can use context services to obtain and manage contexts; however, context services imposes limitations on them which it does not impose on PKM 0–7 or supervisor state resource managers. These limitations include:

- At most, 256 unauthorized resource managers can register from a single address space, unless an operator explicitly allows additional resource managers.
- At most, 256 contexts can be obtained by each unauthorized resource manager, unless an operator explicitly allows the resource manager to get more.
- Unauthorized resource manager names must end with `.UA`
- The CRG_UNREG_EOM option cannot be used as an unregister option on the Register_Resource_Manager service.

- No exits are allowed.
- Interest cannot be expressed in any context. As a result, the unauthorized resource manager cannot get notification of context related events.
- Contexts can only be switched between tasks in the unauthorized resource manager's home address space.
- Only contexts obtained by unauthorized resource managers registered in the same home address space can be affected.
- Only the following services can be used by unauthorized resource managers:
  - Begin_Context
  - End_Context
  - Retrieve_Context_Data
  - Retrieve_Current_Context_Token
  - Switch_Context

Context services exit routines

Your resource manager can provide exit routines to be invoked when events occur for its interest in a context. Table 2 lists the context services exit routines.

Table 2. Context Services Exit Routines

Exit Routine	Exit Number in: Hexadecimal (Decimal) Equate Symbol	Event
CONTEXT_SWITCH	2 (2) CTX_SWITCH_EXIT	A call to the Switch_Context service or the termination of a disassociated context
END_CONTEXT	4 (4) CTX_END_CONTEXT_EXIT	A context is ending for any reason, including a call to the End_Context service
EOM_CONTEXT	5 (5) CTX_EOM_CONTEXT_EXIT	A context is ending because the address space associated with it is ending. This exit routine is invoked before the END_CONTEXT exit routine
EXIT_FAILED	1 (1) CTX_EXIT_FAILED_EXIT	A context services exit routine failed
PVT_CONTEXT_OWNER	3 (3) CTX_PRIVATE_CONTEXT_ OWNER	A work unit associated with a private context is ending

These exit routines are optional. A resource manager could, for example, use context services without any exit routine and call the Express_Context_Interest service just to keep data in the *context_interest_data* area the service provides. If,

however, your resource manager does choose to provide any context services exit routines, it must also provide an EXIT_FAILED exit routine.

When a context ends, the exit routines that context services invokes, and the order in which context services invokes the routines, depend on:

- The type of context (privately-managed context or native context)
- The reason the context is ending

Figure 13 shows the conditions and the order in which exit routines are invoked.

For a privately-managed context:

- When the task associated with the context ends:
  1. PVT_CONTEXT_OWNER
  2. CONTEXT_SWITCH (if the owner tells RRS to disassociate the context from the task)
  3. END_CONTEXT
- When the address space associated with the context ends:
  1. PVT_CONTEXT_OWNER
  2. CONTEXT_SWITCH (if the owner tells RRS to disassociate the context from the task — see note 1)
  3. EOM_CONTEXT
  4. END_CONTEXT (see note 2)
- When the context is not associated with a task and its owner ends:
  1. EOM_CONTEXT
  2. CONTEXT_SWITCH
  3. END_CONTEXT (see note 2)
- When the End_Context service was called to end a context associated with a task:
  1. END_CONTEXT
- When the End_Context service was called to end a context not associated with a task:
  1. CONTEXT_SWITCH
  2. END_CONTEXT

For a native context:

- When the task associated with the context ends:
  1. END_CONTEXT
- When the address space associated with the context ends:
  1. EOM_CONTEXT
  2. END_CONTEXT (see note 2)
- When the End_Context service was called to end the context:
  1. END_CONTEXT

Note:

1. If the PVT_CONTEXT_OWNER exit routine tells context services to disassociate the context from the task, context services drives CONTEXT_SWITCH exit routines. If any CONTEXT_SWITCH exit routine disallows the switch, context services continues processing to end the context, but it changes the reason for the context end to CTX_FORCED_END_OF_CONTEXT.
2. During processing the end of an address space, context services invokes the CONTEXT_SWITCH, EOM_CONTEXT, and PVT_CONTEXT_OWNER exit routines before the RTM resource managers (RESMGRs).

Figure 13. Order of Invocation for Context Services Exit Routines

Programming considerations

The following topics discuss installing, invoking, processing, and returning for an exit routine and the action taken on an exit routine failure.

Installing an exit routine

To ensure that context services can drive its exit routines, the resource manager must:

- Register itself through a call to the Register_Resource_Manager service.
- Set the context services exit routines through one or more calls to the Set_Exit_Information service. If the resource manager specifies any exit routines, it must also specify the EXIT_FAILED routine.

Note that exits might be driven even before control returns from Set_Exit_Information.

If your resource manager needs private context delegation to RRS, you must specify the RRS resource manager name so Context Services knows to communicate with RRS. You specify this through the variable_data_1 parameter on the Set_Exit_Information service. See “Private context delegation” on page 53 for a description of private context delegation to RRS.

Set_Exit_Information returns codes related to its processing. See “Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF)” on page 149. Set_Exit_Information might also return codes from the exit manager. The following table lists the return codes you might get from Context Services when you call the Set_Exit_Information service to set the Context Services exit routines.

Return Code in: Hexadecimal (Decimal) Equate Symbol	**Meaning and action**
8000 (32768) CTX_DELEGATION_INV	**Meaning**: The specified resource manager does not support private context delegation. The system rejects the service request. **Action**: Specify the name of a resource manager that supports private context delegation.

Invoking an exit routine

Before context services can invoke an exit routine, however, the resource manager must also express interest in a context through a call to the Express_Context_Interest service.

The system invokes a context services exit routine when its context-related event occurs and the resource manager has expressed interest in the context related to the event. If your resource manager has more than one interest in a context, the system will invoke the exit routine for each interest.

The *exit_type* parameter in the call to the Set_Exit_Information service specifies how context services is to invoke the exit routine:

- **SRB routine**: The system schedules a service request block (SRB) at local priority in the resource manager's address space to give control to the exit routine.

The exit routine may run synchronously or asynchronously. In either case, it will be nonpreemptable.

A resource manager in a swappable address space must use SRB exit routines.

- **PC routine**: The system issues a stacking Program Call (PC) instruction to give control to the exit routine. The stacking PC must use a system LX so that the routine is available from all address spaces.

  Note: Consider carefully before deciding to use a system LX. Using a system LX improperly can prevent ASIDs from being reused, which can in turn cause unscheduled IPLs. To avoid unnecessary loss of ASIDs, IBM recommends that a resource manager use a system LX only when the resource manager is a long-running address space. See "Reusing ASIDs" in *z/OS MVS Programming: Extended Addressability Guide* for more detail.

  The exit routine will run synchronously; therefore, the resource manager must not suspend processing of the work unit. The system cannot invoke any other exit routines until the PC routine completes.

  The resource manager must be in a nonswappable address space to use PC exit routines. A PC exit routine must remain available to the system until the resource manager ends processing, unregisters, or issues a call to the set exit routine service to change the exit routine.

  A PC exit routine and any routine that it invokes cannot issue an SVC instruction.

The advantage of the PC routine over an SRB routine is a shorter path length to invoke it. Invocation of an SRB routine has the overhead of scheduling and dispatching an SRB.

Processing by an exit routine

A resource manager can have an exit routine for each context services exit or a single routine for all context services exits. At invocation, all context services exit routines receive a parameter list in the same format but with exit-specific meanings for some parameters. If a resource manager uses a single exit routine, the routine can identify the processing needed based on the exit number parameter.

Returning from an exit routine

An exit routine returns to context services as follows:

- An SRB routine must return to the address that was in register 14 on entry to the routine.
- A PC routine must return with a Program Return (PR) instruction.

Action if an exit routine fails

If an exit routine percolates an abend or returns an unexpected return code, the system gives control to the EXIT_FAILED exit routine.

Action if exit routines are unset

If a resource manager's exit routines are unset for any reason:

- Context services will not quiesce any exit routines that are active when the exit routines are unset. The exit routines continue to run.

A quiesced exit routine completes normally or abnormally, then returns to the caller.

If an exit routine that continues to run requests a context service, it will get an error return code from the service.

Environment

Before the exit routine receives control, context services establishes a functional recovery routine (FRR) for error recovery.

An SRB exit routine receives control in the following environment:

Minimum authorization:	Key of the resource manager when it registered, supervisor state
Dispatchable unit mode:	SRB
Cross memory mode:	PASN = HASN = SASN, home address space of the resource manager when it registered
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

A PC exit routine receives control in the following environment:

Minimum authorization:	Determined by the PC instruction characteristics, supervisor state
Dispatchable unit mode:	SRB or Task
Cross memory mode:	Determined by the PC instruction characteristics, home address space unpredictable
AMODE:	Determined by the PC instruction characteristics
ASC mode:	Determined by the PC instruction characteristics
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming requirements

The high level language (HLL) definitions for the exit routine parameter list are:

HLL Definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations

Entry to an exit routine

The exit routine receives information in the registers and a parameter list.

Registers at entry

When an SRB exit routine receives control, the GPRs contain:

Register	Contents
0	Not applicable
1	Address of the parameter list for the exit routine
2-12	Not applicable
13	Address of a 72-byte save area
14	Return address
15	Address of the exit routine's entry point

When an SRB exit routine receives control, the ARs contain:

Register	Contents
0-15	Not applicable

When a PC exit routine receives control, the GPRs contain:

Register	Contents
0	Not applicable
1	Address of the parameter list for the exit routine
2-15	Not applicable

When a PC exit routine receives control, the ARs contain:

Register	Contents
0-15	Not applicable

Parameter list

The parameter list is the same for all context services exit routines.

The parameter list consists of pointers to fields containing the values. If a parameter is not meaningful for the exit routine being invoked, the field contains binary zeros. All parameters, except *return_code*, are input to the exit routine. Access to the parameters is controlled by storage protect key:

- **Input parameters**: For the parameters received by the exit routine, the resource manager and exit routine have READ access, but might not have WRITE access.
- **Output parameters**: For the parameters returned by the exit routine, the resource manager and exit routine have READ and WRITE access.

Syntax:

```
(return_code
,version
,exit_number
,resource_manager_token
,exit_manager_name
,resource_manager_global_data
,context_token
,context_interest_token
,context_interest_data
,value1
,value2
,value3
,value4
,value5)
```

Parameters:

`return_code`
Points to a field that, upon return from the exit routine, is to contain a hexadecimal return code. Define the field as a 4-byte integer.

The return codes have unique meanings for each exit routine. See the individual exit routine descriptions for the return codes.

version
Points to a field that contains the version of the context services interface. The current version is 1. Define the field as a 4-byte integer.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer.

Each of the following exit routine descriptions includes the number of the exit. If a single exit routine is used for multiple exits, the routine can use this number to determine the event that caused the exit to be driven.

resource_manager_token
Points to a field that contains the resource manager token. Define the field as a 16-byte character string. Your resource manager received the token from the Register_Resource_Manager service.

exit_manager_name
Points to a field that contains the name of the exit manager. Define the field as a 16-byte character string. The exit manager for this exit routine is context services: its exit manager name is:

```
CTX.EXITMGR.IBM
```

The equate symbol for the name is CTX_EXIT_MGR_NAME.

resource_manager_global_data
Points to a field that contains the resource manager global data. Define the field as a 16-byte character string. Your resource manager provided this data in the call to the Register_Resource_Manager service.

For the exit routine, this data should be an anchor or anchors for data structures in the resource manager.

context_token
Points to a field that contains the context token for the context for which the system is invoking the exit routine. Define the field as a 16-byte character string.

Your resource manager receives the token from the Express_Context_Interest service or, for a privately-managed context, from the Begin_Context service.

context_interest _token
Points to a field that contains the context interest token for the interest for which the system is invoking the exit routine. Define the field as a 16-byte character string. Your resource manager received the token from the Express_Context_Interest service.

context_interest_data
Points to a field that contains the context interest data. Define the field as a 16-byte character string. Your resource manager provided this data in a call to the Express_Context_Interest service or the Set_Context_Interest_Data service.

value1
value2
value3
value4

value5
Point to fields that contain values unique for the exit routines. Define each field as a 4-byte integer. If a value is not used for an exit routine, its field contains binary zeros.

See the individual exit routine descriptions for the values.

Exit from an exit routine

The exit routine provides information to the system in the return code in the parameter list.

Registers at Exit: When an SRB exit routine returns control, the GPRs must contain:

Register	Contents
0-1	Not applicable
2-13	Restored to contents upon entry
14-15	Not applicable

When an SRB exit routine returns control, the ARs must contain:

Register	Contents
0-1	Not applicable
2-13	Restored to contents upon entry
14-15	Not applicable

When a PC exit routine returns control, the GPRs contain:

Register	Contents
0-15	Not applicable

When a PC exit routine returns control, the ARs contain:

Register	Contents
0-15	Not applicable

CONTEXT_SWITCH exit routine

The CONTEXT_SWITCH exit routine receives control for:

- **A context switch**: A resource manager called the Switch_Context service to switch the context associated with the application's task to another context. The exit routine decides if the switch should be allowed or disallowed. If allowed, the routine does processing needed before a context switch.
- **Context end**: Either a resource manager called the End_Context service to end a context not associated with a task, or a context not associated with a task is ending for another reason. The *value1* parameter reports the reason the exit was called. The exit routine does processing needed when the context ends.

Restrictions

Do not call any of the following services to process the context passed to the exit routine in the *context_token* parameter:

End_Context
Context_Switch
Express_Context_Interest

Unique parameters

For information about common parameters, see "Parameter list" on page 39.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
2

Decimal
2

Equate symbol
CTX_SWITCH_EXIT

value1
Points to a field that describes the event causing the context switch. Define the field as a 4-byte integer. The hexadecimal values for the events are:

Value in: Hexadecimal (Decimal) Equate Symbol	Event
1 (1) CTX_SWITCH_TO	**Switching to a context:** The call to the Switch_Context service requests a switch to the context identified in the *context_token* parameter.
2 (2) CTX_SWITCH_FROM	**Switching from a context:** The call to the Switch_Context service requests a switch from the context identified in the *context_token* parameter.
3 (3) CTX_SWITCH_DISASSOC_END_NORM	**Normal end of a work unit:** The disassociated privately-managed context identified in the *context_token* parameter is being ended by a call to the End_Context service. The *completion_type* from the End_Context service is CTX_NORMAL_TERMINATION.
4 (4) CTX_SWITCH_DISASSOC_END_ ABNORM	**Abnormal end of a work unit:** The disassociated privately-managed context identified in the *context_token* parameter is being ended by a call to the End_Context service. The *completion_type* from the End_Context service is CTX_ABNORMAL_TERMINATION.
5 (5) CTX_SWITCH_END_FORCED	**Forced end of a work unit:** The context identified in the *context_token* parameter is being ended by a call to the End_Context service after a call to the Switch_Context service marked the switch as disallowed. The *completion_type* from the End_Context service is CTX_FORCED_END_OF_CONTEXT.

Value in: Hexadecimal (Decimal) Equate Symbol	Event
6 (6) CTX_SWITCH_MEMTERM	**Memory termination of work unit's address space:** The address space for the work unit associated with the context identified in the *context_token* parameter is ending.
7 (7) CTX_SWITCH_MEMTERM_PRIV_OWNER	**Context owner's address space terminated:** The address space for the owner of the private context identified in the *context_token* parameter is ending. This condition occurs only when the context is not associated with any dispatchable unit.
8 (8) CTX_SWITCH_UNREG_PRIV_OWNER	**Context owner unregistered:** The resource manager that owns the private context identified in the *context_token* parameter has called the Unregister_Resource_Manager service. This event occurs only when the context is not associated with any dispatchable unit.

value2
value3
value4
value5

Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the CONTEXT_SWITCH exit routine returns control to the system, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
0 (0) CTX_OK	**Meaning**: The CONTEXT_SWITCH exit routine allows the switch requested in the call to the Switch_Context service. If the CONTEXT_SWITCH exit routines for all interests in the context return CTX_OK, the Switch_Context service processes the request. **Note:** This is the only code that can be returned when the *value1* parameter contains: CTX_SWITCH_DISASSOC_END_FORCED CTX_SWITCH_MEMTERM CTX_SWITCH_MEMTERM_PRIV_OWNER **Action**: None.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
800 (2048) CTX_DISALLOW_SWITCH	**Meaning**: The CONTEXT_SWITCH exit routine disallows the switch requested in the call to the Switch_Context service. The system rejects the Switch_Context request, does not perform the switch, and returns this return code to the calling resource manager. This return code is provided for resource managers that cannot tolerate a context switch to or from the target environment. Note that by disallowing the switch behavior, the issuing resource manager may present problems for the owing work manager or other resource managers interested in the context. For this reason, IBM discourages usage of this return code. **Note:** This code cannot be returned when the *value1* parameter contains: CTX_SWITCH_DISASSOC_END_FORCED CTX_SWITCH_MEMTERM CTX_SWITCH_MEMTERM_PRIV_OWNER **Action**: Check the resource manager for a probable coding or environmental error. Correct the resource manager and rerun it.
801 (2049) CTX_DISALLOW_SWITCH_WU	**Meaning**: The CONTEXT_SWITCH exit routine disallows the switch requested in the call to the Switch_Context service because the calling resource manager is running under the wrong work unit. The system rejects the Switch_Context request, does not perform the switch, and returns this return code to the calling resource manager. This return code is provided for resource managers that cannot tolerate a context switch to or from the target environment. Note that by disallowing the switch behavior, the issuing resource manager may present problems for the owing work manager or other resource managers interested in the context. For this reason, IBM discourages usage of this return code. **Note:** This code cannot be returned when the *value1* parameter contains: CTX_SWITCH_DISASSOC_END_FORCED CTX_SWITCH_MEMTERM CTX_SWITCH_MEMTERM_PRIV_OWNER **Action**: Check the resource manager for a probable coding or environmental error. Correct the resource manager and rerun it.

END_CONTEXT exit routine

The END_CONTEXT exit routine receives control when a context is ending for any reason, including a call to the End_Context service. The exit routine should clean up private resource manager structures for this context.

Restrictions

Do not call any of the following services to process the context passed to the exit routine in the *context_token* parameter:

End_Context

Context_Switch

Express_Context_Interest

Unique parameters

For information about common parameters, see "Parameter list" on page 39.

exit_number

Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
4

Decimal
4

Equate symbol
CTX_END_CONTEXT_EXIT

value1

Points to a field that contains the completion type for the context. The completion type was specified in a call to the End_Context service. Define the field as a 4-byte integer. The hexadecimal values for the completion types are:

Constant in: Hexadecimal (Decimal) Equate Symbol	Completion type
0 (0) CTX_NORMAL_TERMINATION	The context is ending normally.
1 (1) CTX_ABNORMAL_TERMINATION	The context is ending abnormally.
2 (2) CTX_ABNORMAL_EOM_TERMINATION	The context must end because the address space for the application is ending abnormally.
3 (3) CTX_FORCED_END_OF_CONTEXT	A work manager is forcing the context to end.

Constant in: Hexadecimal (Decimal) Equate Symbol	Completion type
4 (4) CTX_PRIV_OWNER_TERMINATION	The privately-managed context must end because the resource manager that owns the context is ending.

value2
value3
value4
value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the END_CONTEXT exit routine returns control to the system, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
0 (0) CTX_OK	**Meaning**: The END_CONTEXT exit routine completed processing. **Action**: None.

EOM_CONTEXT exit routine

The EOM_CONTEXT exit routine receives control when a context is ending because the address space associated with the context is ending. The purpose of this exit is to inform the resource managers that the address space is ending. If you provide this exit, it is invoked before the END_CONTEXT exit routine

Unique parameters

For information about common parameters, see “Parameter list” on page 39.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
5

Decimal
5

Equate symbol
CTX_EOM_CONTEXT_EXIT

value1
value2
value3
value4

value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the EOM_CONTEXT exit routine returns control to the system, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: **Hexadecimal** **(Decimal)** **Equate Symbol**	**Meaning and action**
0 (0) CTX_OK	**Meaning**: The EOM_CONTEXT exit routine completed processing. **Action**: None.

EXIT_FAILED exit routine

The EXIT_FAILED exit routine receives control when a context services exit routine fails. Context services gives this routine the exit number of the failed routine and the reason why the routine failed. The return code from the EXIT_FAILED routine tells context services what action to take, usually unsetting the resource manager's context services exit routines.

If the EXIT_FAILED exit routine percolates an abend or returns an undefined return code, context services unsets the resource manager's context services exit routines.

Restrictions

Do not call any of the following services to process the context passed to the exit routine in the *context_token* parameter:

End_Context

Context_Switch

Express_Context_Interest

Unique parameters

For information about common parameters, see “Parameter list” on page 39.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
1

Decimal
1

Equate symbol
CTX_EXIT_FAILED_EXIT

value1
Points to a field that contains the exit number of the failed exit routine. See the individual exit routine descriptions for the numbers. Define the field as a 4-byte integer.

value2
Points to a field that contains the reason why the exit routine failed. Define the field as a 4-byte integer. The hexadecimal values for the reasons are:

Value in: Hexadecimal (Decimal) Equate Symbol	Reason
1 (1) CTX_EXIT_INCORRECT_RC	**Incorrect return code**: An exit routine returned an incorrect return code to context services. The incorrect code is in the *value3* parameter.
2 (2) CTX_EXIT_ABENDED	**Exit routine abended**: The abend percolated to the context services FRR. The ABEND code is in the *value3* parameter.
3 (3) CTX_EXIT_ABENDED_RSN	**Exit routine abended**: The abend percolated to the context services FRR. The ABEND code is in the *value3* parameter, and its reason code is in *value4*.
4 (4) CTX_MEMTERM	**Address space ended**: While the exit routine was running, the dispatchable unit's address space or the privately-managed context owner's address space terminated.

value3
Points to a field that contains the following code, depending on *value2*. Define the field as a 4-byte integer.

Value in *value2*	Contents of *value3* Field
CTX_EXIT_INCORRECT_RC	The incorrect return code
CTX_EXIT_ABENDED	The abend code
CTX_EXIT_ABENDED_RSN	The abend code

value4
Points to a field that contains, if *value2* is CTX_EXIT_ABENDED_RSN, the ABEND reason code. Otherwise the field contains binary zeros. Define the field as a 4-byte integer.

value5
Points to a field that contains binary zeros. Define the field as a 4-byte integer.

Return codes

When the EXIT_FAILED exit routine returns control to the system, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and Action
810 (2064) CTX_EXIT_UNSET_RM	**Meaning**: The EXIT_FAILED exit routine completed processing. Context services is to unset the context services exit routines for the resource manager. **Action**: The resource manager should load a new copy of the failed exit routine, then call the Set_Exit_Information service to set all of its exit routines with context services again. If the problem recurs, you should check the failed exit routine for a probable coding error. Correct the routine and rerun the resource manager.
hhh (dddd) An exit-specific equate symbol	**Meaning**: The EXIT_FAILED exit routine completed processing for the exit routine that failed. The return code is valid for the failed exit; see the return codes for the failed exit. **Action**: Perform the action for the return code.

PVT_CONTEXT_OWNER exit routine

The PVT_CONTEXT_OWNER exit routine receives control when a work unit associated with a private context is ending. The exit routine decides if the privately-managed context should be allowed to end or not.

The exit is driven only for the resource manager that issued Begin_Context to create the privately-managed context, and only if that resource manager expressed interest in the context. If the resource manager expressed interest multiple times, the resource manager's PVT_CONTEXT_OWNER exit routine is driven once for each expression of interest. If any exit invocation returns CTX_DIS_PVT_CONTEXT, then RRS disassociates the context from the dispatchable unit and does not end the context.

Unique parameters

For information about common parameters, see “Parameter list” on page 39.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
3

Decimal
3

Equate symbol
CTX_PRIVATE_CONTEXT_OWNER

value1
Points to a field that indicates how the privately-managed context is ending. Define the field as a 4-byte integer. The hexadecimal values for the endings are:

Value in: Hexadecimal (Decimal) Equate Symbol	Type of ending
0 (0) CTX_NORMAL_TERM	Normal ending
1 (1) CTX_ABNORMAL_TERM	Abnormal ending

value2
value3
value4
value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the PVT_CONTEXT_OWNER exit routine returns control to the system, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
0 (0) CTX_END_PVT_CONTEXT	**Meaning**: The PVT_CONTEXT_OWNER exit routine completed processing. The context should be allowed to end. **Action**: None.
1 (1) CTX_DIS_PVT_CONTEXT	**Meaning**: The PVT_CONTEXT_OWNER exit routine completed processing. The privately-managed context from the work unit that is ending should be disassociated from the work unit that is ending but the context should not be ended. **Action**: If the PVT_CONTEXT_OWNER exit routine tells context services to disassociate the context from the task, context services drives CONTEXT_SWITCH exit routines. If any CONTEXT_SWITCH exit routine disallows the switch, context services continues processing to end the context, but it changes the reason for the context end to CTX_FORCED_END_OF_CONTEXT.

Chapter 4. Using resource recovery services

Resource recovery services (RRS) provides a set of services that implement the two-phase commit protocol on the z/OS platform. Your resource manager follows the two-phase commit protocol to protect resources by invoking these services and providing exit routines. While "Planning a resource manager" on page 18 presents an overview of how to plan a resource manager, this section describes how to use RRS services and how to code the exit routines. You can find details about each resource recovery service in Chapter 7, "Callable resource recovery services," on page 227 and details about each exit routine in "Resource recovery exit routines" on page 91.

Resource manager states

While processing protected resources, the resource manager passes through different states. The resource manager state determines the processing the resource manager can perform. Table 3 and the following topics describe the resource manager processing in the order in which it should occur. Table 3 also shows the state the resource manager is in before the processing and the state it is in after the processing.

Table 3. Resource Manager Processing and States

Resource manager processing	**Resource manager state**	**Description**
Registering	Before: Reset After: Registered	The resource manager registers itself with the operating system's registration services.
Setting exit routines	Before: Registered After: Set	The resource manager sets its exit routines by identifying them and their entry points. Note that the resource manager's EXIT_FAILED exit routine or the exit manager can unset the exit routines, leaving the resource manager in a registered state, but with its exit routines unset.
Restarting	Before: Set During: Restart After: Run	The resource manager restarts itself by retrieving and processing any URs that were incomplete from the last time the resource manager was running. Note that the process is the same for starting and restarting.
Expressing interest in a context	Run	When an application program requests access to a resource, the resource manager optionally expresses interest in the context associated with the application and the work request.
Expressing interest in a UR	Run	The resource manager expresses interest in the UR being processed by the application program.
Protecting the resource	Run	The resource manager changes or does not change the resource.

Resource manager roles

For each UR, there is a syncpoint manager and participating resource managers. The syncpoint manager determines the outcome, either commit or back out, for the UR, and the participating resource managers accept the outcome and ensure that the resources they manage are changed or not changed. Normally, RRS takes the role of syncpoint manager, and all resource managers act as participants.

Sometimes, however, a resource manager can tell RRS that it needs to determine the outcome of the syncpoint. For example, a communication resource manager might need this control when the outcome of the syncpoint is actually being determined by a syncpoint manager on another system and the communication resource manager is being used to communicate the outcome to RRS. In this case, the communication resource manager becomes the syncpoint manager.

When the resource manager becomes the syncpoint manager, it is either:

- A distributed syncpoint manager (DSRM)
- A server distributed syncpoint manager (SDSRM)

The difference between the DSRM role and the SDSRM role is how they communicate the outcome to RRS:

- The DSRM role is provided for communication resource managers, such as APPC/MVS, that follow the peer-to-peer model, described earlier in "Peer-to-peer model" on page 9.
- The SDSRM role is provided for communication resource managers that follow the client-server model, described earlier in "Client-server model" on page 14.

To take either the DSRM role or the SDSRM role, the resource manager calls the Set_Syncpoint_Controls service. Note that, for any given syncpoint operation, only one resource manager can take the DSRM or SDSRM role.

Resource manager failures

When a resource manager fails, the effect of the failure on each UR it has expressed interest in is determined by several factors: the state of the UR, the failure action specified by the resource manager when it expressed interest, and, sometimes, whether the interest is protected or unprotected. Table 4 lists the possible UR states and the effect a resource manager failure would have on the UR.

If multiple resource managers fail, or if a failing resource manager has multiple interests in a UR, Table 4 might show different actions for the same UR. In that case, the action for a protected interest overrides the actions for an unprotected interest.

Table 4. UR States and Failure Actions

UR state at time of failure	Action
In-reset	If the failure action specified for the UR is standard, RRS performs backout processing for the UR when commit is requested. If the failure action is forget, RRS takes no action.
In-flight	If the failure action specified for the UR is standard, RRS performs backout processing for the UR. If the failure action is forget, RRS takes no action.

Table 4. UR States and Failure Actions (continued)

UR state at time of failure	Action
In-state-check	If the failure action specified for the UR is standard, RRS performs backout processing for the UR and returns RR_BACKED_OUT_OUTCOME_PENDING to the application. If the failure action is forget, RRS continues as if the resource manager was not associated with the UR. Note that a forget failure action is only valid for unprotected interests.
In-prepare	If the failure action specified for the UR is standard, RRS performs backout processing for the UR and returns RR_BACKED_OUT_OUTCOME_PENDING to the application. If the failure action is forget, RRS continues as if the resource manager was not associated with the UR. Note that a forget failure action is only valid for unprotected interests.
In-doubt	When the UR state is resolved, RRS performs commit or backout processing.
In-commit	RRS returns RR_COMMITTED_OUTCOME_PENDING to the application. For an unprotected interest, RRS returns RR_OK.
In-backout	RRS returns RR_BACKED_OUT_OUTCOME_PENDING to the application. If the failing resource manager had an unprotected interest, RRS returns RR_BACKED_OUT.
In-end, **in-completion**, or **in-forget** with a commit collective vote	RRS returns RR_COMMITTED_OUTCOME_PENDING to the application. If the failing resource manager had an unprotected interest, RRS returns RR_OK.
In-end, **in-completion**, or **in-forget** with a backout collective vote	RRS returns RR_BACKED_OUT_OUTCOME_PENDING to the application. If the failing resource manager had an unprotected interest, RRS returns RR_BACKED_OUT.
In-only-agent	RRS returns RR_BACKED_OUT_OUTCOME_PENDING to the application. If the failing resource manager had an unprotected interest, RRS returns RR_BACKED_OUT.

Private context delegation

RRS does not allow a work manager to terminate when the address space owns contexts that have in-doubt units of recovery associated with them. This restriction causes address space termination to hang until each in-doubt unit of recovery is resolved. This delay will prevent the work manager from restarting with RRS in a different address space. To allow the work manager to terminate while in-doubt units of recovery exist, the work manager can use private contexts and request private context delegation to RRS. Private context delegation allows a work manager to designate another resource manager that will assume ownership of privately-managed contexts when the work manager terminates.

RRS will only assume ownership of privately-managed contexts that have an associated unit of recovery with an SDSRM interest. RRS does not assume ownership of any other privately-managed contexts. If RRS does not assume ownership of a privately-managed context, that context will be ended by Context Services.

Table 5 describes how Context Services processes a privately-managed context:

Table 5. Processing a Privately-Managed Context

Situation	Outcome
Context switched to a task in the owning space, owning space fails	Context Services asks RRS to assume ownership of the context. If RRS will accept ownership: • RM Context_Switch exits are driven. • If the switch is not rejected, RRS becomes the owner of the context. Otherwise, the context is ended. If RRS does not accept ownership, the context is ended.
Context switched to a task in a non-owning space, owning space fails	The context will be marked for deferred delegation to RRS. The context is neither ended nor delegated to RRS.
Context switched to a task in a non-owning space, owning space has already failed, non-owning space fails	Context Services asks RRS to assume ownership of the context. If RRS will accept ownership: • RM Context_Switch exits are driven. • If the switch is not rejected, RRS becomes the owner of the context. Otherwise, the context is ended. If RRS does not accept ownership, the context is ended.
Context not switched to any task, owning space fails	RM Context_Switch exits are driven. If the switch is not rejected, Context Services asks RRS to assume ownership of the context. If RRS will accept ownership, RRS becomes the owner of the context. If RRS does not accept ownership, the context is ended.
Note: If the owning space and the space where the privately-managed context is switched are failing at the same time, the privately-managed context might be ended before it can be delegated to RRS.	

When Context Services asks RRS to assume ownership of a privately-managed context, RRS will take different actions depending on the state of the UR associated with the privately-managed context. Table 6 describes the outcome of the context switch depending on the UR state.

Table 6. RRS Processing of a UR associated with a Privately-Managed Context

UR state	Outcome
In-flight In-state-check In-prepare	RRS does not assume ownership of the context. Context Services will end the context. RRS forces the unit of recovery to be backed out.

Table 6. RRS Processing of a UR associated with a Privately-Managed Context (continued)

UR state	Outcome
In-doubt	RRS assumes ownership of the context. RRS ends the context after the SDSRM resolves the unit of recovery by calling either the ATRACMT or the ATRABAK callable service.
In-only-agent In-commit In-backout In-end In-completion In-forget Forgotten	RRS does not assume ownership of the context. Context Services will end the context.

When RRS restarts

If RRS fails, a resource manager might receive a return code of ATR_NOT_AVAILABLE from a call to an RRS service. If the resource manager has set a NOTIFICATION exit routine for RRS, the system invokes the routine to notify the resource manager when RRS has failed and again when RRS has restarted. The NOTIFICATION exit routine should call Set_Exit_Information to reset the RRS exit routines and perform restart processing.

When RRS restarts after a failure, its records are in the RRS logs. From these logs, RRS re-creates the state of all incomplete protected URs and protected interests in them.

RRS uses the re-created URs and interests to inform the resource managers, which must each reset its RRS exit routines and go through restart processing, of their outstanding obligations and to complete the processing of the incomplete URs.

Restarting

The processing steps required to start a resource manager for the first time or to restart it at a later time are basically the same.

When a resource manager restarts, RRS requires it to use the same resource manager name that it used when it was previously running. This requirement exists because RRS uses the resource manager name to identify incomplete URs across failures that involve the resource manager. The name must be unique in a sysplex.

After a resource manager has set its exit routines, it must:

- Check that the logs being used are the same as the logs used previously
- Obtain any interests in URs that were left as incomplete when it or RRS previously stopped running
- Respond to any incomplete interests, based on the information returned from RRS and the information in the resource manager logs.

  RRS does not return the incomplete URs in any particular order. If the resource manager expressed protected interest several times in a UR, RRS returns each interest separately with its unique UR interest token.

  Table 7 on page 56 describes the recovery records RRS returns to the resource manager when it restarts after a failure, whether the failure was caused by the

resource manager, RRS, or the system. See "Configuring and defining RRS logging requirements" on page 537 for more information about records logged by RRS.

Once a resource manager has set its exits with RRS, it is considered active on the system. If the resource manager cannot successfully restart on the same system it was running on before it failed, then it needs to unregister itself as a resource manager on that system prior to restarting on another system.

Table 7. UR States and Recovery Records

UR State at time of failure	Recovery records at restart
In-reset	Nothing.
In-flight	Nothing.
In-state-check	Nothing.
In-prepare	For presumed nothing protocol, **in-backout** information. Otherwise, nothing. For presumed abort protocol, nothing.
In-doubt	**In-doubt** information. If the UR state is resolved before the resource manager restarts, **in-commit** or **in-backout** information.
In-commit	**In-commit** information.
In-backout	For a presumed nothing expression of interest if RRS has logged an **in-prepare** record, **in-backout** information. Otherwise, nothing. For presumed abort protocol, when RRS has logged an **in-doubt** record, **in-doubt** information. Otherwise, nothing.
In-end, **in-completion**, or **in-forget** with a commit collective vote	**In-commit** information.
In-end, **in-completion**, or **in-forget** with a backout collective vote	For a presumed nothing expression of interest if RRS has logged an **in-prepare** record, **in-backout** information. Otherwise, nothing. For presumed abort protocol, when RRS has logged an **in-doubt** record, **in-doubt** information. Otherwise, nothing.
In-only-agent	Nothing. The resource manager must use its own logs to process the UR at restart. The resource manager should try to back out because RRS returned BACKOUT_ OUTCOME_PENDING to the application. If backout is not possible, the resource manager should notify installation personnel.

When RRS indicates that there are no more incomplete URs, the resource manager ends restart. Once the resource manager ends restart (calls the End_Restart service), it can begin to process new URs.

The services the resource manager uses during restart are:

Callable service	Description
Retrieve_Log_Name	Retrieve the resource manager log name and the RRS log name as a check that the restart logs are the same as the previously used logs.
Set_Log_Name	If the resource manager is starting for the first time, provide the resource manager log name to RRS, in preparation for a future restart.
Begin_Restart	Begin the restart process.
Retrieve_UR_Interest	Retrieve an incomplete UR interest. The resource manager should repeat these calls until RRS indicates it has no more incomplete URs.
End_Restart	End the restart process. The resource manager must issue this call before it can process any new URs.
Respond_to_Retrieved_Interest	Respond to each retrieved incomplete UR. If the resource manager does not complete its processing of a retrieved UR, RRS returns the UR again when the resource manager next restarts.

For information on the calls, see Chapter 7, "Callable resource recovery services," on page 227.

Resource manager restart environments

In z/OS V1R6, RRS removes all resource manager restart affinity restrictions. This enables a resource manager to restart on any z/OS V1R6 system in the same logging group whenever the resource manager terminates:

- independent of whether RRS or the system fails
- independent of whether the terminating resource manager has incomplete interests or not.

There are four different environments for a resource manager restart. Table 8 lists the environments and any RRS restrictions on where the resource manager can restart.

Table 8. Restart Environments and Restrictions

Environment for the restart	Restrictions
The resource manager has no protected interest in any incomplete UR. For example, the resource manager can be starting for the first time.	No restart restrictions. The resource manager can start on any system in the logging group.

Table 8. Restart Environments and Restrictions (continued)

Environment for the restart	Restrictions
The resource manager has an incomplete protected interest in one or more URs. RRS has remained active on the system where the resource manager was last active.	1. The resource manager restarts on a z/OS V1R6 system: • The system where the resource manager failed is z/OS V1R6 or above: – No restart restrictions. The resource manager can restart on this system. • The system where the resource manager failed is z/OS V1R5 or below: – The resource manager must restart on the same system. RRS will fail any attempt by this resource manager to restart on a different system. 2. The resource manager restarts on a z/OS V1R5 or below system: • The resource manager must restart on the same system. RRS will fail any attempt by this resource manager to restart on a different system.
The resource manager has an incomplete protected interest in one or more URs. RRS did not remain active on the system where the resource manager was last active, but RRS might have already restarted on the system where it failed. From the group of resource managers that have protected interests in a common set of URs, no resource managers are currently active with RRS.	1. The resource manager restarts on a z/OS V1R2 or above system: • No restart restrictions. The resource manager can restart on this system. 2. The resource manager restarts on a z/OS V1R1 or below system: • The resource manager can restart on this system, but this force the other resource managers in the same resource manager group to restart on this system also.
The resource manager has an incomplete protected interest in one or more URs. RRS did not remain active on the system where the resource manager was last active, but RRS might have already restarted on the system where it failed. From the group of resource managers that have protected interests in a common set of URs, one or more managers are currently active with RRS.	1. The currently active resource manager(s) restarted on a z/OS V1R2 or above system: • No restart restrictions. The resource manager can restart on this system. 2. The currently active resource manager(s) restarted on a z/OS V1R1 or below system: • The resource manager must restart on the system where the rest of the resource manager group is active.

Log name checks

When a resource manager restarts, it obtains information from RRS to verify that RRS can provide the state of resources at the time the resource manager was last active. Because RRS keeps information about resources in coupling facility log streams, the resource manager can verify that the log streams RRS is now using are the same as when the resource manager was last active. If the log streams RRS is using do not match the log streams the resource manager expects, then the resource manager might need to shut down.

To verify that the RRS log streams are current, the resource manager uses RRS services to compare the log name of the current log streams with the log name it expects. Note that the log name is not the name of an actual log stream but a constant, something like a token, that RRS uses to identify a particular set of log

streams. A resource manager can also create and maintain a resource manager log name to identify the set of logs that it uses

During restart, the resource manager can then obtain information about log names, which involves two calls:

- The resource manager issues a call to Retrieve_Log_Name to retrieve the current RRS log name. If the call also retrieves the resource manager log name stored by RRS, the resource manager knows that it is restarting. If the call returns ATR_RM_LOGNAME_NOT_SET, the resource manager knows that it is performing a cold start.
- If the resource manager is performing a cold start, it issues a call to Set_Log_Name to tell RRS the name of the resource manager log. RRS stores the name.

Later, when restarting, the resource manager can compare the following:

- Information known to the resource manager:
  - The current resource manager log name
  - The RRS log name that the resource manager stored in its log after calling Retrieve_Log_Name when the resource manager was last active
- Information returned by the Retrieve_Log_Name service:
  - The resource manager log name previously stored by RRS
  - The current RRS log name

If the names match, the resource manager can continue restart, because the correct logs are being used. Table 9 indicates the possible comparisons, the meaning of each, and the possible actions the resource manager or installation personnel might take.

Table 9. Log Name Checking

Comparison	Meaning	Action
Stored RRS log name = RRS log name just retrieved Current resource manager log name = resource manager log name just retrieved	Logs match.	Continue with normal restart.
Stored RRS log name ≠ RRS log name just retrieved Current resource manager log name = resource manager log name just retrieved	RRS is using an old log.	Check for incorrect logs being used during restart and correct the condition.
Stored RRS log name = RRS log name just retrieved Current resource manager log name ≠ resource manager log name just retrieved	The resource manager is using an old log.	Check for incorrect logs being used during restart and correct the condition.
Stored RRS log name ≠ RRS log name just retrieved Current resource manager log name ≠ resource manager log name just retrieved	The RRS logs and the resource manager logs do not match.	Check for incorrect logs being used during restart and correct the condition.

Table 9. Log Name Checking (continued)

Comparison	Meaning	Action
Stored RRS log name = RRS log name just retrieved The resource manager log name is not available from Retrieve_Log_Name	RRS or the resource manager failed before the resource manager log name was recorded.	Continue with normal start.
Stored RRS log name is not available from resource manager log Current resource manager log name = resource manager log name just retrieved	RRS or the resource manager failed before the RRS log name was recorded.	Continue with normal start.
Stored RRS log name is not available from resource manager log The resource manager log name is not available from Retrieve_Log_Name	RRS and the resource manager are both doing cold starts, or the resource manager is starting for the first time.	Continue with normal start.
Stored RRS log name is not available from resource manager log The resource manager log name is not available from Retrieve_Log_Name	After a successful Internal Cold Start, in response to a log stream error against the RRS RM.DATA log stream as identified by message ATR250E, Retrieve_Log_Name could return code ATR_RM_LOGNAME_NOT_SET.	Continue with normal start.
Stored RRS log name ≠ RRS log name just retrieved The resource manager log name is not available from Retrieve_Log_Name	RRS is doing a cold start.	If the cold start is expected, continue with normal restart; otherwise, report the error to the support center for the resource manager, and do not restart the resource manager.
Stored RRS log name was not available from resource manager log Current resource manager log name ≠ resource manager log name just retrieved	The resource manager is doing a cold start.	If the cold start is expected, continue with normal restart; otherwise, report the error to the support center for the resource manager, and do not restart the resource manager.

Expressing interest in a UR

When an application program requests access to a resource, the resource manager that owns the resource needs to express an interest in the UR. The expressed interest tells RRS that the resource manager is involved with the UR. The interest can be protected, which means the changes require coordination, or unprotected, which means the changes do not require coordination.

In response, RRS invokes the resource manager's exit routines in the syncpoint processing of the UR, enabling the resource manager to protect its resource.

Most database accesses are read-only accesses. For a read-only access, the resource manager should express an unprotected interest in the UR. The resource manager's exit routines are invoked; however, if a failure occurs, RRS does not needlessly inform the resource manager that it previously expressed interest in the UR.

If your resource manager expects to process multiple URs for the same work context, see Chapter 3, “Using context services,” on page 31.

The services related to expressing interest in a UR are:

Callable service	Description
Express_UR_Interest	Express an interest in a UR and provide persistent and nonpersistent interest data.
Retrieve_Interest_Count	Determine if resource managers have expressed more than one interest in a UR.
Change_Interest_Type	Change the interest type from unprotected to protected.
Retain_UR_Interest	Express interest in the next UR for the current context.
Delete_UR_Interest	Delete an interest in a UR.

For more information, see Chapter 7, “Callable resource recovery services,” on page 227.

On Express_UR_Interest, you specify the type of two-phase commit protocol for the UR, and there are additional techniques you can use to optimize RRS processing.

Types of two-phase commit protocols

A call to the Express_UR_Interest service can specify the type of two-phase commit protocol to be used for a UR. The value specified affects RRS processing if the resource manager has to restart:

- **Presumed nothing**: For a presumed nothing interest in a protected UR, RRS logs an **in-prepare** record, including the persistent interest data RRS passes to the exit routines, in the RRS log before invoking the PREPARE exit routines. During restart, RRS returns such a UR when the resource manager calls the Retrieve_UR_Interest service. RRS tells the resource manager that the UR is to be backed out. The resource manager uses the persistent interest data during the backout.

  If one protected interest in a UR is presumed nothing, RRS uses the presumed nothing protocol. If there is only one presumed nothing protected interest in a UR and this interest is by a distributed syncpoint resource manager, RRS does not log an **in-prepare** record.
- **Presumed abort**: When the UR is in the **in-prepare** state, RRS does not harden any information in the RRS log about the UR. During restart, RRS cannot return such a UR when the resource manager calls the Retrieve_UR_Interest service. The resource manager presumes the UR was backed out.

Other optimizations

In addition to the presumed abort protocol, RRS provides additional protocol optimizations, including the following:

- **Only agent**: If the resource manager provides an ONLY_AGENT exit routine and only the resource manager expresses only one interest in the UR, RRS invokes its ONLY_AGENT exit routine, skipping:
  - Invoking the PREPARE exit routine
  - Invoking the COMMIT exit routine

  - Invoking the END_UR exit routine
  - Hardening any information about the UR in the RRS log

  Note that RRS does not drive the ONLY_AGENT exit routine when the resource manager for the UR is a server distributed syncpoint resource manager (SDSRM).
- **Read-only exit routine minimization**: If the PREPARE exit routine completes its processing for the UR, because the request was read only, the routine should return an ATRX_FORGET return code. RRS assumes the resource manager concurs with the commit. RRS does not invoke additional exit routines for that interest in the UR.
- **All read only**: If the PREPARE exit routines of all of the resource managers interested in the UR return an ATRX_FORGET return code, because all requests are read only, RRS immediately completes the UR without hardening a commit decision in the RRS log.

  For distributed resources, the read-only optimization reduces network flows.

The following optimization applies only to distributed resources:

- **Sending and receiving last agent**: A system uses *sending last agent* to pass responsibility for determining the final outcome of a commit decision to the receiving system. The sending system informs the receiving system that it can commit the work request.

  Because the receiving system knows that the sending system can commit, the network flow required for the prepare phase on the two systems is eliminated.

Protecting the resource

When the resource manager processes the UR, it needs to prepare for changes to its resource, which includes two steps:

- Log the unchanged data
- Log the potential changed data.

If all resource managers vote YES to indicate that they can make the changes, RRS instructs each resource manager to make the changes; only then does the resource manager make the changes. If any resource manager votes NO to indicate that it cannot make the changes, RRS instructs all resource managers to not make the changes; the resource managers backout the changes.

The services used to protect the resource are:

Callable service	Description
Retrieve_UR_Data	Retrieve UR data, including the UR identifier (URID).
Set_Persistent_Interest_Data	Provide persistent interest data.
Retrieve_UR_Interest_Data	Retrieve persistent and nonpersistent interest data.
Post_Deferred_UR_Exit	Give to RRS the response from a resource manager exit routine that previously replied with a defer return code. A later response might be needed in a distributed environment.

For information on the services, see Chapter 7, "Callable resource recovery services," on page 227.

The exit routines used to protect the resource are:

Exit Routine	Event for Invoking the Routine
STATE_CHECK	An application program called the Application_Commit_UR service.
PREPARE	RRS is ready for the resource managers to prepare for resource recovery.
ONLY_AGENT	Only one resource manager expressed only one interest in the UR.
DISTRIBUTED_SYNCPOINT	When a resource manager has taken the DSRM role for an interest in a UR and the UR becomes **in-doubt**, this exit resolves the **in-doubt** condition
COMMIT	The resource manager is to make the changes permanent.
BACKOUT	The resource manager is to not make the changes, thus, backing them out.
END_UR	The UR processing is finished.
EXIT_FAILED	An RRS exit routine in the resource manager failed.
COMPLETION	The exit routine can do processing needed before RRS returns control to the application program.
SUBODINATE_FAILED	Either RRS or any resource manager on a subordinate system failed, the subordinate system itself terminated, or the context associated with the subordinate UR abnormally terminated.

For information on the exit routines, see "Resource recovery exit routines" on page 91.

Vote collection

RRS invokes the PREPARE exit routine to notify the resource managers to prepare for changes to their resources, and each exit routine issues a return code that, like a vote, indicates to RRS how the syncpoint operation should proceed.

RRS collects the votes (checks the return codes set by PREPARE exit routines) from all the resource managers that have expressed an interest in the UR, then determines the overall results.

The possible overall results are:

- **OK or YES**: At least one PREPARE exit routine returned OK or HC. None returned BACKOUT, HR, or HM.
- **Forget or YES**: All PREPARE exit routines returned FORGET or ABSTAIN. None returned BACKOUT, HC, HM, HR, or OK.

  If the return code indicates FORGET and no resource manager returned ABSTAIN, the UR is complete.

If a resource manager has taken the DSRM role and the return code is FORGET, but at least one resource manager returned ABSTAIN, RRS invokes the END_UR exit routine, rather than the DISTRIBUTED_SYNCPOINT exit routine.

- **Backout or NO**: At least one resource manager returned BACKOUT or HR. None returned HC or HM.
- **Heuristic mixed or NO**: Any of the following:
  - A resource manager returned HM.
  - A resource manager returned HC and the result is backout.
  - A resource manager returned HR and the result is commit.

For information about the various codes a PREPARE exit routine can return, see "Return codes" on page 128.

UR states

During processing, a UR assumes different states. In each UR state, a resource manager can issue certain callable services. Table 10 lists each UR state and the callable services that apply. UR states are not related to resource manager states.

Table 10. UR States and Callable Services Allowed

UR state	Description	Callable services allowed
In-reset	The UR is starting. The application has not yet changed any resources.	Backout_UR Commit_UR Create_Cascaded_UR Delete_Post_Sync_PET Express_UR_Interest Retrieve_Interest_Count Retrieve_Log_Name Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Post_Sync_PET
In-flight	The application accesses resources. The application has potential changes to resources, but the changes are not committed. The resource managers for the resources express interest in the UR. Note that only one UR in a context can be in the **in-flight** state at a time.	Application_Backout_UR Application_Commit_UR Backout_Agent_UR Backout_UR Change_Interest_Type Commit_UR Delete_Post_Sync_PET Delete_UR_Interest Express_UR_Interest Prepare_Agent_UR Retrieve_Interest_Count Retrieve_Interest_Data Retrieve_Log_Name Retrieve_Side_Information Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Persistent_Interest_Data Set_Post_Sync_PET Set_Side_Information Set_Syncpoint_Controls Set_Work_Identifier

Table 10. UR States and Callable Services Allowed (continued)

UR state	Description	Callable services allowed
In-state-check	The application program issues a commit. Before starting prepare, the resource managers' STATE_CHECK exit routines check that the resources are in the correct state. RRS stops the commit if the return code from a routine tells RRS to stop it. Some reasons why a routine would request stopping the commit are: • A resource on the same system as the application is not in the proper state for a commit. • A protected conversation is not in the required state: **send**, **send pending**, **defer receive**, **defer allocate**, **sync_point**, **sync_point send**, **sync_point deallocate**. • A protected conversation is in **send** state. The communications manager started sending the basic conversation logical record, but did not finish sending it.	Change_Interest_Type Post_Deferred_UR_Exit Retain_Interest Retrieve_Interest_Count Retrieve_Interest_Data Retrieve_Log_Name Retrieve_Side_Information Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Persistent_Interest_Data Set_Side_Information Set_Syncpoint_Controls Set_Work_Identifier
In-prepare	The application program in the proper state issues a commit. In response, RRS begins the two-phase commit process. RRS invokes the PREPARE exit routines of the interested resource managers; the routines determine if the resource can be changed.	Post_Deferred_UR_Exit Retain_Interest Retrieve_Interest_Count Retrieve_Log_Name Retrieve_Side_Information Retrieve_Interest_Data Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Side_Information Set_Syncpoint_Controls Set_Work_Identifier
In-only-agent	Only one resource manager expressed only one interest in the UR. RRS invokes the resource manager's ONLY_AGENT exit routine to tell the resource manager to process the commit request immediately.	Post_Deferred_UR_Exit Retain_Interest Retrieve_Interest_Count Retrieve_Log_Name Retrieve_Side_Information Retrieve_Interest_Data Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Side_Information Set_Work_Identifier

Table 10. UR States and Callable Services Allowed (continued)

UR state	Description	Callable services allowed
In-doubt	During distributed processing of a UR, the UR is in an **in-doubt** state when a resource manager is coordinating the processing and RRS is waiting for the coordinator to tell it whether to resolve the UR by a commit or a backout. During the **in-doubt** state, a resource manager cannot make a unilateral decision to commit or backout changes to its resource.	Commit_Agent_UR Backout_Agent_UR Post_Deferred_UR_Exit Respond_to_Retrieved_Interest Retain_Interest Retrieve_Interest_Count Retrieve_Log_Name Retrieve_Side_Information Retrieve_Interest_Data Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Side_Information Set_Work_Identifier
In-commit	One of the following occurred: • The PREPARE exit routines replied YES. • The DSRM or SDSRM told RRS to commit an **in_doubt** UR. • The installation used the RRS panels to commit an **in_doubt** UR. RRS notifies the resource managers to make the changes in the resources permanent. The resource managers indicate that the changes have been made.	Forget_Agent_UR_Interest Post_Deferred_UR_Exit Respond_to_Retrieved_Interest Retain_Interest Retrieve_Interest_Count Retrieve_Interest_Data Retrieve_Log_Name Retrieve_Side_Information Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Side_Information Set_Work_Identifier
In-backout	One of the following occurred: • One or more PREPARE exit routines replied NO. • The application issued a backout. • The DSRM or SDSRM told RRS to backout an **in_doubt** UR. • The installation used the RRS panels to backout an **in_doubt** UR. • Before phase 2 of the two-phase commit protocol, the system, application, RRS, or a resource manager failed. RRS notifies the resource managers to **not** make the changes in the resources. The resource managers indicate that the changes have **not** been made.	Forget_Agent_UR_Interest Post_Deferred_UR_Exit Respond_to_Retrieved_Interest Retain_Interest Retrieve_Interest_Count Retrieve_Log_Name Retrieve_Side_Information Retrieve_Interest_Data Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Side_Information Set_Work_Identifier

Table 10. UR States and Callable Services Allowed (continued)

UR state	Description	Callable services allowed
In-end	The resources have been updated.	Forget_Agent_UR_Interest Post_Deferred_UR_Exit Retain_Interest Retrieve_Interest_Count Retrieve_Log_Name Retrieve_Side_Information Retrieve_Interest_Data Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Side_Information Set_Work_Identifier
In-completion	The resources have been updated, and RRS has completed its processing of the UR.	Forget_Agent_UR_Interest Post_Deferred_UR_Exit Retrieve_Interest_Count Retrieve_Log_Name Retrieve_Side_Information Retrieve_Interest_Data Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Side_Information
In-forget	During distributed processing, the UR has completed but RRS is waiting for the SDSRM to indicate how to process the log records for the UR.	Forget_Agent_UR_Interest Retrieve_Interest_Count Retrieve_Log_Name Retrieve_Side_Information Retrieve_Interest_Data Retrieve_UR_Data Retrieve_Work_Identifier Set_Log_Name Set_Side_Information
Forgotten	The UR has completed, and RRS has deleted its log records.	

If any resource manager has a protected interest in a UR, RRS keeps track of the UR's state and records the state in a log. Thus, after a failure, RRS knows the state of the UR and is aware if a UR is incomplete. When the resource manager restarts, RRS provides the resource manager with information about the incomplete UR, so that the resource manager can complete its processing.

Protecting distributed resources

Using the peer-to-peer (DSRM) model for distributed resource recovery, an application's work request across distributed systems is connected by Systems Network Architecture (SNA) LU 6.2 conversations handled by a communications manager such as APPC/MVS. Each work request has a logical unit of work identifier (LUWID).

When the first protected conversation is allocated for a work request, RRS generates an LUWID for it. The resource manager handling the outbound protected conversation uses a call to the Retrieve_Work_Identifier service to obtain

the LUWID and passes the LUWID in the conversation. When the application accepts an inbound protected conversation, the resource manager handling it uses a call to the Set_Work_Identifier service to set the LUWID.

A resource manager can also use a call to the Set_Work_Identifier service to set the next LUWID for transaction chaining. In this case, RRS generates the LUWID for the next UR based on the current and next LUWIDs for the current UR. RRS generates the LUWID before returning control to the application program following a commit or backout call and before invoking any COMPLETION exit routines.

Resource managers in distributed processing need the following services:

Callable service	Description
Set_Syncpoint_Controls	Define the resource manager's role in supporting a distributed environment for protected resources. This call is usually issued in the resource manager's STATE_CHECK or PREPARE exit routine.
Set_Side_Information	Set the side information for an interest in a UR.
Retrieve_Side_Information	Retrieve the side information for an interest in a UR.
Set_Work_Identifier	Set the unit of work identifier (UWID), which is an LU 6.2 logical unit of work identifier (LUWID) or an Enterprise identifier (EID).
Retrieve_Work_Identifier	Retrieve the unit of work identifier (UWID), which is an LU 6.2 logical unit of work identifier (LUWID) or an Enterprise identifier (EID).

For more information, see Chapter 7, “Callable resource recovery services,” on page 227.

A resource manager that has taken the server distributed syncpoint resource manager (SDSRM) role for an interest in a UR additionally needs the following calls:

Callable service	Description
Prepare_Agent_UR	Start the prepare phase of a syncpoint operation for a unit of recovery
Backout_Agent_UR	backout a unit of recovery
Commit_Agent_UR	Commit a unit of recovery
Forget_Agent_UR_Interest	Forget a unit of recovery

For information on the calls, see Chapter 7, “Callable resource recovery services,” on page 227.

Cascaded transactions

A cascaded transaction is a type of distributed transaction, or part of a distributed transaction, in which the coordination between the units of recovery is controlled by RRS. If you have a set of units of recovery, each with a separate work context, which need to be coordinated as a single transaction within a single commit scope, you can group the URs together by performing RRS calls to create a single cascaded transaction. A group of URs related in this way is called a cascaded UR family.

Cascaded UR families

A cascaded UR family is created when a work manager tells RRS to cascade a new UR from an existing UR. A work manager can do that with either the Create_Cascaded_UR service or the Express_UR_Interest service. Figure 14 shows a logical view of a sample cascaded UR family.

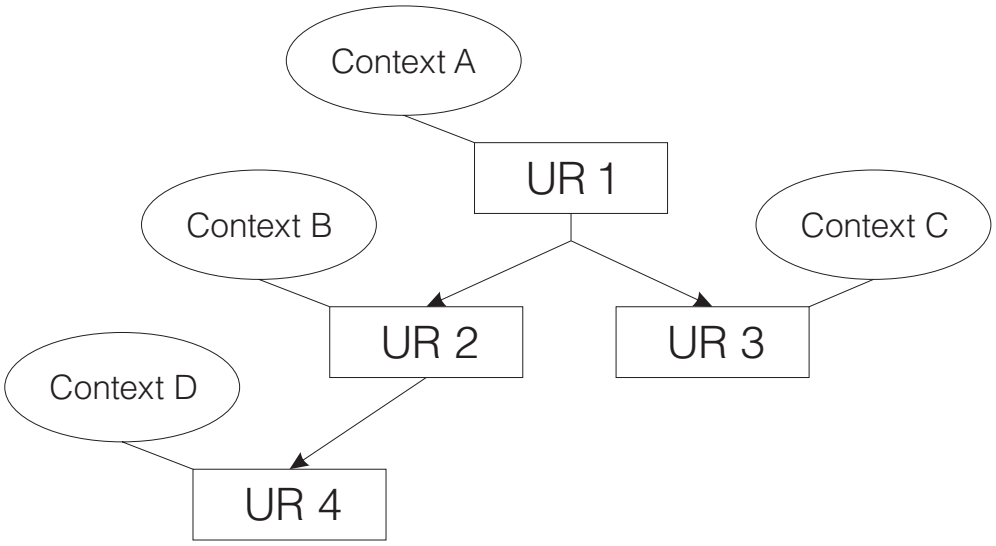

Figure 14. A Sample Cascaded UR Family

The existing UR is called the *parent* of the cascaded UR. The cascaded UR is called a *child* of the parent UR. The URID of a child UR is different from the URID of its parent. Each UR in a cascaded UR family will have a unique LUWID and EID if they have these work identifiers. Each UR will also have an XID with a unique BQUAL, but the FORMATID and GTRID components of the XID are the same for each UR in a cascaded UR family. Every member of a cascaded UR family will have at least an XID as a work identifier. See “Unit of work identifiers” on page 81 for more information about these work identifiers.

Multiple cascaded URs can be associated with the same parent UR. The children of one parent are called *siblings*.

Cascaded URs can be embedded in other cascaded URs to any level. The *ancestors* of a UR are the parent of the cascaded UR and, recursively, the parents of its ancestors. The *descendants* of a UR are the children of the UR and, recursively, the children of its descendants.

A *top-level* UR is one with no parent. A top-level UR and all of its descendants are called a *cascaded UR family*, or sometimes simply a *UR family*.

In Figure 14 on page 69, UR 1 is the top-level UR. UR 2 and UR3 are its children, and siblings of each other. UR 2 is the parent of UR 4. UR 4's ancestors are UR 2 and UR 1. UR 2, UR 3, and UR 4 are all descendants of UR 1.

Note: Context Services has no awareness of any relationship between the contexts. They are only related by the URs being managed by RRS.

Understanding cascaded transactions

A work manager typically needs to create a cascaded UR when a single work request involves multiple work managers. The initial work manager would obtain the initial work context that represented the work request. When the work request moved from the execution environment of the original work manager into a second work manager's environment, the second work manager could obtain a new work context to represent the work request, allowing it to manipulate the work context.

The work manager that creates a cascaded UR is responsible for informing RRS when the application running under the UR is complete by calling the Set_Side_Information service specifying the ATR_APPL_COMPLETE information identifier. The work manager must do this so that RRS can know when all of the parts of a cascaded UR family are ready for syncpoint processing. When a cascaded UR is created, it is not application-complete. An application is application-complete after it has finished its processing and returned any results to its caller, but before its protected resources have been committed. RRS does not allow top-level URs to be marked application-complete or application-incomplete. RRS automatically considers a top-level UR to be complete when backout or commit processing is initiated for the UR.

A cascaded UR is similar to a top-level UR in that changes made on behalf of a cascaded UR are either all committed or all rolled back. However, a cascaded UR cannot be committed by itself. In order to make the changes of a cascaded UR permanent, a request must be made to commit the top-level UR of the cascaded UR's UR family.

Similarly, the changes made by all of the URs in a UR family are either committed or backed out. If any of the URs in a UR family are backed out, all of the other URs in that family will be backed out as well. RRS will wait until all of the URs in the UR family are application-complete before committing the UR family. If a backout is requested against a cascaded UR, RRS will consider that UR to be application-complete and will immediately backout that individual UR.

See “Working with cascaded transactions” on page 556 for more information about working with cascaded transactions in application programs.

Note: The cascaded UR relationship is not the same as the subtransaction relationship defined by nested transactions. Nested transactions allow subtransactions to commit without committing the top-level transaction. When a subtransaction backs out, its descendants are backed out, but not its ancestors. Additionally, locks used to serialize access to resources are shared across nested subtransactions, but they may not be shared across cascaded URs.

Multisystem cascaded transactions

A cascaded transaction can exist across multiple systems in a sysplex, as long as all of the systems involved in the transaction use the same RRS logging group. A cascaded transaction that has URs on multiple systems in a sysplex in which the

cross system coordination is being provided by RRS is known as a *multisystem cascaded transaction* (sometimes also called a *sysplex cascaded transaction* or a *sysplex cascade*).

As with normal (non-multisystem) cascaded transactions, a work manager that creates a multisystem cascaded transaction is responsible for informing RRS when the part of the application executing under a multisystem cascaded UR is complete by using the Set_Side_Information service to mark the UR as application-complete.

The top-level UR of a multisystem cascaded transaction is the UR first built to represent the original work request. The system where the top-level UR of a particular multisystem cascaded transaction resides is called the *coordinator* of that multisystem cascaded transaction. A system where a multisystem cascaded child UR resides is called a *subordinate*.

Multisystem cascaded transactions work very similarly to normal cascaded transactions, but you need to be aware of additional database locking and work manager considerations when you work with cascaded transactions across a sysplex. See "Working with cascaded transactions" on page 556 for more information about the application and work manager considerations of working with multisystem cascaded transactions.

End context processing with cascaded transactions

When working with cascaded URs, contexts become grouped through the URs they are associated with. During the lifetime of a cascaded UR family, the contexts associated with any of the members of the family may end, either normally or abnormally. What RRS does when one of the contexts ends depends on the state the UR is in when the context ends. Those states are grouped into 3 phases:

Table 11. Phases of Ending Context States

Phase	States
0/1	**in-flight**, **in-state-check**, **in-prepare**
Doubt	**in-doubt**
2	**in-commit**, **in-backout**, **in-end**, **in-completion**, **in-forget**
Note: in-only-agent is not considered, because a member of a cascaded UR family can never be in that state. **in-reset** and **forgotten** are not considered because those states indicate that there is no UR to process.	

For each phase shown in Table 11, there are 3 possible ways the context could terminate: normal termination, abnormal termination, or memory termination.

Normal context termination

May occur due to one of the following:

- An explicit End_Context call
- The task that the context is associated with ends normally

Abnormal context termination

May occur due to one of the following:

- An explicit End_Context call
- The task that the context is associated with ends abnormally
- A task RRS is executing syncpoint processing under is asynchronously abended

Memory context termination

May occur due to one of the following:

- Private context owner termination
- Client address space termination

Note: A client address space is one in which a task has a context that has a UR associated with it.

Table 12 shows how RRS responds to each possible condition for each phase.

Table 12. Context Termination Processing

Phase	Condition	Top-level UR terminating	Cascaded UR terminating
0/1	Normal termination — environment indicates commit on normal termination	Implicit commit processing, unless a RM has taken the SDSRM role. If the UR has an SDSRM, application backout processing.	Application backout processing
	Normal termination — environment indicates rollback (backout) on normal termination	Implicit backout processing	Application backout processing
	Abnormal termination	Implicit backout processing	Application backout processing
	Memory termination	Implicit backout processing	Application backout processing
Doubt	Any termination	Operator intervention	Operator intervention
2	Any termination	Continue syncpoint	Continue syncpoint

Application backout processing
RRS considers the terminating UR application-complete. RRS backs out the UR, and marks the entire cascaded UR family backout-required. Each cascaded UR in the family will eventually be backed out. The top-level UR is backed out when the application initiates syncpoint or when the top-level UR's context terminates.

Implicit commit processing
RRS goes through normal syncpoint processing, as if the application had issued Application_Commit_UR normally.

Implicit backout processing
RRS goes through normal backout processing, as if the application had issued Application_Backout_UR normally.

Operator intervention
If the failure occurred due to a CANCEL, RRS issues a WTOR message to the operator. The operator can reply COMMIT, BACKOUT, or WAIT. If the operator replies COMMIT or BACKOUT, RRS marks the UR heuristic mixed and goes into phase 2 to commit or backout the UR as instructed.

If the operator replies WAIT, RRS holds up context termination until the operator resolves the in-doubt condition through DSRM, SDSRM, or the system management panels. It is not always possible to specify WAIT.

If the failure was not due to a CANCEL, RRS holds up context termination until the in-doubt condition is resolved by the DSRM or SDSRM, or by the operator through the system management panels.

Continue syncpoint
RRS continues syncpoint processing.

Local transactions

Some software platforms do not provide transaction coordination as part of the kernel operating system. In these environments, a resource manager (RM) typically provides RM-specific transactional functions so that, for at least the resources that the resource manager controls, applications can rely on the data integrity, coherency, and consistency attributes that transactions provide. For example, a resource manager might define RM-specific commit functions, such as MQCMIT and SQLCMIT, that an application can use to commit its resources.

When a syncpoint coordinator is available and coordinating a transaction which could involve multiple resource managers, the transaction is known as a *global transaction*. When, instead, each resource manager involved is seperately coordinating its own changes, and only its changes, the transaction is known as a *local transaction*. In a typical local transaction, the application changes resources owned by a given resource manager (such as using a Java™ Database Connectivity (JDBC) connection for a relational database) and then uses an RM-specific commit function (such as the commit() method for JDBC connections) to commit the changes.

When operating in a local transaction environment, called *local transaction mode*, a resource manager must behave as follows:

- In local transaction mode, an application acts as its own resource coordinator, and each resource manager must behave independently. It is possible that an application, in the absence of a global transaction coordinator, needs to process multiple resources owned by different resource managers. In local transaction mode, each resource manager is independent. The application acts as transaction coordinator and can, if necessary, direct different resource managers to different outcomes.
- A resource manager must treat separate attachments or connections to the same application independently. It is possible that an application might need to process multiple resources owned by the same resource manager. If an application defines two separate attachments or connections to the same resource manager in local transaction mode, the resource manager must treat each connection independently. This independence allows an application to commit some connections to the same resource manager and roll back others.
- A resource manager that defines local commit functions allows its resources to be accessed locally. When in local transaction mode, a resource manager does not permit global commit services like Commit_UR and Application_Commit_UR. If a resource manager does not provide its own local commit functions, a connection to the resource manager should use commit-on-return when in local transaction mode.
- If a connection is made to a resource manager when in local transaction mode, the connection is known as a *local connection*. If a local connection connects the application to a work manager, and the connection touches resources managed

by different resource managers as a result, it is the responsibility of the work manager to act as the coordinator of those resources.

There are also rules that govern the transition between local transaction mode and global transaction mode. There are rules that govern when a transition between local and global transaction mode is allowed:

- Local to global is only allowed when there are no uncommited local connections.
- Global to local is only allowed when the global transaction is **in-reset**.

Global commit and rollback functions, such as SRRCMIT, are not allowed in local transaction mode. Local commit and rollback functions, such as MQCMIT, are not allowed in global transaction mode.

Implementing local transactions on z/OS involves the participation of four separate entities:

The transaction coordinator, RRS. RRS is responsible for keeping track of when there is an active global or a local transaction associated with a given work context. RRS is also responsible for:

- Ensuring that global transaction functions, such as commit and rollback, are not permitted when a local transaction is active for a given work context
- Preventing the start of a global transaction when the current UR state is **in-flight** or beyond
- Rejecting global commit functions against URs which are in local transaction mode, known as *local URs*, which are in any state except **in-reset**
- Notifying resource managers, by driving their appropriate syncpoint exit routines, when a work manager or an application has ended a global transaction

A work manager, such as WebSphere for OS/390. A work manager is responsible for ensuring that the correct transactional environment is established or restored before it dispatches the application. A work manager is also responsible for:

- Ensuring that the default transaction mode is correctly set either for the address space or for each individual context
- Enforcing transaction policy on the application or method exit and invoking End_Transaction to take the appropriate action

A resource manager, such as DB2. A resource manager must manage its resources correctly regardless of whether the transaction mode is local or global. A resource manager is also responsible for:

- Notifying RRS if it supports local transaction mode
- Ensuring that its local transactional functions are not executed when the transaction environment for the work context is global
- Registering its uncommitted local interest with RRS
- Deleting its interest when the local resource is committed or rolled back via the resource manager's local transaction functions
- Correctly using the local transaction flag in its COMMIT, BACKOUT, and COMPLETION exit routines
- Using its own logs to recover its local transactions during restart, because local URs are not written to RRS logs

An application. An application defines the demarcation between local transaction mode and global transaction mode, based on the transaction mode set by a work manager when it dispatches the application.

An example local transaction

Assume that a simple Java method, which uses JDBC to access DB2, is dispatched in a WebSphere for OS/390 server that is deployed with transaction policy of TX_NOT_SUPPORTED. (There are several different transaction policies, called deployment descriptors, that the container, or object server, can enforce upon method dispatch.) TX_NOT_SUPPORTED indicates that, on method dispatch, the current transaction environment is suspended, and a new local transaction environment is established. When the method completes, the original transaction environment is resumed.

1. During initialization, WebSphere for OS/390 (the work manager) and DB2 (the resource manager accessed by JDBC), inform RRS, using the Set_Exit_Information service, that they can support local transactions.
2. WebSphere for OS/390 receives an inbound request to drive a method that requires a TX_NOT_SUPPORTED environment. WebSphere for OS/390 calls the RRS Set_Environment service to set the environment transaction mode to local, providing a default for new transactions started for this context, and then dispatches the method.
3. At the application's request, the JDBC driver accesses DB2 using local transactional semantics.
4. DB2 registers interest in the UR, informing RRS that DB2 has uncommitted local resources associated with the current UR. If an attempt is made to use a global commit function to commit those local resources, RRS disallows it.
5. The method might have created several connections accessing DB2. In this example, assume that the method uses the JDBC-specific connection commit function to commit some of the local connections and uses the JDBC-specific rollback function on others.
6. When the method commits the last local connection, DB2 no longer has uncommitted local resources, so it deletes its last expression of interest in the unit of recovery.
7. This deletion notifies RRS that the method can, if necessary, now start a global transaction.

In this example, RRS, the transaction coordinator, managed the transaction mode. WebSphere for OS/390, the work manager, ensured that the transaction environment was correct for the dispatch of the method. JDBC/DB2, the resource manager, used local transactional behavior to access the database, and the application used the JDBC-specific commit functions to commit or roll back the local connections.

Local and global interactions

RRS assumes that the logic of current z/OS applications depends on a global transaction, so RRS processing is based on implicit global transaction mode. In this mode, a global transaction is always active for a given application, and, if it accesses resources via an RRS-compliant resource manager, those resources are committed globally, along with any other resources the application might have accessed. In this mode, RRS performs a global commit for a transaction that ends normally, and a global rollback for an application that ends abnormally.

Implicit global transaction mode, however, is not appropriate for an application developed on a platform that does not include a global resource coordinator like RRS. In this environment, the application accesses the resources, then uses RM-specific commit or rollback functions to control the outcome. Such an

application requires an implicit local transaction, as well as the ability to switch between local transaction mode and global transaction mode.

Consider the application code segment shown in Figure 15 on page 77. Before the work manager dispatches the application, the work manager must call Set_Environment to set up the transaction mode correctly, dispatching the application initially with local transaction mode as the default. Thus, when the application creates two connections to different databases and accesses them, it is already in local transaction mode. When the application begins a global transaction, the resource managers switch the transaction mode to global.

Processing would be very similar even if the application processed its transactions in a different order. Assume that it performs the global transaction first, followed by the local accesses to the databases. The work manager, before it dispatches the application, calls Set_Environment to initially set local transaction mode as the default, and the application starts a global transaction. The transaction is successful, however, because there are no outstanding uncommitted local connections; the application is free to start a global transaction. Thus, the application can switch between local and global transaction mode without code changes. Two RRS services, Begin_Transaction and End_Transaction, are available for applications that need to clearly mark the boundaries of a transaction. These services allow an application to demarcate the transitions between local transaction mode and global transaction mode.

```
javax.transaction.UserTransaction ut;
javax.sql.DataSource ds1;
javax.sql.DataSource ds2;
java.sql.Connection con1;
java.sql.Connection con2;
java.sql.Statement stmt1;
java.sql.Statement stmt2;
InitialContext initCtx = new InitialContext();
// Obtain con1 object and set it up for transactions
ds1=(javax.sql.DataSource)initCtx.lookup("Database1");
con1=ds1.getConnection;
stmt1=con1.createStatement;
// Obtain con2 object and set it up for transactions
ds2=(javax.sql.DataSource)initCtx.lookup("Database2");
con2=ds2.getConnection;
stmt2=con2.createStatement;
// LOGICAL BEGIN LOCAL TRANSACTION
// start local transaction on con1 and do some work
stmt1.executeQuery(...);
stmt1.executeUpdate(...);
// start local transaction on con2 and do some work
stmt2.executeQuery(...);
stmt2.executeUpdate(...);
// interleave some work on con1
stmt1.executeUpdate(...);
// commit local transaction 2
con2.commit()
// rollback local transaction 1
con1.rollback()
// LOGICAL END LOCAL TRANSACTION

ut = ejbContext.getUserTransaction();
// EXPLICIT BEGIN GLOBAL TRANSACTION
ut.begin();
// Do some updates on both connections
stmt1.executeQuery(...);
stmt1.executeUpdate(...);
stmt2.executeQuery(...);
stmt2.executeUpdate(...);
// commit both connections
ut.commit();
// EXPLICIT END GLOBAL TRANSACTION
con1.close();
con2.close();
```

Figure 15. Application Code Segment

Planning considerations

When deciding if and how to use the local transaction services and processing that RRS provides, consider the following:

Compatibility with RMs that process local transactions. There are additional transaction mode issues because some z/OS resource managers have defined RM-specific commit functions for some time, and these functions have enabled the resource managers to process local transactions. At least some of the processing done by current z/OS resource managers does not mesh completely with RRS.

Examples:

- DB2 can, when using its current attachment to RRS, escalate transaction mode. If an application touches a DB2 resource, DB2 considers the connection to be local.

If the application, however, subsequently touches another resource manager, DB2 escalates the transaction mode of the connection to global.

- Some work managers have what amounts to a permanent expression of interest in a UR for a given privately-managed context. They use the Retrieve_Interest_Count service to determine whether or not they can perform a resource manager local commit operation. If another resource manager does not express interest, the work manager shortcuts the RRS commit process, for performance reasons, and commits the connection(s) in local transaction mode.

To remain compatible with existing applications that exploit these and other proprietary behaviors, RRS defines a special transaction mode, known as hybrid-global transaction mode. This transaction mode is synonymous with global mode with one exception: hybrid-global mode allows resource managers to process transactions in ways that are not normal for URs in global transaction mode. RRS treats a UR that is in hybrid-global transaction mode as a global transaction, and it does not know when a resource manager might treat it as a local transaction.

If the work manager has not specified an environment setting for transaction mode that would apply to a given UR, RRS sets the transaction mode for the UR to hybrid-global mode when the UR state changes to **in-flight**.

Resource managers can be informed of the transaction mode when they express interest in a UR; and, they can retrieve the transaction mode with a call to Retrieve_Side_Information or Retrieve_Side_Information_Fast.

When in hybrid-global transaction mode, an application cannot use the Begin_Transaction service to start either a local or a global transaction.

Compatibility with RMs that process global transactions. To prevent or minimize failures related to its support of local transactions, RRS provides the following:

- A program cannot begin a local transaction or use the Begin_Transaction service unless the Set_Environment service has been used set the transaction mode environment to something other then hybrid-global.
- A resource manager cannot become involved in a local transaction until it has informed RRS, via the Set_Exit_Information service, that it supports local transactions.
- A work manager can prevent the starting of local transactions by using Set_Environment to establish a hybrid-global transaction mode environment and setting the transaction mode environment as protected.
- If a resource manager that does not support local transaction mode expresses an interest in a UR that is in local transaction mode, the service fails with an existing return code, ATR_UR_STATE_ERROR. This return code indicates that the resource manager cannot express interest in the UR, but it is not the resource manager's fault. When receiveing this return code, the resource manager should not unset its exits with RRS.

Ported work manager environments. When porting a work environment, such as a webserver, it is important to minimize the cost of the port. In order to facilitate the porting of work managers from local transaction mode only environments, while maintaining consistency, RRS provides the Set_Environment service. Set_Environment allows a work manager to set a default transaction mode for URs that are started in a specified scope, such as address space or context.

End context. With the Set_Environment service, a work manager can explicitly set a termination option to affect the processing that RRS automatically performs in

the case of normal and abnormal context termination (implicit commit). This capability can be used in any transaction mode.

Restart conditions. RRS never hardens interests that resource managers have expressed in a local UR. Therefore, RRS never returns any local interests when it restarts. Resource managers must use their own recovery logs to ensure the correct processing of resources accessed while in local transaction mode.

Archive logging. RRS does not write local URs to its archive log, but you can display URs in local transaction mode and **in-flight** state, or beyond, through an RRS panel, assuming RRS is active.

Connection techniques. Most z/OS resource managers use connection techniques that are not RRS-enabled. When resource managers use these techniques, any local uncommitted resources are invisible to RRS, and RRS permits global transactions to be started. In addition, the resource manager is not notified of any cleanup efforts of the work managers, such as calls to End_Transaction. Because of these potential problems, do not intermix connection techniques that are not RRS-enabled with those that are.

Global transaction identifiers. A work identifier cannot be set for a local transaction regardless of its state.

Transaction state transitions

Local transaction mode affects UR state transitions. The transaction mode for a given UR is not set until the UR changes to **in-flight**, and, once set, the transaction mode cannot be changed for that UR. The transaction mode of a given UR is determined when the UR state changes from in-reset to in-flight. Table 13 lists the events that can cause this state change, along with the resulting transaction mode. Events which cause RRS to select a transaction mode other then the default mode are highlighted in the table.

Table 13. UR Transaction State Changes (in-reset to in-flight)

Event	Default local transaction mode [1]	Default global transaction mode [1]	Default implicit global transaction mode [1]
Begin_Transaction with MODE = ATR_LOCAL_MODE	local mode	**local mode**	Rejected
Begin_Transaction with MODE = ATR_GLOBAL_MODE	**global mode**	global mode	Rejected
Create_Cascaded_UR (Both parent and child UR)	**global mode**	global mode	hybrid-global mode
Express_UR_Interest (Applies to first expression of interest)	local mode [2]	global mode	hybrid-global mode
Retrieve_UR_Data (only with a states option of ATR_STANDARD_STATES)	**global mode**	global mode	hybrid-global mode
Retrieve_Work_Identifier	**global mode**	global mode	hybrid-global mode
Set_Post_Sync_PET	local mode	global mode	hybrid-global mode

Table 13. UR Transaction State Changes (in-reset to in-flight) (continued)

Event	Default local transaction mode 1	Default global transaction mode 1	Default implicit global transaction mode 1
Notes: **1** The default transaction mode is set by a call to the Set_Environment service. **2** Before it can express interest in a local UR, a resource manager must have called Set_Exit_Information to notify RRS that it can tolerate local transaction mode. If the resource manager has not done this, RRS will reject the Express_UR_Interest service with a return code of ATR_UR_STATE_ERROR.			

Table 14 shows the RRS-related actions that eventually cause an **in-flight** UR to return to **in-reset** state.

Table 14. UR State Changes (in-flight to in-reset) and Transaction Mode

Event	Local transaction mode	Global transaction mode	Hybrid global transaction mode
Explicit end of transaction via the End_Transaction service	**in-reset**	**in-reset**	**in-reset**
Explicit end of transaction via a service other than End_Transaction (for example, Commit_UR or Backout_UR)	Rejected. Not permitted for local URs that are **in-flight** or beyond.	**in-reset**	**in-reset**
The deletion of the last expression of interest in a UR which did not transition from **in-reset** to **in-flight** by a call to the Begin_Transaction service	**in-reset**	No state change	No state change

State transitions for URs in local mode are different from state transitions for URs in global mode. Figure 16 on page 81 shows a state transition diagram for URs in local transaction mode.

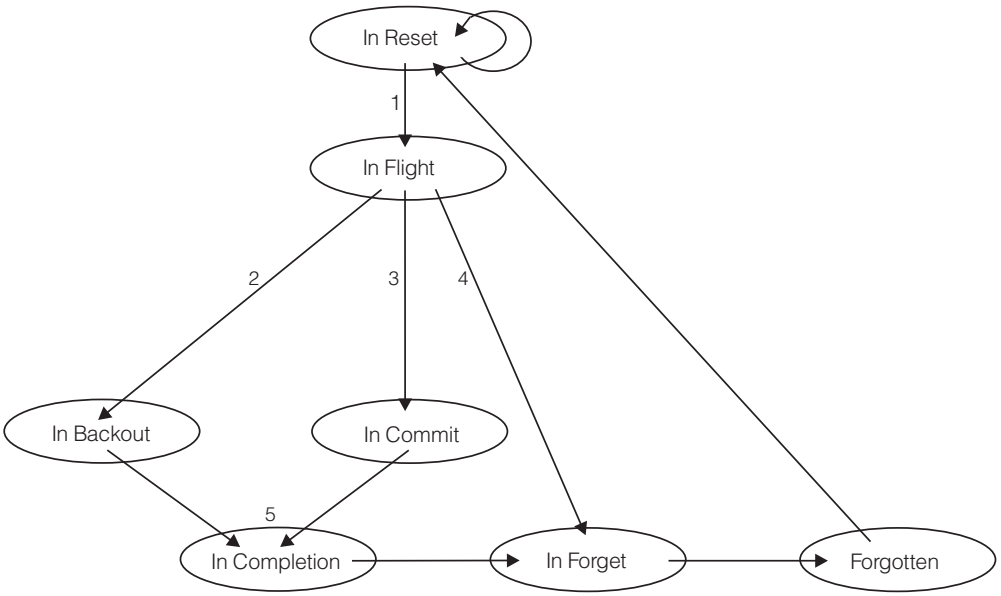

Figure 16. State Transitions for URs in Local Transaction Mode

Unit of work identifiers

Unit of work IDs are persistent tokens used to identify a transaction. They are persistent because they are hardened by resource managers and transaction managers during syncpoint processing. Because they are persistent, they can be used to identify a particular transaction, part of a transaction, or both during normal processing and during restart processing.

Note: Resource managers sometimes use work IDs for other purposes. For example, XIDs are sometimes used as "lock tokens" by data managers. A lock token identifies the owner of a database lock; therefore, any program executing with a given lock token can access data controlled by the token.

RRS supports the following work IDs:

- Unit of recovery identifier (URID)
- Logical unit of work identifier (LUWID)
- Enterprise identifier (EID)
- X/Open identifier (XID)

A URID is a local identifier specific to RRS. It is RRS-specific because it is not defined by any standards body and is only used by RRS and the resource managers that work directly with RRS.

A URID is a local identifier because a URID is associated with a single unit of recovery. A distributed or cascaded transaction with many separate nodes and many URs will have many different URIDs. LUWIDs, EIDs, and XIDs are all global identifiers or a combination of a global identifier and a nonglobal identifier.

A LUWID is defined by the SAA LU 6.2 syncpoint architecture to identify a distributed transaction using the LU 6.2 protocols. It is a global identifier: all of the nodes in a distributed transaction that are part of a nondisjoint LU 6.2 transaction subtree have the same LUWID.

An EID is both a local and a global identifier. The first 4–bytes of an EID are the transaction identifier (TID). The TID is a local identifier: each node in a distributed transaction can have a different TID. The remaining 8–40 bytes of the EID are the global transaction identifier (GTID). The GTID is a global identifier: all of the nodes in a nondisjoint distributed transaction subtree managed by a communication resource manager using EIDs have the same GTID.

An XID is defined by the X/Open XA standard. In addition to the length and FormatID fields, an XID has two important parts: the global transaction identifier (GTRID) and the branch qualifier (BQUAL). The GTRID is a global identifier. All of the nodes in a nondisjoint distributed transaction managed by an X/Open compliant communications manager will have the same GTRID. The BQUAL, however, is not a simple local identifier. Communications resource managers use the BQUAL to denote tightly-coupled and loosely-coupled transaction nodes. Table 15 shows the differences between tightly-coupled and loosely-coupled transaction nodes.

Table 15. Differences Between Tightly-Coupled and Loosely-Coupled Transaction Nodes

Tightly-coupled transaction nodes	Loosely-coupled transaction nodes
• Have exactly the same XIDs, including the BQUAL • Normally share database locks • A communications resource manager **may** send a single set of two-phase commit flows to the entire set of tightly-coupled nodes. **Note:** When RRS manages tightly-coupled nodes, it does not do this.	• Have XIDs with different BQUALs • Do not normally share database locks • A communications resource manager sends separate two-phase commit flows to each node.

Nodes in a distributed transaction connected by a communication resource manager that does not use XIDs are normally loosely-coupled, as are cascaded transactions created by RRS. There are, however, exceptions. For example, APPC/MVS will set the same XID in each UR it creates, but send separate two-phase commit flows to each node, under the following conditions:

- APPC is asked to allocate multiple protected conversations between the same two LUs (source and target), and
- The LUs are part of the same distributed transaction (they are using the same LUWID), and
- The target LU is an alternate scheduler, which causes APPC to create a work context and a UR for the alternate scheduler.

When RRS creates a cascaded transaction, each UR in the cascaded UR family will have an XID. Each UR's XID will have the same FORMATID and GTRID, but by default each will have a different BQUAL. When creating a cascaded UR via a call

to the Express_UR_Interest service, a work manager can override this default behavior and directly control the BQUAL that RRS will use for the UR by specifying an XID on the call.

RRS automatically creates a URID whenever it creates a UR. RRS assigns an XID to a UR whenever the UR becomes part of a cascaded UR family. Otherwise, RRS assigns LUWIDs, EIDs, and XIDs to URs as indicated on either the Set_Work_Identifier service or the Retrieve_Work_Identifier service.

Table 16 lists each identifier, its format, whether RRS can generate it, and how RRS propagates the identifier when creating a cascaded UR.

Table 16. Unit of Work Identifiers

Unit of work identifier (UWID)	**Format**	**Generate via Retrieve_Work_ Identifier service**	**Propagated to a cascaded UR**
LU 6.2 logical unit of work identifier (LUWID)	`netid.luname.instnum.seqnum` **netid.luname** 1-17 character identifier of the network and LU, preceded by a 1-byte fixed length field **instnum** 6-byte fixed TP instance **seqnum** 2-byte fixed sequence number	Allowed.	Not propagated.
Enterprise Identifier (EID)	`tidgtid` **tid** 4-byte transaction identifier (TID) **gtid** 8-40 byte global transaction identifier (GTID)	Not allowed.	Not propagated.
X/Open Identifier (XID)	`FormatIDGtrid_lengthBqual_lengthID` **FormatID** 4-byte fixed format ID **Gtrid_length** 4-byte fixed GTRID length **Bqual_length** 4-byte fixed BQUAL length **ID** 128-byte character XID The GTRID length and BQUAL length define the length of the first and second subsection of the ID. The GTRID must have a length of at least 1 byte, however the BQUAL can have a length of 0. The length of the entire XID cannot exceed 140 bytes.	Required. RRS automatically generates an XID whenever any request for an XID is made against a UR that does not already have one.	The format ID is propagated. GTRID and GTRID length are propagated. BQUAL and BQUAL length are not propagated. BQUAL will be different from all other BQUALs with the same FORMATID and GTRID. This behavior may be overridden by specifying an XID when creating a cascaded UR with the Express_UR_Interest service.

Setting exits with RRS

Your resource manager can provide exit routines to be invoked for events during the two-phase commit protocol for a unit of recovery (UR). Table 17 on page 84 lists the RRS exit routines. A resource manager can cause RRS to bypass the required COMMIT, BACKOUT, and PREPARE exit routines by supplying the return codes for these exits in a call to the Set_Syncpoint_Controls service.

Table 17. Summary of RRS Exit Routines

Exit Number in: Hexadecimal (Decimal) Equate Symbol	Exit routine	Event	Required or optional
1 (1) ATR_STATE_CHECK_ EXIT	STATE_CHECK	An application program called the Commit_UR service or the Application_Commit_UR service, or a server distributed syncpoint resource manager (SDSRM) called the Prepare_Agent_UR service.	Optional. If not provided, RRS assumes the state of the resource manager's resources is correct for all commit requests. See "STATE_CHECK exit routine" on page 130.
2 (2) ATR_PREPARE_EXIT	PREPARE	The UR entered the **in-prepare** state.	Required. If the Set_Syncpoint_Controls service provides a return code for the PREPARE exit, RRS bypasses the exit for that expression of interest only. See "PREPARE exit routine" on page 126.
3 (3) ATR_DISTRIBUTED_ SYNCPOINT_EXIT	DISTRIBUTED_ SYNCPOINT	In distributed processing of a UR, the return codes from the PREPARE exit routines include at least one OK, no backout, and no heuristic-mixed effect. The UR state is **in-doubt**. Note that your resource manager must call the Set_Syncpoint_Controls service to enable the DISTRIBUTED_ SYNCPOINT exit routine.	Optional. A resource manager that takes the distributed syncpoint manager (DSRM) role provides this exit routine. If it is not provided, RRS coordinates the commit processing and rejects any resource manager call to the Set_Syncpoint_Controls service that specifies the DSRM role. See "DISTRIBUTED_SYNCPOINT exit routine" on page 113.
4 (4) ATR_COMMIT_EXIT	COMMIT	The UR entered the **in-commit** state, or a server distributed syncpoint resource manager (SDSRM) called the Commit_Agent_UR service.	Required. If the Set_Syncpoint_Controls service provides a return code for the COMMIT exit, RRS bypasses the exit for that expression of interest only. See "COMMIT exit routine" on page 108.
5 (5) ATR_BACKOUT_EXIT	BACKOUT	The UR entered the **in-backout** state, or a server distributed syncpoint resource manager (SDSRM) called the Backout_Agent_UR service.	Required. If the Set_Syncpoint_Controls service provides a return code for the BACKOUT exit, RRS bypasses the exit for that expression of interest only. See "BACKOUT exit routine" on page 104.

Table 17. Summary of RRS Exit Routines (continued)

Exit Number in: Hexadecimal (Decimal) Equate Symbol	Exit routine	Event	Required or optional
6 (6) ATR_END_UR_EXIT	END_UR	The UR entered the **in-end** state.	Optional. If not provided, RRS considers the resource manager's interest in the UR complete when its COMMIT or BACKOUT exit routine completes, unless either of the following is true: • The resource manager has an enabled COMPLETION exit routine. • The resource manager is an SDSRM; RRS must wait for the resource manager to call Forget_Agent_UR_Interest before RRS can complete UR processing. See "END_UR exit routine" on page 116.
7 (7) ATR_EXIT_FAILED_ EXIT	EXIT_FAILED	An RRS exit routine failed.	Required. See "EXIT_FAILED exit routine" on page 118.
8 (8) ATR_COMPLETION_ EXIT	COMPLETION	A UR being processed across distributed systems completed, and a call to the Set_Side_Information service specified ATR_DRIVE_ COMPLETION.	Optional. If not provided, RRS assumes the resource manager requires no actions when distributed UR processing is completed, unless the following is true: • The resource manager is an SDSRM; RRS must wait for the resource manager to call Forget_Agent_UR_Interest before RRS can complete UR processing. See "COMPLETION exit routine" on page 111.
9 (9) ATR_ONLY_AGENT_ EXIT	ONLY_AGENT	Only one resource manager expressed only one interest in the UR, and the application requested commit. (If the application requested backout, RRS drives the BACKOUT exit routine.)	Optional. If not provided, RRS invokes the resource manager's PREPARE exit routine and continues with the standard two-phase commit protocol. See "ONLY_AGENT exit routine" on page 122.

Table 17. Summary of RRS Exit Routines (continued)

Exit Number in: Hexadecimal (Decimal) Equate Symbol	Exit routine	Event	Required or optional
10 (10) ATR_SUBORDINATE_ FAILED_EXIT	SUBORDINATE_ FAILED	FEither RRS or a resource manager on a subordinate system failed, the subordinate system itself terminated, or the context associated with the subordinate UR abnormally terminated.	Optional. See “SUBORDINATE_FAILED exit routine” on page 133.
B (11) ATR_ PRE-PREPARE_EXIT	PRE_PREPARE	An application program called: • Commit_ UR (ATRCMIT, ATR4CMIT) service • Application_Commit_UR (SRRCMIT) service • End_Transaction (ATREND, ATR4END) service • End_Context (CTXENDC, CTX4ENDC) service • Server distributed syncpoint resource manager (SDSRM) called the Prepare_Agent_UR (ATRAPRP, ATR4APRP) service • Delegate_Commit_ Agent_UR (ATRADCT, ATRADCT1, ATR4ADCT) service The UR stays in-flight state.	Optional. If not provided, RRS will proceed to process the optional State_Check exit. See “PRE_PREPARE exit routine” on page 125 for more information.

Example of resource manager processing

This section gives pseudo-code examples of resource manager processing in the following topics:

The examples assume SRB exit routines; see Chapter 4, “Using resource recovery services,” on page 51 for information about SRB routines.

These examples indicate possible processing, but the processing can be done in other ways. For instance, in the first example on restart processing:

- Commit and backout processing are performed inline. The resource manager could specify ATR_RESPOND_CONTINUE in the Respond_to_Retrieved_Interest call and perform the commit and backout processing in the COMMIT and BACKOUT exit routines.
- The resource manager responds to all retrieved URs after it begins to process new URs. A resource manager can, however, respond to all retrieved interests before calling the End_Restart service.

Also, the examples emphasize processing related to RRS; processing related to database management is very high level.

Restart processing

The following example indicates, in pseudo-code, the restart processing by a resource manager.

```
Start processing

Call to Register_Resource_Manager service

Call to Set_Exit_Information service for context services exit routines

Call to Set_Exit_Information service for RRS exit routines

Call to Retrieve_Log_Name service

   If call returns ATR_RM_LOGNAME_NOT_SET

      Cold start

      Call to Set_Log_Name service

   Else

      Match resource manager log name and RRS log token
      to previous name and token

      If match

         Proceed with restart

      Else

         Fail resource manager restart

Perform resource manager log processing, if needed

Read resource manager logs and build resource manager structures

Perform media recovery, if needed

Call to Begin_Restart service

Do until return code is ATR_NO_MORE_INCOMPLETE_INTERESTS

   Call Retrieve_UR_Interest service to obtain incomplete URs

End do

Call to End_Restart service

Do until no records remain
```

```
If UR is not in resource manager logs

   Call Respond_to_Retrieved_Interest service,
   specifying ATR_RESPOND_COMPLETE

If UR state is in-backout

   Do UNDO action to backout UR

      Read UNDO records from the resource manager log in LIFO order

      Do until no UNDO records remain

         Write a compensating REDO record

         Rewrite the database record

      End do

   End do

   Call Respond_to_Retrieved_Interest service,
   specifying ATR_RESPOND_COMPLETE

If UR state is in-commit

   Do REDO action to commit UR

      Read REDO records from the resource manager log in FIFO order

      Do until no REDO records remain

         Write the database record

      End do

   End do

   Call Respond_to_Retrieved_Interest service,
   specifying ATR_RESPOND_COMPLETE

If UR state is in-doubt

   If role is distributed syncpoint resource manager

      Do TODO action to resolve UR

      End do

   If role is participant
      Do TODO action to resolve UR

         Read REDO records from the resource manager log in FIFO order

         Do until no REDO records remain

            Apply database changes

         End do

      End do

   Call Respond_to_Retrieved_Interest service,
   specifying ATR_RESPOND_CONTINUE

End do
```

```
Do until no resource manager log records remain

   If UR not known by RRS, do action indicated by resource
   manager log and do not inform RRS of result

End do

Enable connections to application program

Begin processing new URs
```

UR processing

The following example indicates, in pseudo-code, the processing of a UR by a resource manager. For the example, the resource manager has requested a presumed abort protocol, the Application_Commit_UR service was called, and the resource manager needs to maintain an interest in the context.

```
Respond to application program connection request

Receive read request from application

Call Express_Context_Interest service

Call Express_UR_Interest service to express an unprotected interest in UR

Provide requested data to the application

Receive write request from application to change data

Call Change_Interest_Type service to change UR interest from
unprotected to protected

Write recovery data in the resource manager log

   Write UNDO record

   Write REDO record

< RRS receives commit UR call from application to commit the change >

< RRS might invoke the STATE_CHECK exit routine, especially for
  distributed processing >

< RRS invokes the PREPARE exit routine >

< Depending on the PREPARE votes, RRS invokes the COMMIT or
  BACKOUT exit routine >

< RRS invokes the END_UR exit routine >
```

PREPARE exit routine

The following example indicates, in pseudo-code, the processing a PREPARE exit routine might perform. There are other possibilities; some resource managers, for example, might hold the database locks until the COMMIT or BACKOUT exit gets control.

```
If request is read

   Release the database locks

   Set exit routine return code to ATRX_FORGET

   Return to address in general purpose register 14
```

```
Else

   Harden UNDO record to external storage

   Harden REDO record to external storage

   Write an in-doubt record to the resource manager log

   Set exit routine return code to ATRX_OK

   Return to address in general purpose register 14
```

COMMIT exit routine

The following example indicates, in pseudo-code, the processing of a COMMIT exit routine. In the parameter list RRS passes to the exit routine, the nonpersistent interest data points to resource manager structures that represent the resources being changed.

```
Write the database record or records to permanently change the
database

Release the database locks

Write a successful commit record to the resource manager log

Set exit routine return code to ATRX_OK

Return to address in general purpose register 14
```

BACKOUT exit routine

The following example indicates, in pseudo-code, the processing of a BACKOUT exit routine. In the parameter list RRS passes to the exit routine, the nonpersistent interest data points to resource manager structures that represent the resources the application had intended to change.

```
Rewrite the database record or records

Release the database locks

Set exit routine return code to ATRX_OK

Return to address in general purpose register 14
```

DISTRIBUTED_SYNCPOINT exit routine

The following example indicates, in pseudo-code, the processing of a DISTRIBUTED_SYNCPOINT exit routine.

```
Send a REQUEST_COMMIT message to the conversational partner

Wait for the return message

If the partner sent a COMMIT message

   Set exit routine return code to ATRX_OK

If the partner sent a BACKOUT message

   Set exit routine return code to ATRX_BACKOUT

Return to address in general purpose register 14
```

Resource recovery exit routines

This section describes, under "Programming considerations," information that is common to all RRS exit routines, followed by a description of each exit routine.

Programming considerations

The following topics describe installing, invoking, processing, and returning for an exit routine and the action taken on an exit routine failure.

Installing an exit routine

To install the RRS exit routines, the resource manager must:

- Call the Register_Resource_Manager service.
- Set the RRS exit routines through one or more calls to the Set_Exit_Information service. If the resource manager issues more than one call for RRS exit routines, the first call must specify all the required exit routines.

If your resource manager will need RRS to generate logical unit of work identifiers (LUWIDs), you must specify the LU name RRS is to use. You specify this name through the *variable_data_1* parameter on the Set_Exit_Information service.

Specifying RRS specific Set_Exit_Information return codes

Set_Exit_Information return codes related to its processing; see "Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF)" on page 149. The service might also return codes from their exit manager.

The following table lists the return codes you might get from RRS when you call the Set_Exit_Information service to set the RRS exit routines.

Return Code in: Hexadecimal (Decimal) Equate Symbol	**Meaning and action**
8000 (32768) ATR_EXIT_PREPARE_NOT_ SPECIFIED	**Meaning**: The call did not specify the required PREPARE exit. The system rejects the service request. **Action**: Check the calling program for a probable coding error.
8001 (32769) ATR_EXIT_COMMIT_NOT_ SPECIFIED	**Meaning**: The call did not specify the required COMMIT exit. The system rejects the service request. **Action**: Check the calling program for a probable coding error.
8002 (32770) ATR_EXIT_BACKOUT_NOT_ SPECIFIED	**Meaning**: The call did not specify the required BACKOUT exit. The system rejects the service request. **Action**: Check the calling program for a probable coding error.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
8003 (32771) ATR_EXIT_EXIT_FAILED_NOT_ SPECIFIED	**Meaning**: The call did not specify the required EXIT-FAILED exit. The system rejects the service request. **Action**: Check the calling program for a probable coding error.
8004 (32772) ATR_RM_ACTIVE_ON_ ANOTHER_SYSTEM	**Meaning**: The resource manager named in the call currently has exits set on another system in the sysplex. The system rejects the service request. **Action**: Check the calling program for a probable coding error.
8005 (32773) ATR_RM_NEW_KEY_INV	**Meaning**: The key of the resource manager is different from the key RRS expects. This difference could occur if the resource manager registered again with a different key while RRS remains active. The system rejects the service request. **Action**: Check the calling program for a probable coding error.
8006 (32774) ATR_SEIF_PARM_NOT_ADDR	**Meaning**: The caller passed parameters to RRS that were not addressable. The key of the data is different from the key RRS expects. The system rejects the service request. **Action**: Check the calling program for a probable coding error.
8007 (32775) ATR_EM_WRONG_STATE	**Meaning**: The calling resource manager tried to set its exit routines during unset processing for its exit routines. The system rejects the service request. **Action**: Check the calling program for a probable coding error.
8008 (32776) ATR_RM_WRONG_STATE	**Meaning**: The calling resource manager tried to set its exit routines while not in **Unset**, **Reset**, or **Set** state. The system rejects the service request. **Action**: Check the calling program for a probable coding error.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
800A ATR_RM_METADATA_UNSUPPORTED	**Meaning**: The caller requested RM Metadata support, but the system does not have RM Metadata support active. RRS was not able to connect to the RM Metadata log stream. The system rejects the service request. **Action**: • Define the RM Metadata log stream and retry the service request. • Retry the service request without setting the ATR_8K_RM_METADATA_REQUESTED flag. • Restart the resource manager on a system that can connect to the RM Metadata log stream.

Specifying RRS specific RM UNSET reason codes

CRG_RM_EXITS_UNSET reason codes related to its processing; see the section named value2 in "Parameters" on page 26.

The following table lists the reason codes you might get from RRS when exits were unset.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
8009 (32777) ATR_EM_UNAVAILABLE	**Meaning**: The resource manager has been `Unset` because RRS was terminated through the SETRRS SHUTDOWN command. **Action**: Wait for RRS to become active and `Reset` the resource manager's exit with RRS.

Invoking an exit routine

The system invokes an RRS exit routine when an event occurs. Before an exit routine can be invoked, the resource manager must:

- Express interest in a UR through a call to the Express_UR_Interest service
- For a restart UR obtained by a call to the Retrieve_UR_Interest service, specify a response code of ATR_RESPOND_CONTINUE on each call to Respond_to_Retrieved_Interest, then call the End_Restart service

If there are multiple interests in a UR, either through one resource manager or multiple resource managers, the order in which RRS invokes the exit routines for a given event is as follows:

1. All SRB type exit routines, either one after another or all at once asynchronously

2. All PC type exit routines, in the order in which the resource managers first expressed interest in the UR. If any resource manager has multiple expressions of interest in the UR, RRS invokes the exit routines for those interests one after another in the order in which the resource manager expressed the interest.

The *exit_type* parameter in the call to the Set_Exit_Information service specifies how RRS is to invoke the exit routine:

- **SRB routine**: The system schedules a service request block (SRB) at local priority in the resource manager's address space to give control to the exit routine.

  The exit routine may run synchronously or asynchronously. In either case, it is nonpreemptable. Because RRS might, under certain conditions, wait for one SRB routine to complete before driving another, it is not a good idea to suspend exit routine processing.

  A resource manager in a swappable address space must use SRB exit routines.
- **PC routine**: The system issues a stacking Program Call (PC) instruction to give control to the exit routine. The stacking PC must either use a system LX so that the routine is available from all address spaces or establish a connection in the RRS address space to the PC table, as described in "Establishing a connection to the PC table" on page 95.

  Note: Consider carefully before deciding to use a system LX. Using a system LX improperly can prevent ASIDs from being reused, which can in turn cause unscheduled IPLs. To avoid unnecessary loss of ASIDs, IBM recommends that a resource manager use a non-system LX or an SRB. See Reusing ASIDs in *z/OS MVS Programming: Extended Addressability Guide* for more detail.

  The exit routine will run synchronously; therefore, the resource manager must not suspend processing of the work unit. The system cannot invoke exit routines for other interests in the UR until the PC routine completes.

  The resource manager must be in a nonswappable address space to use PC exit routines. A PC exit routine must remain available to the system until the resource manager ends processing, unregisters, or issues a call to the set exit routine service to change the exit routine.

  It is also important to note that a PC exit routine and any routine that it invokes cannot issue any SVC instruction.

  When RRS invokes a PC type exit routine, the way the PC is defined in its entry table determines the cross memory environment in effect when it gets control:

  – If a nonspace switching PC is used, the primary address space and the secondary address space will be the RRS address space.
  – If a space switching PC is used, depending on the PC definition, the RRS address space will either be the secondary address space or not be part of the environment at all.

  If RRS is the primary or secondary address space and exit routine recovery is defined as MODE=FULLXM, the exit routine will not receive control if the RRS address space is cancelled. If RRS is the primary address space and exit routine recovery is defined as MODE=PRIMARY, the exit routine will not receive control if the RRS address space is cancelled.

The advantage of the PC routine over an SRB routine is a shorter path length. Invocation of an SRB routine has the overhead of scheduling and dispatching an SRB. Choose the type of exit routine as follows:

- Choose an SRB routine if the exit routine's processing is longer. The routine can perform the processing asynchronously and provide the return code later.

- Choose a PC routine if the exit routine's processing is short, without I/O or implicit waits by services called. Because the routine runs synchronously, the return code indicates that its processing is complete.

If you need to suspend the processing of either a PC routine or an SRB routine, set an ATRX_LATER return code and return to RRS, then call the Post_Deferred_UR_Exit service when processing is complete.

Establishing a connection to the PC table: To use a nonsystem LX, a resource manager must issue the ETCON macro to connect the RRS address space to its own LX. The resource manager must issue ETCON before it calls the Set_Exit_Information service. For information about using ETCON, see *z/OS MVS Programming: Extended Addressability Guide*.

So that the resource manager has the information it needs to issue ETCON, RRS provides the STOKEN of its address space through a nonpersistent system level name/token pair. The name is defined through ATR_RRS_STOKEN_NAME in ATRRASM and ATRRC. For information about using name/token pairs, see *z/OS MVS Programming: Assembler Services Guide* and *z/OS MVS Programming: Authorized Assembler Services Guide*.

Figure 17 on page 96 shows a sample of Assembler language code to obtain the STOKEN of the RRS address space. Note that the sample in Figure 17 on page 96 does not include the entry and exit linkage.

```
*********************************************************************
*  Initialize the NAME field
*********************************************************************
         MVC   NAME,ATR_RRS_STOKEN_NAME Get RRS STOKEN Name
*********************************************************************
*  Attempt to retrieve the STOKEN
*********************************************************************
         LOAD  EP=IEANTRT              Get address of IEANTRT routine
         LR    R15,R0                  Set address for Call
         CALL  (15),(LEVEL,NAME,TOKEN,RETCODE)
*
         LA    R15,IEANT_OK            Get successful return code value
         C     R15,RETCODE             Was TOKEN Returned?
         BNE   RRSDOWN                 No, RRS must not be available
         EJECT
*********************************************************************
*  Save RRS' STOKEN
*********************************************************************
         LA    R2,TOKEN                Set pointer to TOKEN area
         USING ATR_STKN_TOKEN,R2       Set addressability -
*                                      ATR_STKN_TOKEN area maps
*                                      the returned TOKEN
         MVC   SAVESTKN,ATR_RRS_STOKEN Move RRS STOKEN to return
*                                      area
         DROP  R2                      Free up register 2
*********************************************************************
* Do processing with the STOKEN as necessary here
*********************************************************************
*
         B     CONT
         EJECT
*********************************************************************
* Do processing when STOKEN unavailable here
*********************************************************************
RRSDOWN  DS    0H
*
         B     CONT
         EJECT
CONT                                   Return to caller
*********************************************************************
* Local working storage declares
*********************************************************************
NAME     DS    CL16                    Name for Name/Token pair
TOKEN    DS    XL16                    Token for Name/Token Pair
RETCODE  DS    F
SAVESTKN DS    CL8                     RRS STOKEN
*********************************************************************
* Constant and Equates
*********************************************************************
LEVEL    DC    A(IEANT_SYSTEM_LEVEL)   SYSTEM LEVEL
R0       EQU   0
R2       EQU   2
R14      EQU   14
R15      EQU   15
         EJECT
*********************************************************************
* RRS Constants
*********************************************************************
         ATRRASM
         EJECT
*********************************************************************
* NAME/TOKEN VARIABLE DECLARES
*********************************************************************
         IEANTASM
```

Figure 17. Obtaining the STOKEN

Processing by an exit routine

A resource manager can have an exit routine for each RRS exit or a single routine for all RRS exits. At invocation, all RRS exit routines receive a parameter list in the same format but with exit-specific meanings for some parameters. If a resource manager uses a single exit routine, the routine can identify the processing needed based on the exit number parameter.

When an exit routine receives control, the routine can perform the expected processing. Alternatively, at any time in its processing, the exit routine can postpone its processing, taking the following steps:

- Log the exit notification
- Request scheduling of a task that can perform the processing asynchronously
- Pass a return code of ATRX_LATER or ATRX_LATER_CONTINUE back to RRS.

Later, when the exit routine completes processing, the resource manager can provide its return code to RRS through a call to the Post_Deferred_UR_Exit service.

Returning from an exit routine

An exit routine returns to RRS as follows:

- An SRB routine must return to the address that was in register 14 on entry to the routine.
- A PC routine must return with a Program Return (PR) instruction.

If an exit routine returns ATRX_FORGET, the system does not invoke any subsequent RRS exit routines for the current interest in the UR. RRS deletes the interest in the UR.

Action if an exit routine fails: If an exit routine abends or returns an unexpected return code, the system gives control to the EXIT_FAILED exit routine.

Action if exit routines are unset: If a resource manager's exit routines are unset for any reason:

- RRS will quiesce any SRB exit routines, but PC exit routines continue to run.
- Context services will not quiesce SRB or PC exit routines. They continue to run.

A quiesced exit routine completes normally or abnormally, then returns to the caller.

If an exit routine that continues to run requests an RRS service, it will get an error return code from the service.

A resource manager's failure causes its exit routines to become unset with the exit managers. A resource manager's exit routines can also be unset if its EXIT_FAILED exit routine fails. If the exit routines are unset, RRS takes the actions shown earlier in Table 4 on page 52 for an incomplete interest in an active UR.

Overlapping of exit routine processing

While a resource manager's exit routine is running, exit routines for other interests in the UR are also active. *Active* means they are running or have returned an ATRX_LATER or ATRX_LATER_CONTINUE return code. (If a routine returns ATRX_LATER or ATRX_LATER_CONTINUE, it is considered active until the resource manager provides its final return code through a call to the Post_Deferred_UR_Exit service.)

Table 18 indicates potential overlaps of exit routine processing. The table columns indicate other exit routines that may be active when the exit routine of the row is active.

Table 18. Exit Routine Processing Overlap

Active Exit	STATE_ CHECK	PRE- PARE	DIST_ SYNCPT	COMMIT	BACK- OUT	END_ UR	ONLY_ AGENT	COMPLE- TION	EXIT_ FAILED	SUBOR- DINATE_ FAILED
STATE_ CHECK	Y	N	N	N	N	N	N	N	O	N
PREPARE	C	Y	N	N	N	N	N	N	O	N
DIST_ SYNCPT	C	C	N	N	N	N	N	N	O	N
COMMIT	C	C	C	Y	N	N	N	N	O	N
BACKOUT	C	C	C	N	Y	N	N	N	O	N
END_UR	C	C	C	C	C	Y	N	N	O	N
ONLY_ AGENT	C	N	N	N	N	N	N	N	O	N
COMPLE- TION	C	C	C	C	C	C	N	Y	O	N
EXIT_ FAILED	O	O	O	O	O	O	O	O	O	N
SUBOR- DINATE_ FAILED	N	N	N	N	N	N	N	N	N	Y

Note: The table symbols are:

C The column exit routine has completed for this UR.

N No, the column exit routine is not invoked for this UR when the row exit routine is active.

Y Yes, the column exit routine may be concurrently active.

O Yes, the column exit routine may be concurrently active but only when the following are true:

- On the Set_Exit_Information service, the resource manager used *variable_data_2* to set ATR_EF_ON_LATER_WITH_SYNC.
- The row exit routine returned either ATRX_LATER or ATR_LATER_CONTINUE to RRS

Additional details are:

- RRS does not overlap exit routines for the same interest in a UR.
- **STATE_CHECK exit routine**: If a STATE_CHECK exit routine returns an ATRX_REDRIVE code, RRS invokes all STATE_CHECK exit routines again, but not until all the originally invoked STATE_CHECK exit routines complete.
- **PREPARE exit routine**: If one PREPARE exit routine votes NO, requesting backout of the UR, RRS stops invoking PREPARE exit routines. When all PREPARE exit routines that were invoked are complete, RRS starts to invoke BACKOUT exit routines.
- **DISTRIBUTED_SYNCPOINT exit routine**: RRS does not invoke other exit routines for a UR if the DISTRIBUTED_SYNCPOINT exit routine has not completed, **with two exceptions**:
  1. When the installation uses a panel to resolve an **in-doubt** UR
  2. When a cancelled application leaves an **in-doubt** UR and the installation responds COMMIT or BACKOUT to the message that identifies the **in-doubt** UR

  In either case, RRS starts invoking COMMIT, BACKOUT, END_UR, or COMPLETION exit routines even though the DISTRIBUTED_SYNCPOINT exit

routine has not completed. If the DISTRIBUTED_SYNCPOINT exit routine subsequently passes back a return code, RRS ignores the code. If, however, the DISTRIBUTED_SYNCPOINT exit routine is an SRB routine, RRS purges the exit routine before it drives any other exits.

Environment

Before the exit routine receives control, RRS establishes a function recovery routine (specifically, an EUT FRR) for error recovery. Because RRS has already established an EUT FRR when the exit routine receives control, the exit routine cannot establish an ESTAE-type recovery environment. Do not schedule an IRB to the task under which syncpoint processing was initiated. RRS may run on that task (with its FRR), waiting for your exit to complete.

An SRB exit routine receives control in the following environment:

Minimum authorization:	Key of the resource manager when it registered, supervisor state
Dispatchable unit mode:	SRB
Cross memory mode:	PASN = HASN = SASN, home address space of the resource manager when it registered
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

A PC exit routine receives control in the following environment:

Minimum authorization:	Determined by the PC instruction characteristics, supervisor state
Dispatchable unit mode:	Task
Cross memory mode:	Determined by the PC instruction characteristics, home address space unpredictable
AMODE:	Determined by the PC instruction characteristics
ASC mode:	Determined by the PC instruction characteristics
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming requirements: The high level language (HLL) definitions for the exit routine parameter list are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations

Entry to an exit routine: The exit routine receives information in the registers and a parameter list.

Registers at entry: When an SRB exit routine receives control, the GPRs contain:

Register	Contents
0	Not applicable

1	Address of the parameter list for the exit routine
2-12	Not applicable
13	Address of a 72-byte save area
14	Return address
15	Address of the exit routine's entry point

When an SRB exit routine receives control, the ARs contain:

Register	**Contents**
0-15	Not applicable

When a PC exit routine receives control, the GPRs contain:

Register	**Contents**
0	Not applicable
1	Address of the parameter list for the exit routine
2-15	Not applicable

When a PC exit routine receives control, the ARs contain:

Register	**Contents**
0-15	Not applicable

Parameter list

The parameter list is the same for all RRS exit routines.

The parameter list consists of pointers to fields that contain the values. If a parameter is not meaningful for the exit routine being invoked, the field contains binary zeros. All parameters, except *return_code*, are input to the exit routine. Access to the parameters is controlled by storage protect key:

- **Input parameters**: For the parameters received by the exit routine, the resource manager and exit routine have READ access, but might not have WRITE access.
- **Output parameters**: For the parameters returned by the exit routine, the resource manager and exit routine have READ and WRITE access.

Syntax:

```
(return_code
,version
,exit_number
,resource_manager_token
,exit_manager_name
,resource_manager_global_data
,ur_interest_token
,nonpersistent_interest_data
,exit_flags
,value1
,value2
,value3
,value4
,value5)
```

Parameters:

return_code
Points to a field that, upon return from the exit routine, is to contain a hexadecimal return code. Define the field as a 4-byte integer.

The return codes have unique meanings for each exit routine. See the following exit routine descriptions for the return codes.

version
Points to a field that contains the version of the RRS interface. The current version is 1. Define the field as a 4-byte integer.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer.

See the following exit routine descriptions for the values. If a single exit routine is used for multiple exits, the routine can use this number to branch to the correct action logic.

resource_manager_token
Points to a field that contains the resource manager token. Define the field as a 16-byte character string. Your resource manager received the token from the register resource manager service.

exit_manager_name
Points to a field that contains the name of the resource manager that is functioning as the exit manager. Define the field as a 16-byte character string. The exit manager for this exit routine is RRS; its exit manager name is:

```
ATR.EXITMGR.IBM
```

resource_manager_global_data
Points to a field that contains the resource manager global data. Define the field as a 16-byte character string. Your resource manager provided this data in the call to the Register_Resource_Manager service.

For the exit routine, this data should be an anchor or anchors for data structures in the resource manager.

ur_interest_token
Points to a field that contains the UR interest token for the interest for which

the system is invoking the exit routine. Define the field as a 16-byte character string. Your resource manager received the token from the Express_UR_Interest service or the Retrieve_UR_Interest service.

nonpersistent_interest_data
Points to a field that contains the nonpersistent interest data. Define the field as a 16-byte character string. Your resource manager provided this data in a call to the service: express UR interest, process interest, or retain interest.

exit_flags
Points to a field that contains flags for the exit routine. Define the field as a 4-byte integer. See the following exit routine descriptions for the flags that are meaningful for each routine. If a restart occurs, RRS preserves the settings for bits 2-8 as they were at the last time data was logged before the restart.

The bits in the field mean the following:

Table 19. Bits in the exit_flags field

Bit	Bit Mask, in Hex	Meaning, if Bit is On
0	X'80000000'	**Name:** ATRXFLAGRESTARTINTEREST **Meaning:** The UR interest being processed is for a restart expression of interest.
1	X'40000000'	**Name:** ATRXFLAGTERMINATING SYNCPOINT **Meaning:** The context is ending. Therefore RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
2	X'20000000'	**Name:** ATRXFLAGRESOLVEDBYINSTALLATION **Meaning:** The installation used an RRS panel to commit or back out the UR, which had been in an **in-doubt** state.
3	X'10000000'	**Name:** ATRXFLAGHEURISTICMIXED **Meaning:** RRS detected a heuristic-mixed condition for this UR.
4	X'08000000'	**Name:** ATRXFLAGRESYNCINPROGRESS **Meaning:** RRS detected a resync in progress for the UR.
5	X'04000000'	**Name:** ATRXFLAGPREPARERESULTFORGET **Meaning:** For the UR, the collected prepare vote was FORGET, meaning that the prepare votes were ABSTAIN or FORGET. The vote for this interest in the UR was ABSTAIN.
6	X'02000000'	**Name:** ATRXFLAGIMMEDIATEBACKOUT **Meaning:** The backout operation occurred because the application, either explicitly or implicitly, requested backout, not because a resource manager could not commit its resources.

Table 19. Bits in the exit_flags field (continued)

Bit	Bit Mask, in Hex	Meaning, if Bit is On
7	X'01000000'	**Name:** ATRXFLAGREDRIVELIMIT **Meaning:** The STATE_CHECK redrive limit has been reached for this UR; thus, the exit routine cannot issue the ATRX_REDRIVE return code.
8	X'00800000'	**Name:** ATRXFLAGCOMMIT **Meaning:** The overall vote for the UR is commit.
9	X'00400000'	**Name:** ATRXFLAGAPPLICATIONASYNCABEND **Meaning:** The application encountered an asynchronous abend.
10	X'00200000'	**Name:** ATRXFLAGRETAININTINV **Meaning:** The ATRSROI callable service should not be called from this exit routine.
11	X'00100000'	**Name:** ATRXFLAGCASCADEDUR **Meaning:** The UR being processed is a cascaded UR.
12-13	X'00'	**Name:** ATRXFLAGHYRBIDGLOBALMODE **Meaning:** The UR being processed is in hybrid-global transaction mode.
	X'01'	**Name:** ATRXFLAGGLOBALMODE **Meaning:** The UR being processed is in global transaction mode.
	X'10'	**Name:** ATRXFLAGLOCALMODE **Meaning:** The UR being processed is in local transaction mode.
14	X'00020000'	**Name:** ATRXFlagTerminatingSP_TERM **Meaning:** Context is ending (flag ATRXFLAGTERMINATING SYNCPOINT is ON) and the request is from the termination of the Task or Address Space.
15-31		Reserved for future use.

value1
value2
value3
value4
value5

Point to fields that contain values unique for the exit routines. Define each field as a 4-byte integer. If a value is not used for an exit routine, its field contains binary zeros.

See the following exit routine descriptions for the values.

Exit from an exit routine

The exit routine provides information to the system in the return code in the parameter list.

Registers at Exit: When an SRB exit routine returns control, the GPRs must contain:

Register	Contents
0-1	Not applicable
2-13	Restored to contents upon entry
14-15	Not applicable

When an SRB exit routine returns control, the ARs must contain:

Register	Contents
0-1	Not applicable
2-13	Restored to contents upon entry
14-15	Not applicable

When a PC exit routine returns control, the GPRs contain:

Register	Contents
0-15	Not applicable

When a PC exit routine returns control, the ARs contain:

Register	Contents
0-15	Not applicable

BACKOUT exit routine

The BACKOUT exit routine receives control when the UR state is **in-backout**. RRS invokes BACKOUT exit routines when:

- The application calls a service to back out the UR.
- The application calls a service to commit the UR, but a resource manager voted BACKOUT in its PREPARE exit routine.
- A resource manager that has taken the SDSRM role calls the Backout_Agent_UR service.
- An application or a work manager calls the End_Transaction service.

Note: The resource manager can specify in a call to the Set_Syncpoint_Controls service the return code for the BACKOUT exit routine. In this case, RRS does not invoke the routine.

Processing

The BACKOUT exit routine should back out the UR by **not** making the changes in the resources for this interest in the UR.

Defer exit processing

Return code, ATRX_DEFER, is provided to allow a resource manager to request RRS to defer the backout processing for a particular interest. Upon receiving the ATRX_DEFER return code, RRS will requeue the interest to be processed later for syncpoint processing. The Backout exit can only be deferred once for any one expression of interest. RRS will invoke the resource manager's Exit_Failed exit if the resource manager attempts to defer the Backout exit twice for any one of its interests.

- **Syncpoint processing:**

  When the syncpoint processing is first initiated, RRS builds a syncpoint queue in the order in which the resource manager expressed its interests in the UR. There could be multiple resource managers with multiple interests on the queue. When a resource manager defers the backout processing for one interest, this particular interest will be moved to the end of the syncpoint queue. The exit ordering is thus disrupted and no longer preserved for this resource manager. RRS is not sensitive if the Backout exit is driven in the correct order. RRS will continue to drive the backout exit for all the outstanding interests on the syncpoint queue. The resource manager is in control of the Backout exit ordering by returning the proper exit return code.

- **Cascaded transactions:**

  If a resource manager has an interest in multiple URs in a locally cascaded transaction, the exit invocation for the RM's interest will be deferred after all the child UR interests have been handled as opposed to after the current UR's interests have been processed. Deferring and rescheduling the backout exit is not enabled for multisystem cascaded transactions.

- **Exit failed processing:**

  The Exit_Failed exit receives control when one of the RRS exit routines fails. RRS gives this routine the exit number of the failed routine and the reason why the routine failed. RRS, at most, invokes the Exit_Failed once for each UR state for any one expression of interest. When the Exit_Failed exit returns control to RRS, the routine must provide a return code for the failed exit. If the failed exit is the Backout exit and it has been deferred before, the resource manager cannot defer it again. An attempt to do so will cause RRS to unset the resource manager's RRS exit routines. If the Backout exit has not been deferred before, the resource manager can return a ATRX_DEFER return code to request RRS to drive the exit later.

  As a part of defer backout exit processing the Exit_Failed exit could be driven for the following reasons:

  – A resource manager returns a "defer" return code more than once for any one of its expression of interest
  – A resource manager defers the backout processing for all of its expressions of interest. (The Exit_Failed exit will be driven for the last interest the resource manager expressed in the UR.)
  – A "defer" return code is returned after the resource manager had previously returned an OK return code for one of its expression of interest (an OK return code indicates the backout processing is completed for the "backout first" expression of interest).

Restrictions

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:

Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:

End_Context

Express_Context_Interest

Switch_Context

Unique parameters

For information about common parameters, see "Parameter list" on page 100.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
5

Decimal
5

Equate symbol
ATR_BACKOUT_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. The following flags are meaningful; the others are set to zero.

Bit	Meaning, if Bit is On
0	The UR interest being processed is for a restart expression of interest.
1	The context is ending. Therefore RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
2	The installation used an RRS panel to commit or back out the UR, which had been in an **in-doubt** state.
3	RRS has detected a heuristic-mixed condition for this UR.
6	The backout was requested by the application.
9	The application encountered an asynchronous abend.
10	The ATRSROI callable service should not be issued.
11	The UR being processed is a cascaded UR.
12	The UR being processed is in local transaction mode. If neither bit 12 nor bit 13 is on, the UR being processed is in hybrid-global transaction mode.
13	The UR being processed is in global transaction mode. In this case, bit 1 is also on, indicating that the resource manager should not attempt to call the Retain_Interest service. If neither bit 12 nor bit 13 is on, the UR being processed is in hybrid-global transaction mode.

value1
value2
value3
value4
value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the BACKOUT exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	**Meaning and action**
0 (0) ATRX_OK	**Meaning**: The resource manager completed backout processing for the interest in the UR. RRS continues with backout processing. **Action**: None.
4 (4) ATRX_OK_OUTCOME_PENDING	**Meaning**: The resource manager completed backout processing for the interest in the UR; however, not all changes are complete. RRS records this exception in logrec. RRS continues with backout processing. **Action**: None.
10 (16) ATRX_FORGET	**Meaning**: The resource manager completed backout processing for the interest in the UR. The resource manager no longer has an interest in the UR. RRS deletes this interest in the UR. RRS does not invoke any subsequent exit routines for this interest in the UR. **Action**: None.
24 (36) ATRX_HC	**Meaning**: The exit routine detected a heuristic commit for the UR. Resources have already been changed. RRS records this exception in logrec. RRS continues with backout processing. **Action**: None.
28 (40) ATRX_HR	**Meaning**: The exit routine detected a heuristic reset for the UR. RRS records this exception in logrec. RRS continues with backout processing. **Action**: None.
2C (44) ATRX_HM	**Meaning**: The exit routine detected a heuristic mix for the UR. That is, some resources have been committed and some have been backed out. RRS records this exception in logrec. RRS continues with backout processing for the UR. **Action**: None.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
30 (48) ATRX_LATER	**Meaning**: The resource manager will provide a return code at a later time. Later, the resource manager calls the Post_Deferred_UR_Exit service to give the return code to RRS. **Action**: None.
40 (64) ATRX_DEFER	**Meaning**: The resource manager requests that RRS defer the exit processing for this expression of interest. **Action**: RRS will re-invoke the backout exit for this expression of interest after all other back out exits for this resource manager have been invoked for this UR.

COMMIT exit routine

The COMMIT exit routine receives control when the UR state is **in-commit**. RRS invokes the COMMIT exit routine when it determines that the UR should be committed or when:

- A resource manager that has taken the SDSRM role calls the Commit_Agent_UR service.
- An application or a work manager calls the End_Transaction service.

Note: The resource manager can specify in a call to the Set_Syncpoint_Controls service the return code for the COMMIT exit routine. In this case, RRS does not invoke the routine.

Processing

The COMMIT exit routine should commit the UR by making the changes in all resources for this interest in the UR.

Restrictions

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:

Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:

End_Context

Express_Context_Interest

Switch_Context

Unique parameters

For information about common parameters, see “Parameter list” on page 100.

exit_number

Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
4

Decimal
4

Equate symbol
ATR_COMMIT_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. The following flags are meaningful; the others are set to zero.

Bit	Meaning, if Bit is On
0	The UR interest being processed is for a restart expression of interest.
1	The context is ending. Therefore RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
2	The installation used an RRS panel to commit or back out the UR, which had been in an **in-doubt** state.
3	RRS has detected a heuristic-mixed condition for this UR.
6	The backout was requested by the application.
9	The application encountered an asynchronous abend.
10	The ATRSROI callable service should not be issued.
11	The UR being processed is a cascaded UR.
12	The UR being processed is in local transaction mode. If neither bit 12 nor bit 13 is on, the UR being processed is in hybrid-global transaction mode.
13	The UR being processed is in global transaction mode. In this case, bit 1 is also on, indicating that the resource manager should not attempt to call the Retain_Interest service. If neither bit 12 nor bit 13 is on, the UR being processed is in hybrid-global transaction mode.

value1
value2
value3
value4
value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the COMMIT exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
0 (0) ATRX_OK	**Meaning**: The resource manager completed commit processing for the interest in the UR. RRS continues processing the UR. **Action**: None.
4 (4) ATRX_OK_OUTCOME_PENDING	**Meaning**: The resource manager completed commit processing for the interest in the UR; however, not all updates are complete. RRS records this exception in logrec. RRS continues with commit processing. **Action**: None.
10 (16) ATRX_FORGET	**Meaning**: The resource manager completed commit processing for the interest in the UR. The resource manager no longer has an interest in this UR. RRS deletes this interest in the UR. RRS does not invoke any subsequent exit routines for this interest in the UR. **Action**: None.
24 (36) ATRX_HC	**Meaning**: The exit routine detected a heuristic commit for the UR. Resources have already been changed. RRS records this exception in logrec. RRS continues with commit processing. **Action**: None.
28 (40) ATRX_HR	**Meaning**: The exit routine detected a heuristic reset for the UR. RRS records this exception in logrec. RRS continues with commit processing. **Action**: None.
2C (44) ATRX_HM	**Meaning**: The exit routine detected a heuristic mix for the UR. That is, some resources have been committed and some have been backed out. RRS records this exception in logrec. RRS continues with commit processing for the UR. **Action**: None.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
30 (48) ATRX_LATER	**Meaning**: The resource manager will provide a return code at a later time. Later, the resource manager calls the Post_Deferred_UR_Exit service to give the return code to RRS. **Action**: None.

COMPLETION exit routine

The COMPLETION exit routine receives control for each interest in a UR after any resource manager with an interest in the UR specifies ATR_DRIVE_COMPLETION in a call to the Set_Side_Information service. RRS invokes the COMPLETION exit routine after:

- The resource manager completes syncpoint processing but before RRS returns control to the application program.
- An application or a work manager calls the End_Transaction service.

Bit 8 in *exit_flags* indicates whether the exit routine is invoked for commit or for backout.

The exit routine is useful when a work request is distributed across several systems.

Processing

The COMPLETION exit routine allows a communication resource manager to deallocate some or all conversations after RRS completes processing for a UR, but before RRS gives control to the application program.

For example, the COMPLETION exit routine can abnormally deallocate all outbound conversations for a distributed syncpoint communication resource manager, if a failure occurs on one conversation for a UR.

A resource manager can also use the COMPLETION exit routine to obtain the LUWID (logical unit of work identifier) for the next UR, created by an earlier call to the Retain_Interest service. A call to the Retrieve_Work_Identifier service, using the UR interest token created through the Retain_Interest service, returns the LUWID of the next UR. This technique works only when the current UR state is **in-completion** or, if the resource manager has taken the SDSRM role, **in-forget**.

Restrictions

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:

Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:

End_Context

Express_Context_Interest

Switch_Context

Unique parameters

For information about common parameters, see “Parameter list” on page 100.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
8

Decimal
8

Equate symbol
ATR_COMPLETION_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. The following flags are meaningful; the others are set to zero.

Bit	Meaning, if Bit is On
0	The UR interest being processed is for a restart expression of interest.
1	The context is ending. Therefore RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
2	The installation used an RRS panel to commit or back out the UR, which had been in an **in-doubt** state.
3	RRS has detected a heuristic-mixed condition for this UR.
4	RRS detected a resync in progress for the UR.
5	For the UR, the collected prepare vote was FORGET. meaning that the prepare votes were ABSTAIN or FORGET. The vote for this interest in the UR was ABSTAIN.
6	The backout was requested by the application.
8	If set, the overall vote for the UR is commit. If not set, the overall vote for the UR is backout.
9	The application encountered an asynchronous abend.
10	The ATRSROI callable service should not be issued.
11	The UR being processed is a cascaded UR.
12	The UR being processed is in local transaction mode. If neither bit 12 nor bit 13 is on, the UR being processed is in hybrid-global transaction mode.
13	The UR being processed is in global transaction mode. In this case, bit 1 is also on, indicating that the resource manager should not attempt to call the Retain_Interest service. If neither bit 12 nor bit 13 is on, the UR being processed is in hybrid-global transaction mode.

value1
value2
value3
value4

value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the COMPLETION exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	**Meaning and action**
0 (0) ATRX_OK	**Meaning**: The resource manager completed COMPLETION exit processing for the interest in the UR. RRS continues processing the UR. **Action**: None.
10 (16) ATRX_FORGET	**Meaning**: The resource manager completed COMPLETION exit processing for the interest in the UR. The resource manager no longer has an interest in this UR. RRS deletes this interest in the UR. RRS does not invoke any subsequent exit routines for this interest in the UR.
30 (48) ATRX_LATER	**Meaning**: The resource manager will provide a return code at a later time. Later, the resource manager calls the Post_Deferred_UR_Exit service to give the return code to RRS. **Action**: None.
34 (52) ATRX_LATER_CONTINUE	**Meaning**: The resource manager will provide a return code at a later time, but the application or transaction program can continue processing, which means that the context could end before the UR is complete. (If any resource manager has called Set_Syncpoint_Controls to take the SDSRM role for this UR, RRS treats this return code as if it were ATRX_LATER.) Later, the resource manager calls the Post_Deferred_UR_Exit service to give the return code to RRS. **Action**: None.

DISTRIBUTED_SYNCPOINT exit routine

The DISTRIBUTED_SYNCPOINT exit routine is for resource managers in a distributed resource recovery environment. A resource manager's DISTRIBUTED_SYNCPOINT exit routine is enabled when the resource manager takes the distributed syncpoint resource manager role (specifies ATR_DSRM in a

call to the Set_Syncpoint_Controls service). RRS invokes the DISTRIBUTED_SYNCPOINT exit routine after the PREPARE exit routines for all interests in a UR:

- Vote OK to request commit.
- Return ATRX_ABSTAIN or ATRX_FORGET.

If the overall prepare vote is FORGET, but at least one resource manager returned ABSTAIN, RRS does not invoke the DISTRIBUTED_SYNCPOINT exit routine. Instead, RRS invokes the END_UR exit routines for those interests that have returned ABSTAIN. Additional information appears in "Vote collection" on page 63 and "PREPARE exit routine" on page 126.

Note: For a UR, only one resource manager can request the distributed syncpoint role and the request can be for only one interest. The call to the Set_Syncpoint_Controls service is usually issued in the resource manager's STATE_CHECK or PREPARE exit routine.

Processing

The DISTRIBUTED_SYNCPOINT routine is responsible for resolving an **in-doubt** UR:

- It communicates with another system to inform it of the result of the local prepare vote and to receive from that system the overall distributed prepare vote.
- It returns the other system's overall commit or backout vote to RRS.

RRS can then continue with the appropriate commit or backout processing.

If all the local PREPARE exit routines reply ABSTAIN or FORGET, RRS does not drive the DISTRIBUTED_SYNCPOINT exit routine.

Restrictions

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:

Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:

End_Context

Express_Context_Interest

Switch_Context

Unique parameters

For information about common parameters, see "Parameter list" on page 100.

exit_number

Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
3

Decimal
3

Equate symbol
ATR_DISTRIBUTED_SYNCPOINT_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. The following flags are meaningful; the others are set to zero.

Bit	Meaning, if Bit is On
0	The UR interest being processed is for a restart expression of interest.
1	The context is ending. Therefore RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
10	The ATRSROI callable service should not be issued.
11	The UR being processed is a cascaded UR.

value1
value2
value3
value4
value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the DISTRIBUTED_SYNCPOINT exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and Action
0 (0) ATRX_OK	**Meaning**: The exit routine determined that the UR should be committed. RRS continues commit processing for the UR. **Action**: None.
8 (8) ATRX_BACKOUT	**Meaning**: The exit routine determined that the UR should be backed out. RRS proceeds with backout processing for the UR. **Action**: None.
10 (16) ATRX_FORGET	**Meaning**: The exit routine determined that the UR should be committed. RRS continues commit processing for the UR. The resource manager no longer has an interest in this UR. RRS deletes this interest in the UR. RRS does not invoke any subsequent exit routines for this interest in the UR. **Action**: None.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and Action
30 (48) ATRX_LATER	**Meaning**: The resource manager will provide a return code at a later time. Later, the resource manager calls the Post_Deferred_UR_Exit service to give the return code to RRS. **Action**: None.
38 (56) ATRX_HM_BACKOUT	**Meaning**: The exit routine detected a heuristic mix for the UR. RRS records this exception in logrec. RRS proceeds with backout processing for the UR. **Action**: None.
3C (60) ATRX_HM_COMMIT	**Meaning**: The exit routine detected a heuristic mix for the UR. RRS records this exception in logrec. RRS proceeds with commit processing for the UR. **Action**: None.

END_UR exit routine

The END_UR exit routine receives control when the UR state reaches **in-end**.

Processing

The END_UR exit routine should clean up private resource manager structures used for the UR.

A communications resource manager can use an END_UR exit routine to determine the final outcome of the commit process and communicate this outcome to other systems.

Note: Do not use the END_UR exit routine to delay completion of commit or backout processing. A delay might cause an incorrect final return code to be sent to the application programs on other systems.

Restrictions

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:

Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:

End_Context

Express_Context_Interest

Switch_Context

Unique parameters

For information about common parameters, see "Parameter list" on page 100.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
6

Decimal
6

Equate symbol
ATR_END_UR_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. The following flags are meaningful; the others are set to zero.

Bit	Meaning, if Bit is On
0	The UR interest being processed is for a restart expression of interest.
1	The context is ending. Therefore RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
3	RRS has detected a heuristic-mixed condition for this UR.
4	RRS detected a resync in progress for the UR.
5	For the UR, the collected prepare vote was FORGET. meaning that the prepare votes were ABSTAIN or FORGET. The vote for this interest in the UR was ABSTAIN.
6	The backout was requested by the application.
8	The overall vote for the UR is commit.
10	The ATRSROI callable service should not be issued.
11	The UR being processed is a cascaded UR.

value1
value2
value3
value4
value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the END_UR exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and Action
0 (0) ATRX_OK	**Meaning**: The resource manager completed processing for the interest in the UR. RRS continues processing the UR. **Action**: None.
10 (16) ATRX_FORGET	**Meaning**: The resource manager completed processing for the interest in the UR. RRS continues processing the UR. The resource manager no longer has an interest in this UR. RRS deletes this interest in the UR. RRS does not invoke any subsequent exit routines for this interest in the UR. **Action**: None.
30 (48) ATRX_LATER	**Meaning**: The resource manager will provide a return code at a later time. Later, the exit routine calls the post exit routine service to give the return code to RRS. **Action**: None.

EXIT_FAILED exit routine

The EXIT_FAILED exit routine receives control when one of the RRS exit routines fails. RRS gives this routine the exit number of the failed routine and the reason why the routine failed.

RRS, at most, invokes the EXIT_FAILED exit once for each UR state for any one expression of interest. Also, EXIT_FAILED processing might overlap with resource manager processing to post the results of an exit by calling Post_Deferred_UR_Exit at a later time. Thus, the completion of Post_Deferred_UR_Exit and the EXIT_FAILED exit routine can occur in any order:

- If Post_Deferred_UR_Exit completes first, RRS ignores the return code from EXIT_FAILED.
- If EXIT_FAILED completes first, RRS returns the ATR_POST_NOT_PENDING code to Post_Deferred_UR_Exit.

If the resource manager requests it on a call to the Set_Side_Information service, RRS will drive the EXIT_FAILED exit routine when an exit routine has returned ATRX_LATER and an asynchronous abend or address space termination has occurred.

Processing

The EXIT_FAILED exit routine should provide RRS a return code for the failing exit or tell RRS to unset the resource manager's RRS exit routines.

If the EXIT_FAILED exit routine fails, RRS unsets the resource manager's RRS exit routines.

Restrictions

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:

Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:

End_Context

Express_Context_Interest

Switch_Context

Unique parameters

For information about common parameters, see "Parameter list" on page 100.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
7

Decimal
7

Equate symbol
ATR_EXIT_FAILED_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. The flags apply to the failed exit routine; see the description of the failed exit routine.

value1
Points to a field that contains the exit number of the failed exit routine. For the number, see the description of the failed exit routine or Table 17 on page 84. Define the field as a 4-byte integer.

value2
Points to a field that contains the reason why the exit routine failed. Define the field as a 4-byte integer. The field contains one of the following:

Value in: Hexadecimal (Decimal) Equate Symbol	**Reason**
1 (1) ATR_EXIT_RC_NOT_VALID	**Incorrect return code**: An exit routine returned an incorrect return code to RRS. The incorrect code is in the *value3* parameter.
2 (2) ATR_EXIT_ABENDED	**Exit routine abended**: The abend percolated to the FRR for RRS. The ABEND code is in the *value3* parameter.
3 (3) ATR_REDRIVE_LIMIT	**STATE_CHECK invoked too many times**: The STATE_CHECK exit routine has been invoked for a UR more times than allowed.

Value in: Hexadecimal (Decimal) Equate Symbol	Reason
4 (4) ATR_RC_INCORRECT_AFTER_POST	**Inappropriate call**: The resource manager called the Post_Deferrerd_UR_Exit service with a valid completion code. However, the exit routine specified in the call had previously returned a code other than ATRX_LATER or ATRX_LATER_CONTINUE.
5 (5) ATR_MEMTERM	**Abnormal end**: The dispatchable unit associated with the context or the owner of the unassociated privately managed context started abnormal termination while the exit routine was running.
6 (6) ATR_FORGET_NOT_VALID	**Incorrect use of FORGET**: An exit routine set an ATRX_FORGET return code when the resource manager it represents had the DSRM or SDSRM role, but the exit routine has not yet completed all of its responsibilities to resolve the UR. A DSRM or SDSRM can return ATRX_FORGET only after the UR has come out of **in-doubt** state.
7 (7) ATR_EXIT_ABENDED_RSN	**Exit routine abended**: The abend percolated to the FRR for RRS. The ABEND code is in the *value3* parameter and its reason code is in *value4*.
8 (8) ATR_ASYNC_ABEND	**Asynchronous abend while RRS was waiting:** The SRB or task that requested the syncpoint was asynchronously abended while RRS was waiting for Post_Deferred_UR_Exit to complete a response from an exit, or an exit returned ATRX_LATER while RRS was processing an asynchronously abended syncpoint. The *value3* parameter contains the abend code. This value appears only when ATR_EF_ON_LATER_WITH_ASYNC was specified on a call to Set_Exit_Information for RRS.
9 (9) ATR_ASYNC_ABEND_RSN	**Asynchronous abend while RRS was waiting:** The SRB or task that requested the syncpoint was asynchronously abended while RRS was waiting for Post_Deferred_UR_Exit to complete a response from an exit, or an exit returned ATRX_LATER while RRS was processing an asynchronously abended syncpoint. The *value3* parameter contains the abend code, and *value4* contains the reason code. This value appears only when ATR_EF_ON_LATER_WITH_ASYNC was specified on a call to Set_Exit_Information for RRS.

Value in: Hexadecimal (Decimal) Equate Symbol	Reason
A (10) ATR_ASYNC_MEMTERM	**Address space termination while RRS was waiting:** The SRB or task that requested the syncpoint had its address space terminated while RRS was waiting for Post_Deferred_UR_Exit to complete a response from an exit, or an exit returned ATRX_LATER while RRS was processing a syncpoint that had its address space terminated. The *value3* parameter contains the abend code, and *value4* contains the reason code. This value appears only when ATR_EF_ON_LATER_WITH_ASYNC was specified on a call to Set_Exit_Information for RRS.
B (11) ATR_ALREADY_DEFERRED	**Exit routine already deferred:** The resource manager has already requested RRS to defer the exit routine for this expression of interest.
C (12) ATR_ALL_DEFERRED	**All deferred:** The resource manager requested RRS to defer the exit routine for all of its expressions of interest.
D (13) ATR_DEFER_NOT_VALID	**Inappropriate defer request:** The resource manager requested RRS to defer the exit routine but exit processing has previously completed for one of the resource manager's expression of interest.
E (14) ATR_DEFER_SRB_NOT_VALID	**Deferment invalid for SRB exit routine:** The resource manager requested RRS to defer the SRB exit routine. SRB exit routines cannot be deferred.

value3

Points to a field that contains the following code, depending on *value2*. Define the field as a 4-byte integer.

Value in *value2*	Contents of *value3* Field
ATR_EXIT_RC_NOT_VALID	The incorrect return code
ATR_EXIT_ABENDED	The ABEND code
ATR_EXIT_ABENDED_RSN	The ABEND code
ATR_ASYNC_ABEND	The ABEND code
ATR_ASYNC_ABEND_RSN	The ABEND code
ATR_ASYNC_MEMTERM	The ABEND code
Any other value	Binary zeros

value4
Points to a field that contains the following code, depending on *value2*. Define the field as a 4-byte integer.

Value in *value2*	**Contents of** *value4* **Field**
ATR_EXIT_ABENDED_RSN	The ABEND reason code
ATR_ASYNC_ABEND_RSN	The ABEND reason code
ATR_ASYNC_MEMTERM	The ABEND reason code
Any other value	Binary zeros

value5
Points to a field that contains binary zeros. Define the field as a 4-byte integer.

ABEND codes

The DISTRIBUTED_SYNCPOINT exit routine might end with an abend X'5C4' with a X'xxxx0006' reason code. If your exit routine provides a recovery environment, do not retry for this abend; RRS must stop the exit routine. To detect this abend, ignore the first four digits of the reason code and test for X'0006' in the lower half of the word that contains the abend reason code.

Return codes

When the EXIT_FAILED exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: **Hexadecimal** **(Decimal)** **Equate Symbol**	**Meaning and action**
hh (ddd) An exit-specific equate symbol	**Meaning**: The EXIT_FAILED exit routine completed processing. The return code is valid for the failed exit; see the return codes for the failed exit. **Action**: Perform the action for the return code.
404 (1028) ATRX_UNSET_RM	**Meaning**: The EXIT_FAILED exit routine completed processing. RRS is to unset the RRS exit routines for the resource manager. **Action**: The resource manager should load a new copy of the failed exit routine, then call the set exit routine service to set all of its exit routines with RRS again. If the problem repeats, you should check the failed exit routine for a probable coding error. Correct the routine and rerun the resource manager.

ONLY_AGENT exit routine

The ONLY_AGENT exit routine receives control when there is only one expression of interest in the UR. RRS does not need to process this UR with a two-phase commit protocol.

Besides ONLY_AGENT, a resource manager could also have PRE_PREPARE and/or STATE_CHECK exit routines. The exit routines get control in this order if they exist: PRE_PREPARE first, STATE_CHECK second, and ONLY_AGENT last.

If the PRE_PREPARE exit routine adds interests, the ONLY_AGENT exit routine will not be called since ONLY_AGENT requires only one Resource Manager with a single expression of interest.

Processing

The ONLY_AGENT exit routine should change or not change its resources. The routine can unilaterally decide to commit or back out the resources. When the ONLY_AGENT exit routine returns control, RRS considers the UR processing to be complete.

The exit routine or resource manager is responsible for hardening into a log any information required to ensure its commit or backout completes. RRS does not log any information for the UR.

Restrictions

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:

Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:

End_Context

Express_Context_Interest

Switch_Context

Unique parameters

For information about common parameters, see “Parameter list” on page 100.

exit_number

Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
9

Decimal
9

Equate symbol
ATR_ONLY_AGENT_EXIT

exit_flags

Points to a field of exit flags. Define the field as a 4-byte integer. The following flags are meaningful; the others are set to zero.

Bit	Meaning, if Bit is On
1	The context is ending. Therefore RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
10	The ATRSROI callable service should not be issued.
11	The UR being processed is a cascaded UR.

value1

value2
value3
value4
value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the ONLY_AGENT exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	**Meaning and action**
0 (0) ATRX_OK	**Meaning**: The resource manager completed commit processing for the interest in the UR. RRS returns to the application. **Action**: None.
4 (4) ATRX_OK_OUTCOME_PENDING	**Meaning**: The resource manager completed commit processing for the interest in the UR; however, not all updates are complete. RRS records this exception in logrec. RRS returns to the application. **Action**: None.
8 (8) ATRX_BACKOUT	**Meaning**: The resource manager backed out the interest in the UR. RRS returns to the application. **Action**: None.
C (12) ATRX_BACKOUT_OUTCOME_ PENDING	**Meaning**: The resource manager backed out the interest in the UR. However, not all processing is complete. RRS returns to the application. **Action**: None.
24 (36) ATRX_HC	**Meaning**: The exit routine detected a heuristic commit for the UR. RRS records this exception in logrec. RRS returns to the application. **Action**: None.
28 (40) ATRX_HR	**Meaning**: The exit routine detected a heuristic reset for the UR. RRS records this exception in logrec. RRS returns to the application. **Action**: None.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
2C (44) ATRX_HM	**Meaning**: The exit routine detected a heuristic mix for the UR. RRS records this exception in logrec. RRS returns to the application. **Action**: None.
30 (48) ATRX_LATER	**Meaning**: The resource manager will provide a return code at a later time. Later, the exit routine calls the Post_Deferred_UR_Exit service to give the return code to RRS. **Action**: None.

PRE_PREPARE exit routine

The PRE-PREPARE exit routine receives control after either of the following events occurs:

- An application requests commit (calls the Commit_UR service, or the Application_Commit_UR service, or the End_Transaction service).
- A resource manager that has:
  - Taken SDSRM role calls the Prepare_Agent_UR service or the Degegate_Commit_Agent_UR service.
  - Called the End_Context service.

If a resource manager has both PRE-PREPARE and STATE_CHECK exits, PRE_PREPARE exit gets control before the STATE_CHECK exit.

Processing

The PRE_PREPARE exit routine should perform any processing the resource manager requires prior to proceeding with the commit request.

Restrictions

The PRE_PREPARE exit routine will not be re-driven if the STATE_CHECK exit cause the transaction to transition back to in-flight.

Unique Parameters

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

- Hexadecimal B
- Decimal 11
- Equate symbol ATR_PRE_PREPARE_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. The following flags are meaningful;

Bit	Meaning, if Bit is On
1	The context is ending. Therefore RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
11	The UR being processed is a cascaded UR.

Return codes

When the PRE_PREPARE exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	**Meaning and action**
0 (0) ATRX_OK	**Meaning**: The exit routine has completed its required processing, and now it's ready to proceed with the commit request. **Action**: None.
10 (16) ATRX_FORGET	**Meaning**: The exit routine determined that the resources are ready for a commit. RRS continues commit processing for the UR. The resource manager no longer has an interest in this UR. RRS deletes this interest in the UR. RRS does not invoke any subsequent exit routines for this interest in the UR. **Action**: None.
30 (48) ATRX_LATER	**Meaning**: The resource manager will provide a return code at a later time. Later, the resource manager calls the Post-Deferred_UR_Exit Service to give the return code to RRS. **Action**: None.

PREPARE exit routine

The PREPARE exit routine receives control when the UR state is **in-prepare**, meaning that either (1) the application requested a commit or (2) a resource manager that has taken the SDSRM role called the Prepare_Agent_UR service. In either case, the routine indicates, through a return code, how the commit is to proceed.

As with all exit routines, there is no way to predict the order in which RRS invokes the PREPARE exit routines for the UR when multiple resource managers have interests in a UR. Once RRS has invoked all the PREPARE exit routines for the UR, it uses their return codes to determine the overall vote on the requested commit. See “Vote collection” on page 63 for information about how RRS uses the return codes.

In distributed processing, the PREPARE exit routine of a resource manager on an agent system can return an ABSTAIN return code to keep from influencing the overall prepare vote. If all other PREPARE exit routines on the agent system also return FORGET or ABSTAIN, then the distributed syncpoint resource manager for the agent system can return FORGET to the system initiating the work request. This action prevents an unintended commit, when the other resource managers have indicated that a commit is not necessary.

Processing

The PREPARE exit routine determines if the resource manager can commit the UR and votes, or reports its findings, in the return code. Processing in the routine should consist of:

1. The routine should check to see if the resources are to be changed.

   If not, such as for a read only request, the routine should release locks on all resources involved and return an ATRX_FORGET return code. RRS will not invoke any more resource manager exit routines for this interest in the UR.
2. If resource are to be changed, the routine hardens the undo and redo change records, which the resource manager wrote when the change was requested, by writing the records on nonvolatile external storage that can be accessed during restart after a failure. The routine returns an ATRX_OK return code.

   For all the possible return codes, see “Return codes” on page 128.
3. If the resource manager requires persistent interest data in its own log to recover during restart, the routine must harden the persistent interest data.

Note: For distributed processing, the resource manager can specify in a call to the Set_Syncpoint_Controls service the return code for the PREPARE exit routine. In this case, RRS does not invoke the routine.

Restrictions

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:

Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:

End_Context

Express_Context_Interest

Switch_Context

Unique parameters

For information about common parameters, see “Parameter list” on page 100.

exit_number

Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
2

Decimal
2

Equate symbol
ATR_PREPARE_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. The following flag is meaningful; the others are set to zero.

Bit	Meaning, if Bit is On
1	The context is ending. Therefore RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
10	The ATRSROI callable service should not be issued.
11	The UR being processed is a cascaded UR.

value1
value2
value3
value4
value5
Point to fields that contain binary zeros. Define each field as a 4-byte integer.

Return codes

When the PREPARE exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
0 (0) ATRX_OK	**Meaning**: The resource manager votes to commit the UR. If the resource manager requires persistent interest data to recover during restart, the resource manager must write the data in its log before returning with this return code. RRS continues collecting prepare votes. **Note:** This code is a YES vote. **Action**: None.
8 (8) ATRX_BACKOUT	**Meaning**: The resource manager votes to back out the UR. If, to recover during restart, the resource manager requires persistent interest data not logged by RRS, the resource manager must write the data in its own log before returning with this return code. RRS backs out the UR by invoking the BACKOUT exit routines for all resource managers currently interested in the UR. **Note:** This code is a NO vote. **Action**: None.

Return Code in: **Hexadecimal** **(Decimal)** **Equate Symbol**	**Meaning and action**
C (12) ATRX_BACKOUT_OUTCOME_ PENDING	**Meaning**: The resource manager votes to back out the UR. If the resource manager requires persistent interest data to recover during restart, the resource manager must write the data in its log before returning with this return code. However, all updates are not necessarily complete. RRS backs out the UR by invoking the BACKOUT exit routines for all resource managers currently interested in the UR. RRS records this exception in logrec. **Note:** This code is a NO vote. **Action**: None.
10 (16) ATRX_FORGET	**Meaning**: The resource manager agrees with the commit of the UR. The resource manager no longer has an interest in this UR. RRS deletes this interest in the UR. RRS does not invoke any subsequent exit routines for this interest in the UR. RRS continues collecting prepare votes. **Note:** This code is effectively a YES vote. **Action**: None.
14 (20) ATRX_ABSTAIN	**Meaning**: The resource manager concurs with the prepare vote by the other exit routines for the interest in the UR. The resource manager continues to be interested in the UR. RRS continues collecting prepare votes. Unless the resource manager has taken the SDSRM role, IBM recommends that any resource manager that votes ABSTAIN should also provide an END_UR exit routine. RRS uses exit flag 5 to tell the END_UR exit routine if the overall prepare vote is FORGET. **Note:** This code is effectively a YES vote. **Action**: None.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
24 (36) ATRX_HC	**Meaning**: The resource manager detects a heuristic commit for the UR. Resources have already been changed. RRS records this exception in logrec. RRS continues collecting prepare votes. RRS notifies the resource manager of the overall decision for the UR by invoking its COMMIT or BACKOUT exit routine. **Note:** This code is effectively a YES vote. **Action**: None.
28 (40) ATRX_HR	**Meaning**: The resource manager made a heuristic reset decision for the UR. Resources have already been backed out. RRS records this exception in logrec. RRS backs out the UR by invoking the BACKOUT exit routines for all resource managers currently interested in the UR. **Note:** This code is a NO vote. **Action**: None.
2C (44) ATRX_HM	**Meaning**: The resource manager detects a heuristic mix for the UR. That is, some resources have been committed and some have been backed out. RRS records this exception in logrec. RRS backs out the UR by invoking the BACKOUT exit routines for all resource managers currently interested in the UR. **Note:** This code is a NO vote. **Action**: None.
30 (48) ATRX_LATER	**Meaning**: The resource manager will provide a return code at a later time. The exit routine does the PREPARE processing asynchronously. Later, the resource manager calls the Post_Deferred_UR_Exit service to give the return code to RRS. **Action**: None.

STATE_CHECK exit routine

The STATE_CHECK exit routine receives control after either of the following events occurs:

- An application requests commit (calls the Commit_UR service, or the Application_Commit_UR service).

- A resource manager that has taken the SDSRM role calls the Prepare_Agent_UR service.

The resource manager can then verify the state of its resources.

If a resource manager has both an ONLY-AGENT and a STATE_CHECK exit routine, STATE_CHECK gets control before ONLY_AGENT.

Processing

The STATE_CHECK exit routine should check that the state of the resource manager's resources permits the resource manager to proceed with the commit request. This exit routine can be used for distributed resources, typically to check the states of protected conversations.

The STATE_CHECK exit routine might find that the resources are not in a state to proceed with a commit; for example, one protected conversation is in **send** state, but another is in **receive** state. In this case, the routine can change the state of one of the resources and pass an ATRX_REDRIVE return code. In response, RRS invokes all the STATE_CHECK exit routines again.

Note: RRS performs a maximum of N redrives, where N is the number of UR interests with STATE_CHECK exit routines. RRS treats any ATRX_REDRIVE return code as an error if the redrive limit has been exceeded.

RRS invokes the STATE_CHECK exit routines in any order. If any exit routine changes the state of the resources and the change might affect another resource manager, the exit manager should specify the ATRX_REDRIVE return code, which causes RRS to invoke all STATE_CHECK exit routines again.

Alternatively, if a resource is not in a state to proceed, RRS can return the RR_PROGRAM_STATE_CHECK return code for the Application_Commit_UR call or the Prepare_Agent_UR call. In response, the application should initiate resource manager actions that will make the resource manager able to accept the commit when the application issues it again.

If a STATE_CHECK exit driven for a cascaded UR returns ATRX_STATE_INCORRECT, RRS will back out the UR's entire cascaded UR family and return RR_BACKED_OUT to the application.

Restrictions

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:

Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:

End_Context

Express_Context_Interest

Switch_Context

Unique parameters

For information about common parameters, see “Parameter list” on page 100.

exit_number

Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
1

Decimal
1

Equate symbol
ATR_STATE_CHECK_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. The following flag is meaningful; the others are set to zero.

Bit	Meaning, if Bit is On
7	For the UR, the STATE_CHECK exit routine returned an ATRX_REDRIVE code. However, because the redrive limit has been reached for the UR, the return code is not valid.
10	The ATRSROI callable service should not be issued.
11	The UR being processed is a cascaded UR.

value1
value2
value3
value4
value5
Point to fields that contain binary zeros. Define the field as a 4-byte integer.

Return codes

When the STATE_CHECK exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
0 (0) ATRX_OK	**Meaning**: The exit routine determined that the resources are ready for a commit. RRS continues commit processing for the UR. **Action**: None.
10 (16) ATRX_FORGET	**Meaning**: The exit routine determined that the resources are ready for a commit. RRS continues commit processing for the UR. The resource manager no longer has an interest in this UR. RRS deletes this interest in the UR. RRS does not invoke any subsequent exit routines for this interest in the UR. **Action**: None.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
1C (28) ATRX_REDRIVE	**Meaning**: The exit routine determined that RRS should invoke the STATE_CHECK exit routines again. This return code should be used by the STATE_CHECK exit routine of a resource manager that changes the state of a resource when the change might affect other resource managers. For example, a conversation resource manager can change the conversation state. In response, RRS invokes all STATE_CHECK exit routines again for this UR. In this way, the resource manager making the change can ensure that all resource managers will see the changed state of the resource. **Action**: None.
20 (32) ATRX_STATE_INCORRECT	**Meaning**: The exit routine determined that the resources are not ready for a commit. This return code does not mean that the UR should be backed out. Rather, it means that the resource manager is not ready to perform commit processing. In response, RRS stops invoking exit routines. When all of the running STATE_CHECK exit routines have completed, RRS returns to the application with a STATE_CHECK return code unless the UR is cascaded. For a cascaded UR, RRS backs out the UR's entire cascaded UR family and returns RR_BACKED_OUT to the application. **Action**: None.
30 (48) ATRX_LATER	**Meaning**: The resource manager will provide a return code at a later time. Later, the resource manager calls the Post_Deferred_UR_Exit service to give the return code to RRS. **Action**: None.

SUBORDINATE_FAILED exit routine

The SUBORDINATE_FAILED exit routine receives control for a sysplex cascade top-level UR when RRS or any resource manager on a subordinate system fails, the subordinate system itself terminates, or the context associated with the subordinate UR abnormally terminates.. The UR state is in-flight. In addition to marking the

top-level UR of the sysplex cascaded transaction *backout required*, RRS will, if requested, invoke this exit to inform the top-level resource manager about the failure.

Subordinate Failed Exits will not be driven if Pre_Prepare processing detects the failure first. In this case, Pre_Prepare will set the transaction to BackOut before starting the syncpoint.

Processing

The SUBORDINATE_FAILED exit routine should return an ATRX_OK return code. RRS takes no action against the UR.

Restrictions

RRS does not invoke this exit for a resource manager who has expressed interest in a cascaded UR.

Do not call the following service to process the UR passed to the exit routine in the *ur_interest_token* parameter:
Forget_Agent_UR_Interest

Do not call any of the following services to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter:
End_Context
Express_Context_Interest
Switch_Context

Unique parameters

For information about common parameters, see "Parameter list" on page 100.

exit_number
Points to a field that contains the exit number. Define the field as a 4-byte integer. The exit number is:

Hexadecimal
A

Decimal
10

Equate symbol
ATR_SUBORDINATE_FAILED_EXIT

exit_flags
Points to a field of exit flags. Define the field as a 4-byte integer. All flags are set to zero.

value1
value2
value3
value4
value5
Points to a field that contains binary zeros. Define the field as a 4-byte integer.

Return codes

When the SUBORDINATE_FAILED exit routine returns control to RRS, the routine must provide a hexadecimal return code in the *return_code* parameter.

Return Code in: Hexadecimal (Decimal) Equate Symbol	Meaning and action
0 (0) ATRX_OK	**Meaning**: The resource manager acknowledged the notification about the subordinate failure. **Action**: None.

RRS version information

RRS supports the use of the IEFSSREQ macro interface function code 54 to obtain information about the RRS subsystem. RRS fills the following SSVI fields:

Field	Contents
SSVIRLEN	X'0050'
SSVIRVER	X'01'
SSVIFLEN	X'0030'
SSVIVERS	Value in CVTPRODN (for example, SP 6.0.3)
SSVIFMID	Value in CVTPRODI (for example, HBB6603)
SSVICNAM	RRS
SSVIUDOF	X'00000000'
SSVISDOF	X'00000030'
SSVIVLEN	X'001E'
SSVIVDAT	,RRS_COMMAND_PREFIX='SETRRS '

For details about the IEFSSREQ macro interface, see *z/OS MVS Using the Subsystem Interface*.

Chapter 5. Callable registration services

This section describes the callable services that an authorized resource manager can use to request work registration services. The chapter presents the callable services in alphabetical order by descriptive name.

Register_Resource_Manager (CRGGRM, CRG4GRM)

- CRGGRM is for AMODE(31) callers.
- CRG4GRM is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Register_Resource_Manager service (CRGGRM) to register itself with the system. In response to the call, the system returns:

- A return code
- The resource manager token (RM_TOKEN)

A resource manager needs to register before it can issue the required calls to the Set_Exit_Information service to identify itself and its exit routines with exit managers. Note also that the exit routines the resource manager sets are invoked in the same key as the resource manager at the time it calls Set_Exit_Information.

Resource Manager Token: The resource manager token is a random value that is not preserved across restarts of the system, exit manager, or resource manager. Thus:

- Do not use the resource manager token as an identifier in resource manager log records.
- Do not try to discern the contents of the token or create any dependencies on the contents.

You need this token for calls to the following services: Begin_Context, Begin_Restart, End_Restart, Express_Context_Interest, Express_UR_Interest, Retrieve_UR_Interest, Retrieve_Log_Name, Set_Exit_Information, Set_Log_Name, and Unregister_Resource_Manager.

Resource Manager Global Data: Your resource manager can specify global data in the call to the Register_Resource_Manager service. Exit managers obtain the resource manager's global data from the system and pass this global data to all of the resource manager's exit routines. This data should provide the exit routines with an anchor or anchors to resource manager data structures.

A resource manager cannot change this data without unregistering and then registering again. Therefore, it is a good idea to use the data to identify data structures that the resource manager can modify rather than passing the contents of the structures.

The global data can be retrieved by a call to the Retrieve_Resource_Manager_Data service.

Environment

The requirements for the resource manager are:

Minimum authorization:	None
Dispatchable unit mode:	Task
Cross memory mode:	If PKM 8–15 problem state, PASN=HASN=SASN; otherwise, any PASN, any HASN, any SASN
AMODE:	
	31 bit (CRGGRM) 64 bit (CRG4GRM)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine CRGCSS (31 bit) or CRG4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

Table 20. Register_Resource_Manager (CRGGRM, CRG4GRM) Programming requirements

HLL Definition	Description
CRGASM	390 Assembler declarations
CRGC	C/390 declarations
FOMUCRGC	z/OS HFS header files

Restrictions

To call the service, the resource manager state must be **unregistered**. After a successful call, the resource manager state is **registered**.

Resource managers with a PKM allowing PSW key 0–7 or supervisor state must have a PSW key of 0–7 when invoking this service. Resource managers that are PKM 8–15 problem state may not specify CRG_UNREG_EOM for *unregister_option*.

PKM 8–15 problem state RMs must specify a resource manager name that ends in `.UA`, followed by enough trailing blanks to make the entire name 32 bytes long.

By default, a single address space may register a maximum of 256 PKM 8–15 problem state resource managers. This limit may be removed by the operator.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register
Contents

0-1 Used as work registers by the system

2-13 Unchanged

14 Used as a work register by the system

15 Return code

When control returns to the resource manager, the ARs contain:

Register
Contents

0-1 Used as work registers by the system

2-13 Unchanged

14-15 Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CRGGRM	(return_code ,resource_manager_name ,resource_manager_token ,unregister_option ,resource_manager_global_data)

CALL CRG4GRM	(return_code ,resource_manager_name ,resource_manager_token ,unregister_option ,resource_manager_global_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A

- Length: 4 bytes

Contains the return code from the Register_Resource_Manager service.

,resource_manager_name
Supplied parameter

- Type: Character string
- Character Set: See note
- Length: 32 bytes

Specifies the name of the resource manager making the call. For RRS, the name must be unique across a sysplex. For context services, the name must be unique within a system. In either case, the resource manager must use the same name each time it registers.

PKM 8–15 problem state resource managers must specify a resource manager name that ends with the characters `.UA` and trailing blanks (if they are needed). For example, `IEA.GENERAL.SERVER.IBM.UA` would be a valid value.

Note: The name can consist of the following printable characters:

- Alphanumeric characters: A-Z and 0-9
- National characters: $ (X'5B'), # (X'7B'), @ (X'7C')
- The period (.)
- The underscore (_)
- The trailing blank characters needed to fill the 32-byte field

The name may not start with a blank or contain embedded blanks. Lower case characters are folded to upper case characters.

Use the following conventions to avoid name conflicts:

- IBM-provided PKM 0–7 or supervisor state programs use A-C or G-I as the first character and `.IBM` as the ending qualifier.
- IBM-provided PKM 8–15 programs use A-C or G-I as the first character and `.IBM.UA` as the ending qualifier.
- Other PKM 0–7 or supervisor state resource managers should begin the name with D-F or J-Z and end the name with a period and the company name or acronym.
- Other PKM 8–15 resource managers should begin the name with D-F or J-Z and end the name with a period and the company name or acronym followed by `.UA` as the ending qualifier.

For example, PKM 0–7 or supervisor state names could be:

```
IEAV5.IBM
DATAMGR.VENDORCORP
RESMANAGER.GROWTHCOMPANY
```

PKM 8–15 problem state names could be:

```
IEAV5.IBM.UA
DATAMGR.VENDORCORP.UA
RESMANAGER.GROWTHCOMPANY.UA
```

The name specified in this call is also used in a call to the Retrieve_Resource_Manager data service.

,resource_manager_token
Returned parameter

- Type: Character string
- Character Set: No restriction

- Length: 16 bytes

Receives the resource manager token that uniquely identifies the resource manager.

If the resource manager name is already registered, the system returns CRG_RM_NAME_REGISTERED as the return code and returns the token that is already associated with the resource manager name.

,unregister_option

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates how the system is to determine that the resource manager is ending unexpectedly. Specify the event as one of the following:

Table 21. Register_Resource_Manager (CRGGRM, CRG4GRM) unregister_option parameter

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) CRG_UNREG_CMRO	**Cross memory resource-owning task ends:** The system monitors for the ending of the cross memory resource-owning task (the top, or first, job step task in the address space). When the task ends, the system implicitly unregisters the resource manager.
1 (1) CRG_UNREG_CURRENT	**Resource manager, which is the current task, ends:** The system monitors for the ending of the current task. When it ends, the system implicitly unregisters the resource manager.
2 (2) CRG_UNREG_EOM	**The resource manager's address space, which is the home address space, ends:** The system monitors for the ending of the current address space. When it ends, the system implicitly unregisters the resource manager. **Note:** This option cannot be used by PKM 8–15 problem state callers.

,resource_manager_global_data

Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the global data for the resource manager. The system passes this data to all exit routines for the resource manager.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00330000' or X'00330001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Table 22. Register_Resource_Manager (CRGGRM, CRG4GRM) Return codes

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 CRG_OK	**Meaning**: Successful completion. **Action**: None.
103 CRG_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
104 CRG_MODE_INV	**Meaning**: Program error. The resource manager is not in task mode. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 CRG_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CRG_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.
108 CRG_KEY_INV	**Meaning**: Program error. The resource manager is in supervisor state, but in an unauthorized key (8-F). The system rejects the service call because it could cause a system integrity exposure. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Table 22. Register_Resource_Manager (CRGGRM, CRG4GRM) Return codes (continued)

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
10A CRG_XMEM_INV	**Meaning**: Environment error. The resource manager is PKM 8–15 problem state, but not PASN=HASN=SASN. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
300 CRG_RM_NAME_INV	**Meaning**: Program error. The resource manager name specified in the call is incorrect. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
302 CRG_UNREGOPT_INV	**Meaning**: Program error. The *unregister_option* value specified in the call is not valid, possibly because a PKM 8–15 problem state resource manager used the CRG_UNREG_EOM option. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
700 CRG_RM_NAME_REGISTERED	**Meaning**: Program error. The resource manager is already registered. The system rejects the service call. However, the system returns the resource manager token in the *resource_manager_token* field. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 CRG_MAX_RM_EXCEEDED	**Meaning**: Environment error. There are already 256 PKM 8–15 problem state resource managers registered in the current home address space. The system rejects the service call. **Action**: Either allow additional resource managers or change your program so less than 256 resource managers are required.
FFF CRG_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to register. To indicate if the system needs to implicitly unregister the resource manager, the call requests that the system monitor for the ending of the current task.

```
⋮
RM_NAME = MY_RM_NAME
RM_GL_DATA = MY_GLOBAL_DATA
UNREG_OPT = CRG_UNREG_CURRENT
CALL CRGGRM(RC,RM_NAME,RM_TOKEN,
            UNREG_OPT,RM_GL_DATA)
IF RC = CRG_OK THEN
  MY_RM_TOKEN = RM_TOKEN
⋮
```

Retrieve_Resource_Manager_Data (CRGRRMD, CRG4RRMD)

- CRGRRMD is for AMODE(31) callers.
- CRG4RRMD is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Retrieve_Resource_Manager_Data service to obtain the global data related to a specified resource manager. In response to the call, the system returns:

- A return code
- The resource manager token
- The resource manager global data, which was specified in a call to the Register_Resource_Manager service

Environment

The requirements for the resource manager are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (CRGRRMD) 64 bit (CRG4RRMD
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.
Linkage:	Uses standard MVS linkage conventions.

Programming requirements

Either link edit your object code with the linkable stub routine CRGCSS (31 bit) or CRG4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

Table 23. Retrieve_Resource_Manager_Data (CRGRRMD, CRG4RRMD) Programming requirements

HLL Definition	Description
CRGASM	390 Assembler declarations

Table 23. Retrieve_Resource_Manager_Data (CRGRRMD, CRG4RRMD) Programming requirements (continued)

HLL Definition	Description
CRGC	C/390 declarations
FOMUCRGC	z/OS HFS header files

Restrictions

The state of the resource manager associated with the specified resource manager name must be **registered**.

Resource managers that are PKM 8–15 problem state must register using the Register_Resource_Manager service from the home address space before invoking this service. They must specify a resource manager token of a key 8–15 problem state resource manager that registered from the home address space. Some exit managers may not permit unauthorized resource managers to set exits.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CRGRRMD	(return_code ,resource_manager_name ,resource_manager_token ,resource_manager_global_data)

CALL CRG4RRMD	(return_code ,resource_manager_name ,resource_manager_token ,resource_manager_global_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Retrieve_Resource_Manager_Data service.

,resource_manager_name
Supplied parameter

- Type: Character string
- Character Set: See note
- Length: 32 bytes

Specifies the name of the resource manager making the call. The name must match the name specified on a call to the Register_Resource_Manager service.

Note: The name can consist of the following printable characters:

- Alphanumeric characters: A-Z and 0-9.
- National characters: $ (X'5B'), # (X'7B'), @ (X'7C').
- The period (.).
- The underscore (_).
- The trailing blank characters needed to fill the 32-byte field.

The name may not start with a blank or contain embedded blanks. Lower case characters are folded to upper case characters.

,resource_manager_token
Returned parameter

- Type: Character string

- Character Set: No restriction
- Length: 16 bytes

Receives the resource manager token that uniquely identifies the resource manager. The token was assigned when the resource manager called the Register_Resource_Manager service.

,resource_manager_global_data
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives the global data the resource manager provided when it registered.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00350000' or X'00350001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Table 24. Retrieve_Resource_Manager_Data (CRGRRMD, CRG4RRMD) Return codes

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 CRG_OK	**Meaning**: Successful completion. **Action**: None.
103 CRG_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 CRG_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Table 24. Retrieve_Resource_Manager_Data (CRGRRMD, CRG4RRMD) Return codes (continued)

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
107 CRG_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.
300 CRG_RM_NAME_INV	**Meaning**: Program error. The resource manager name specified in the call is incorrect. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 CRG_RM_STATE_ERROR	**Meaning**: Program error. The resource manager is not in a valid state to issue the service call. The resource manager state must be **registered**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
756 CRG_AUTH_FAILURE	**Meaning**: Program error. The resource manager is PKM 8–15 problem state and specified a resource manager token that does not belong to a PKM 8–15 problem state resource manager registered in the home address space. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
FFF CRG_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to retrieve its global data. Storage for the call parameters has been allocated.

```
⋮
RM_TOKEN = MY_RM_TOKEN
```

```
CALL CRGRRMD(RC,RM_NAME,RM_TOKEN,RM_DATA)
⋮
```

Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF)

- CRGSEIF is for AMODE(31) callers.
- CRGSEIF1 is for AMODE(31) callers and also supports the LX reuse facility.
- CRG4SEIF is for AMODE(64) callers, allows parameters in 64 bit addressable storage, and also supports the LX reuse facility.

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)

In response to the call, the system returns a return code.

When the resource manager calls the Set_Exit_Information service, all exit routines specified must be ready to receive control; the exit routines might be driven even before control returns from the call.

Following a successful call to the Set_Exit_Information service, the resource manager is considered to be set with the specified exit manager. The call to Set_Exit_Information is required; your resource manager can, however, issue the call without specifying any exit routines, if the exit manager requires no exit routines, and your resource manager has no exit routines. In this case, specify binary zero in the *exit_count* parameter. If you do not want to set a NOTIFICATION exit routine with registration services, code a binary zero (0) in the *notification_exit_type* parameter.

A PKM 8–15 problem state resource manager cannot set a NOTIFICATION exit routine. A PKM 8–15 problem state resource manager must specify binary zero (0) in the *notification_exit_type* parameter.

Specifying the Call: A resource manager must issue a call to the Set_Exit_Information service before an exit manager can invoke any of the resource manager's exit routines. The resource manager should call Set_Exit_Information one or more times after its call to the Register_Resource_Manager service. Each call specifies one exit manager and its exit routines. All exit routines required by an exit manager must be specified on the first call for the exit manager. The required exit routines are:

- For context services: None
- For RRS: BACKOUT, COMMIT, PREPARE, and EXIT_FAILED

Note: Context services supports PKM 8–15 problem state resource managers, but RRS does not.
Subsequent calls for an exit manager can replace exit information, and can add and delete optional exit routines, but cannot delete a required exit routine.

Note: A resource manager can bypass the required RRS BACKOUT, COMMIT, and PREPARE exit routines by specifying their return codes in a call to the Set_Syncpoint_Controls service. This process is called prevoting.

Notification Exit Routine: Like a resource manager, each exit manager must register with the system. If the exit manager specified in a Set_Exit_Information call is not registered, the system returns CRG_EM_STATE_ERROR as the return code. If and when the exit manager registers, the system gives control to the NOTIFICATION exit routine, if provided; and the routine can call the Set_Exit_Information service again.

The NOTIFICATION exit routine, if provided, also gets control when an exit manager unregisters or has its exits unset. Note that some exit managers, such as context services, are always available and never unregister.

Parameter Array for Exit Routines: To set more than one exit routine in a call, specify each of the following parameters as a 1-dimensional array: *exit_number*, *exit_entry*, and *exit_type*. The first position in each parameter array specifies the first exit routine, the second position specifies the second exit routine, and so on. The *exit_count* indicates the number of exit routines and, thus, the number of positions in each array.

Environment

The requirements for the resource manager are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (CRGSEIF) 31 bit (CRGSEIF1) 64 bit (CRG4SEIF)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Link edit your object code with the linkable stub routine CRGCSS (31 bit) or CRG4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
CRGASM	390 Assembler declarations
CRGC	C/390 declarations
FOMUCRGC	z/OS HFS header files

Restrictions

The state of the resource manager associated with the specified resource manager token must be **registered**. After a successful call, the resource manager state is **set**, which means it has set its exits with the exit manager specified in the call.

The resource manager can specify an exit entry, then later alter it. Make such a change very carefully because exit routines could be currently running.

Resource managers that are PKM 8–15 problem state must register using the Register_Resource_Manager service from the home address space before invoking this service. They must specify a resource manager token of a key 8–15 problem state resource manager that registered from the home address space.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CRGSEIF	(return_code ,resource_manager_token ,notification_exit_type ,notification_exit_entry ,exit_manager_name ,exit_count ,exit_number ,exit_entry ,exit_type ,variable_data_1 ,variable_data_2 ,variable_data_3)

CALL CRGSEIF1	(return_code ,resource_manager_token ,notification_exit_type ,notification_exit_entry8 ,exit_manager_name ,exit_count ,exit_number ,exit_entry8 ,exit_type ,variable_data_1 ,variable_data_2 ,variable_data_3)

CALL CRG4SEIF	(return_code ,resource_manager_token ,notification_exit_type ,notification_exit_entry64 ,exit_manager_name ,exit_count ,exit_number ,exit_entry64 ,exit_type ,variable_data_1 ,variable_data_3)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Set_Exit_Information service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

,notification_exit_type
Supplied parameter

- Type: Integer
- Length: 4 bytes

Indicates the type of NOTIFICATION exit routine, based on how the system is to give the routine control if the exit manager registers or unregisters after this call, or unsets the resource manager's exits. Specify one of the following:

Table 25. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) notification_exit_type parameter

Constant in: Hexadecimal (Decimal) Equate Symbol	**Description**
0 (0) CRG_EXIT_TYPE_NONE	**No routine:** When the exit manager registers or unregisters, the system does not give control to a NOTIFICATION exit routine. The system ignores the *notification_exit_entry* field in the call. **Note:** This is the only Notification exit type supported for PKM 8–15 problem state resource managers.
1 (1) CRG_EXIT_TYPE_SRB	**SRB:** When the exit manager registers or unregisters, or has its exits unset, the system schedules an SRB in the resource manager's address space to give control to the NOTIFICATION exit routine. The primary address space when the resource manager registers is the resource manager's address space. **Note:** This exit type is not supported for PKM 8–15 problem state resource managers.
2 (2) CRG_EXIT_TYPE_PC	**Program Call (PC):** When the exit manager registers or unregisters, or has its exits unset, the system issues a PC instruction to give control to the NOTIFICATION exit routine. The primary address space when the resource manager registers is the resource manager's address space. **Note:** This exit type is not supported for PKM 8–15 problem state resource managers.

Table 25. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) notification_exit_type parameter (continued)

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
4 (4) CRG_EXIT_TYPE_PCS	**Program Call (PC):** When the exit manager registers or unregisters, or has its exits unset , the system issues a PC instruction to give control to the NOTIFICATION exit routine. The exit routine is defined through a PC number with a sequence number. The primary address space when the resource manager registers is the resource manager's address space. **Note:** 1. This exit type is not supported for PKM 8-15 problem state resource managers. 2. This exit type is only valid with CRGSEIF1 and CRG4SEIF.

,notification_exit_entry
Supplied parameter

- Character Set: N/A
- Length: 4 bytes

Specifies the entry point for the NOTIFICATION exit routine, as follows:

- The address of the routine

If you coded CRG_EXIT_TYPE_NONE on the *notification_exit_type* parameter, the system ignores this parameter, but you might code a binary zero to indicate no exit routine here as well.

,notification_exit_entry8
Supplied parameter

- Character Set: N/A
- Length: 8 bytes

Specifies the entry point for the NOTIFICATION exit routine, as follows:

If you coded CRG_EXIT_TYPE_NONE on the *notification_exit_type* parameter, the system ignores this parameter, but you might code binary zeros to indicate no exit routine here as well.

If you coded CTX_EXIT_TYPE_SRB on the exit_type parameter, the high order word should be binary zeros, the low order word is the address of the SRB routine.

If you coded CRG_EXIT_TYPE_PC on the notification_exit_type parameter, the high word must contain binary zeros.

If you coded CRG_EXIT_TYPE_PCS on the notification_exit_type parameter, the high word must contain the sequence number.

,notification_exit_entry64
Supplied parameter

- Character Set: N/A

- Length: 8 bytes

Specifies the entry point for the NOTIFICATION exit routine, as follows:

If you coded CRG_EXIT_TYPE_NONE on the *notification_exit_type* parameter, the system ignores this parameter, but you might code binary zeros to indicate no exit routine here as well.

If you coded CTX_EXIT_TYPE_SRB on the exit_type parameter, the high order word should be binary zeros, the low order word is the address of the SRB routine.

If you coded CRG_EXIT_TYPE_PC on the notification_exit_type parameter, the high word must contain binary zeros.

If you coded CRG_EXIT_TYPE_PCS on the notification_exit_type parameter, the high word must contain the sequence number.

,exit_manager_name

Supplied parameter

- Type: Character string
- Character Set: See note
- Length: 16 bytes

Specifies the name of the resource manager that is functioning as an exit manager and that is being informed of the exit routines. If a resource manager sets exits with RRS, its name is preserved across restarts of the system or restarts of RRS.

The names of the IBM-provided exit managers are:

- Context services: `CTX.EXITMGR.IBM`
- Registration services: `CRG.REGSERV.IBM`

  Note: While `CRG.REGSERV.IBM` is an exit manager, you cannot set exits using CRGSEIF. You can only set exits for registration services with CRGGRM.
- RRS: `ATR.EXITMGR.IBM`

,exit_count

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies, in hexadecimal, the number of exit routines defined in the call. The maximum count is the total number of exits for the exit manager:

- For context services: 5
- For registration services: 1
- For RRS: 10

When the call specifies more than one exit routine, the count indicates the number of positions in the array for each of the following parameters: *exit_number*, *exit_entry*, and *exit_type*.

When the call does not define any exit routines, specify binary zeros in the *exit_count* parameter.

Context services does not support any exit types for PKM 8–15 problem state resource managers. A PKM 8–15 problem state resource manager must specify binary zero (0) in the *exit_count* parameter when setting exits with context services.

,exit_number

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the exit number assigned to the exit by the exit manager. When the call specifies more than one exit routine, the numbers are the first row of the array.

If the call does not specify any exit routines in the *exit_count* and associated parameters, specify binary zeros in the *exit_number* parameter.

For **context services**, the exit routines and the numbers assigned by the exit manager are:

Table 26. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) exit_number parameter

Exit	Exit Number in: Hexadecimal (Decimal) Equate Symbol
EXIT_FAILED	1 (1) CTX_EXIT_FAILED_EXIT
CONTEXT_SWITCH	2 (2) CTX_SWITCH_EXIT
PVT_CONTEXT_OWNER	3 (3) CTX_PRIVATE_CONTEXT_OWNER
END_CONTEXT	4 (4) CTX_END_CONTEXT_EXIT
EOM_CONTEXT	5 (5) CTX_EOM_CONTEXT_EXIT

For **RRS**, the exit routines and the numbers assigned by the exit manager are:

Table 27. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) exit number paramater

Exit	Exit Number in: Hexadecimal (Decimal) Equate Symbol
STATE_CHECK	1 (1) ATR_STATE_CHECK_EXIT

Table 27. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) exit number paramater (continued)

Exit	Exit Number in: Hexadecimal (Decimal) Equate Symbol
PREPARE	2 (2) ATR_PREPARE_EXIT
DISTRIBUTED_SYNCPOINT	3 (3) ATR_DISTRIBUTED_SYNCPOINT_EXIT
COMMIT	4 (4) ATR_COMMIT_EXIT
BACKOUT	5 (5) ATR_BACKOUT_EXIT
END_UR	6 (6) ATR_END_UR_EXIT
EXIT_FAILED	7 (7) ATR_EXIT_FAILED_EXIT
COMPLETION	8 (8) ATR_COMPLETION_EXIT
ONLY_AGENT	9 (9) ATR_ONLY_AGENT_EXIT
SUBORDINATE_FAILED	A (10) ATR_SUBORDINATE_FAILED_EXIT
PRE_PREPARE	B (11) ATR_PRE_PREPARE_EXIT

,exit_entry

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the entry point for an exit routine, as follows:

- The address of the routine
- Binary zeros, to indicate no exit routine

When the call specifies more than one exit routine, code the parameters as a 1-dimensional array.

Do not specify zero for a required exit routine.

If zero is specified, the exit is not set. If a subsequent Set_Exit_Information call specifies zero for an exit that was previously set, the exit routine is deleted from the list of exit routines for this resource manager. However, if the previously set exit routine is required, the system does not delete the exit routine and continues to use the previously specified address or PC number.

,exit_entry8
Supplied parameter

- Type: Integer
- Length: 8 bytes

Specifies the entry point for an exit routine, as follows:

When the call specifies more than one exit routine, code the parameters as a 1-dimensional array.

Do not specify zero for a required exit routine. If an exit address is specified, the system only supports a padded 31-bit address.

If you coded CTX_EXIT_TYPE_SRB on the exit_type parameter, the high order word should be binary zeros, the low order word is the address of the SRB routine.

If you coded CTX_EXIT_TYPE_PC or ATR_EXIT_TYPE_PC on the exit_type parameter, the high word must contain zeros.

If you coded CTX_EXIT_TYPE_PCS or ATR_EXIT_TYPE_PCS on the exit_type parameter, the high word must contain the sequence number.

If zero is specified, the exit is not set. If a subsequent Set_Exit_Information call specifies zero for an exit that was previously set, the exit routine is deleted from the list of exit routines for this resource manager. However, if the previously set exit routine is required, the system does not delete the exit routine and continues to use the previously specified address or PC number.

,exit_entry64
Supplied parameter

- Type: Integer
- Length: 8 bytes

Specifies the entry point for an exit routine, as follows:

When the call specifies more than one exit routine, code the parameters as a 1-dimensional array.

Do not specify zero for a required exit routine. If an exit address is specified, the system only supports a padded 64-bit address.

If you coded CTX_EXIT_TYPE_SRB on the exit_type parameter, the high order word should be binary zeros, the low order word is the address of the SRB routine.

If you coded CTX_EXIT_TYPE_PC or ATR_EXIT_TYPE_PC on the exit_type parameter, the high word must contain zeros.

If you coded CTX_EXIT_TYPE_PCS or ATR_EXIT_TYPE_PCS on the exit_type parameter, the high word must contain the sequence number.

If zero is specified, the exit is not set. If a subsequent Set_Exit_Information call specifies zero for an exit that was previously set, the exit routine is deleted from the list of exit routines for this resource manager. However, if the previously set exit routine is required, the system does not delete the exit routine and continues to use the previously specified address or PC number.

,exit_type

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the type of exit routine, based on how the system is to give the routine control. When the call specifies more than one exit routine, code the parameters as a 1-dimensional array.

If the *exit_entry* value is zero, the system ignores the *exit_type* value; you can specify binary zeros as the *exit_type* value.

The exit types are defined by each exit manager.

For **context services**, the exit types valid for PKM 0–7 or supervisor state resource managers are:

Table 28. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) exit_type parameter

Constant in: Hexadecimal (Decimal) Equate Symbol	**Description**
1 (1) CTX_EXIT_TYPE_SRB	**SRB:** The system schedules an SRB in the resource manager's address space to give control to the exit routine.
2 (2) CTX_EXIT_TYPE_PC	**Program call (PC):** The system issues a PC instruction to give control to the exit routine.
3 (3) CTX_EXIT_TYPE_PCS	**Program call (PC):** The system issues a PC instruction to give control to the exit routine which is defined through a PC number with a sequence number.

Note: Context services does not support any exit types for PKM 8–15 problem state resource managers.

For **RRS**, the exit types valid for PKM 0–7 or supervisor state resource managers are:

Table 29. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) exit_type parameter

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
1 (1) ATR_EXIT_TYPE_SRB	**SRB:** The system schedules an SRB in the resource manager's address space to give control to the exit routine.
2 (2) ATR_EXIT_TYPE_PC	**Program call (PC):** The system issues a PC instruction to give control to the exit routine.
3 (3) ATR_EXIT_TYPE_PCS	**Program call (PC):** The system issues a PC instruction to give control to the exit routine which is defined through a PC number with a sequence number.

Note: RRS does not support PKM 8–15 problem state resource managers.

,variable_data_1

Supplied/Returned parameter

- Type: Integer
- Length: 4 bytes

For Context Services, specifies the address of a storage area containing the name of the RRS resource manager that will assume ownership of privately-managed contexts when the address space owning the privately-managed contexts terminates. See “Private context delegation” on page 53 for a description of private context delegation to RRS.

The storage area has the following format:

- A 4–byte data area length
- A 4–byte data area version
- A 32–byte resource manager name. Currently, the only resource manager that supports private context delegation is RRS. The RRS resource manager name is: ATR.RESOURCEMANAGER.IBM

Note: The data area length includes the length and version fields, not just the RM name. For example, this parameter's length is 40 bytes.

When private context delegation is requested, privately-managed contexts are marked as needing private context delegation when they are created by the Begin_Context call.

For example:

1. A work manager requests private context delegation.
2. It creates a privately managed context.
3. It turns off private context delegation by calling Set_Exit_Information again, specifying hexadecimal zeros for variable_data_1.
4. It creates a new privately managed context.

Note: After this sequence of events, the privately-managed context from step 2 will still go through private context delegation if the work manager terminates.

The new privately managed context from step 4 on page 160 will **not** go through private context delegation if the work manager terminates.

For RRS, specifies the address of the prefix required if the resource manager is to issue a call to the Retrieve_Work_Identifier service and specify that the service is to generate an LUWID.

The prefix has the following format:

- A 1-byte hexadecimal integer that specifies the length of the prefix.
- The prefix for a unit of work identifier (UWID); the system uses the prefix as the LU name when generating an LUWID.

  The prefix of an LUWID has the following format:

  ```
  netid.luname
  ```

  where:

  netid.luname
  1-17 character identifier of the network and LU, preceded by a 1-byte length field

Exit managers other than Context Services and RRS expect *variable_data_1* to contain binary zeros. If you do not need the UWID prefix or private context delegation, specify binary zeros.

,variable_data_2
Supplied/Returned parameter

- Type: Integer
- Length: 4 bytes

If the exit manager does not expect any data, specify binary zeros in this parameter. If the exit manager is RRS (ATR.EXITMGR.IBM), *variable_data_2* has the following format:

Table 30. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) variable_data_2 parameter

Byte and Name	Meaning
0 ATR_EXIT_OPTION_FLAGS	Flags that control RRS exit processing
1 ATR_RESOURCE_MANAGER_OPTION _FLAGS	Flags that specify resource manager options to RRS
2–3 ATR_VARDATA_2_RSRVD_2_3	Reserved for IBM use. You must set these bytes to 0.

ATR_EXIT_OPTION_FLAGS has the following format:

Table 31. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) variable_data_2 parameter

Bit and Name	Meaning
0 ATR_EF_ON_LATER_WITH_ASYNC	RRS is to drive the resource manager's EXIT_FAILED exit when both of the following are true: • An exit responds (or has responded) ATRX_LATER. • The dispatchable unit that requested the syncpoint has abended asynchronously or was running in an address space that has been terminated.
1–7 ATR_EXIT_OPTS_RSRVD	Reserved for IBM use. You must set these bits to 0.

ATR_RESOURCE_MANAGER_OPTION_FLAGS has the following format:

Table 32. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) variable_data_2 parameter

Bit and name	Meaning
0 ATR_SUPPORTS_LOCAL_TRAN_MODE	Indicates that the resource manager supports local transaction mode; RRS can permit this resource manager to express interest in local transaction mode URs.
1 ATR_8K_RM_METADATA_REQUESTED	Indicates that the resource manager wants to be able to set and retrieve up to 8K of RM Metadata.
2–7 ATR_RM_OPTS_RSRVD	Reserved for IBM use. You must set these bits to 0.

If ATR_EF_ON_LATER_WITH_ASYNC is not set and an exit responds (or has responded) ATRX_LATER, RRS will not drive any exit routines but will wait for Post_Deferred_UR_Exit to supply the deferred response.

,variable_data_3
Supplied/Returned parameter

This parameter can be:

- Data in a 4-byte field supplied to the exit manager.
- A 4-byte field to receive character data from the exit manager.

The exit manager defines the data to be specified or received.

If the exit manager does not expect any data, specify binary zeros in this parameter. Nothing is returned in this parameter.

ABEND codes

The call might result in an abend X'AC7' with a reason code of X'00360000', X'00360001', or X'00360002'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Table 33. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) Return codes

Return Code in: Hexadecimal Equate Symbol	**Meaning and Action**
0 CRG_OK	**Meaning**: Successful completion. **Action**: None.
103 CRG_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 CRG_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CRG_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.
301 CRG_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is incorrect. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
305 CRG_SEIF_CURRENTLY_INVOKED	**Meaning**: Program error. The resource manager is currently setting exits with this exit manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Table 33. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) Return codes (continued)

Return Code in: Hexadecimal Equate Symbol	**Meaning and Action**
310 CRG_NOTIF_EXIT_TYPE_INV	**Meaning**: Program error. The notification exit type specified in the call is not valid, possibly because the resource manager is PKM 8–15 problem state. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
311 CRG_NOTIF_EXIT_ENTRY_INV	**Meaning**: Program error. The NOTIFICATION exit entry provided in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
320 CRG_EM_NAME_INV	**Meaning**: Program error. The exit manager name specified in the call is syntactically incorrect. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
340 CRG_EXIT_CNT_INV	**Meaning**: Program error. The exit count specified in the call exceeds the total number of exits for the exit manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
341 CRG_EXIT_NUM_INV	**Meaning**: Program error. The exit number specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
342 CRG_EXIT_TYPE_INV	**Meaning**: Program error. The exit type specified in the call is not valid, possibly because the resource manager is PKM 8–15 problem state. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.

Table 33. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) Return codes (continued)

Return Code in: Hexadecimal Equate Symbol	**Meaning and Action**
343 CRG_VAR1_INV	**Meaning**: Program error. The data in the *variable_data_1* parameter is not valid for this exit manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
344 CRG_VAR2_INV	**Meaning**: Program error. The data in the *variable_data_2* parameter is not valid for this exit manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
345 CRG_VAR3_INV	**Meaning**: Program error. The data in the *variable_data_3* parameter is not valid for this exit manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
346 CRG_REQ_EXIT_NOT_SET	**Meaning**: Program error. The call failed to specify an exit routine that is required by the exit manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
347 CRG_DELEXIT_INV	**Meaning**: Program error. The call attempted to delete an exit routine that is required by the exit manager by specifying zero for the exit routine in the *exit_entry* parameter. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
348 CRG_DUP_EXIT_SET	**Meaning**: Program error. The call tried to set a duplicate exit; that is, the resource manager has already set, with this exit manager, an exit with the specified number. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Table 33. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) Return codes (continued)

Return Code in: Hexadecimal Equate Symbol	**Meaning and Action**
349 CRG_EXIT_TYPE_SRV	**Meaning**: Program error. The exit type specified is valid for this exit manager but is not valid for this exit. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
34A CRG_EXIT_ENTRY_INV	**Meaning**: Program error. The *exit_entry* value specified on the call is not valid for this exit manager. You might, for example, have specified zero when the corresponding *exit_number* specifies a value. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
720 CRG_EM_STATE_ERROR	**Meaning**: Environmental error. The exit manager specified in the call is not registered as an exit manager. The system rejects the service call. **Action**: When the exit manager registers, the system gives control to a notification exit routine, if specified in the call. The notification exit routine can issue the set exit routine call again. **Note:** The exit manager might never register.
756 CRG_AUTH_FAILURE	**Meaning**: Program error. The resource manager is PKM 8–15 problem state and specified a resource manager token that does not belong to a PKM 8–15 problem state resource manager registered in the home address space. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
758 CRG_EM_FAILED_RM_AUTH	**Meaning**: Program error. The exit manager specified in the call does not support PKM 8–15 problem state resource managers. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.

Table 33. Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF) Return codes (continued)

Return Code in: **Hexadecimal** **Equate Symbol**	**Meaning and Action**
FFF CRG_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.
1000 - FFFF	**Meaning**: Additional returns codes that the specific exit manager can issue. **Action**: See the information about the exit manager. The return codes that RRS issues are defined in “Installing an exit routine” on page 91. The return codes that Context Services issues are defined in “Installing an exit routine” on page 36.

Example

In the pseudocode example, the resource manager issues a call to set its exit routines with context services. Because context services does not require any variable data, the call has null variable data parameters. Because context services is always available, the call does not specify a notification exit routine.

```
⋮
RM_TOKEN = MY_RM_TOKEN
EM_NAME = CTXSER_NAME
EXIT_CNT = 2
EXIT_NUM(1) = CTX_END_CONTEXT_EXIT
EXIT_ADDR(1) = ADDR(END_EXIT_PROC)
EXIT_TYPE(1) = CRG_EXIT_TYPE_SRB
EXIT_NUM(2) = CTX_SWITCH_EXIT
EXIT_ADDR(2) = ADDR(SWITCH_PROC)
EXIT_TYPE(2) = CRG_EXIT_TYPE_SRB
CALL CRGSEIF(RC,RM_TOKEN,CRG_EXIT_TYPE_NONE,CRG_NULL_PARAMETER,
             EM_NAME,EXIT_CNT,EXIT_NUM,EXIT_ADDR,EXIT_TYPE,
             CRG_NULL_PARAMETER,CRG_NULL_PARAMETER,
             CRG_NULL_PARAMETER)
⋮
```

Unregister_Resource_Manager (CRGDRM, CRG4DRM)

- CRGDRM is for AMODE(31) callers.
- CRG4DRM is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Unregister_Resource_Manager service to unregister itself explicitly. In response to the call, the system returns a return code.

Explicit and Implicit Unregistration: Normally, when a resource manager is ending processing, it issues a call to unregister itself. The call can be issued from any address space. If your resource manager does not explicitly unregister, the

system implicitly unregisters it as follows, depending on the *unregister_option* specified in the call to the Register_Resource_Manager service that registered the resource manager:

- When the resource manager's task ends. The resource manager runs as a task in the home address space.
- When the cross memory resource-owning task of the resource manager ends. This task is the top, or first, job step task in the home address space.
- When the resource manager's address space ends.

The system can also unregister a resource manager because of errors, such as consecutive exit errors.

Environment

The requirements for the resource manager are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (CRGDRM) 64 bit (CRG4DRM)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.
Linkage:	Uses standard MVS linkage conventions.

Programming requirements

The resource manager does not have to issue the call to the Unregister_Resource_Manager service from the same task and address space in which it issued the corresponding call to the Register_Resource_Manager service.

Either link edit your object code with the linkable stub routine CRGCSS (31 bit) or CRG4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL Definition	Description
CRGASM	390 Assembler declarations
CRGC	C/390 declarations
FOMUCRGC	z/OS HFS header files

Restrictions

The state of the resource manager associated with the resource manager token specified must be **registered**, **set**, **reset**, or **run**. After a successful call, the resource manager state is **unregistered**.

Resource managers that are PKM 8–15 problem state must register using the Register_Resource_Manager service from the home address space before invoking this service. They must specify a resource manager token of a key 8–15 problem state resource manager that registered from the home address space.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CRGDRM	(return_code ,resource_manager_token)

CALL CRG4DRM	(return_code ,resource_manager_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Unregister_Resource_Manager service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that uniquely identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00310000' or X'00310001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 CRG_OK	**Meaning**: Successful completion. **Action**: None.
103 CRG_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 CRG_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
107 CRG_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.
301 CRG_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not one of the currently valid resource manager tokens. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
756 CRG_AUTH_FAILURE	**Meaning**: Program error. The resource manager is PKM 8–15 problem state and specified a resource manager token that does not belong to a PKM 8–15 problem state resource manager registered in the home address space. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
FFF CRG_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, a resource manager issues a call to unregister itself.

```
⋮
RM_TOKEN = MY_RM_TOKEN
CALL CRGDRM(RC,RM_TOKEN)
⋮
```

Chapter 6. Callable context services

This section describes the callable services that an authorized resource manager can use to request work context services. The chapter presents the callable services in alphabetical order by descriptive name.

Begin_Context (CTXBEGC, CTX4BEGC)

- CTXBEGC is for AMODE(31) callers
- CTX4BEGC is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Begin_Context service to create a privately-managed context. Begin_Context is intended for use in a program that manages work on behalf of another program or user. The program accepts the responsibility to manage the environment for the other program.

In response to the call, context services returns:

- A return code.
- The context token for the privately-managed context. You need this token for a call to the following services: End_Context, Express_Context_Interest, Express_UR_Interest, Retrieve_Interest_Count, or Switch_Context.

Contexts: A context represents the resources for a work request; a context consists of the application program requesting the work and the protected resources involved in the work. The two types of contexts are:

- Native context
- Privately-managed context

An application's task has a native context associated with it. A resource manager can use a call to the Begin_Context service to obtain a privately-managed context, then use a call to the Switch_Context service to associate the privately-managed context with a task. While the privately-managed context is associated with a task, interactions with the application are related to the privately-managed context.

Later, the resource manager can use a call to the Switch_Context service to disassociate the privately-managed context; subsequent interactions are related to the native context for the task.

Current context: The native context is the original current context for an application's task. A Begin_Context call obtains a privately-managed context, and a call to Switch_Context associates the privately-managed context with the application; the native context still exists but is not current. The privately-managed context is the current context. If a call to the Switch_Context service later disassociates the privately-managed context, the native context again becomes the current context.

Context token: The context token is a random value that is not preserved across restarts of the system, exit manager, or resource manager. Thus:

- Do not use the context token as an identifier in log records.

- Do not try to discern the contents of the token or create any dependencies on the contents.

Environment

The requirements for the resource manager are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (CTXBEGC) 64 bit (CTX4BEGC)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager

Programming requirements

The resource manager's object code must be linked with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

To call the service, the resource manager associated with the resource manager token specified in the call must be in **set** state, which means it has registered and called the Set_Exit_Information service, specifying context services as the exit manager.

Resource managers that are PKM 8–15 problem state must register using the Register_Resource_Manager service from the home address space before invoking this service. They must specify a resource manager token of a key 8–15 problem state resource manager which registered from the home address space.

If a PKM 8–15 problem state resource manager attempts to create a context and doing so will result in the PKM 8–15 problem state resource manager registered in the space owning more than 256 contexts per unauthorized resource manager, context services will request confirmation of the request from a system operator. If the operator allows the request, the PKM 8–15 problem state resource managers registered in the space will be able to create as many contexts as they want. If the operator does not allow the request, a context will not be returned.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CTXBEGC	(return_code ,resource_manager_token ,context_token)

CALL CTX4BEGC	(return_code ,resource_manager_token ,context_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Begin_Context service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

,context_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives the token for the privately-managed context that the resource manager is creating. The context token uniquely identifies the privately-managed context.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00110000' or X'00110001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 CTX_OK	**Meaning**: Successful completion. **Action**: None.
103 CTX_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports context services. Then rerun the resource manager.
301 CTX_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 CTX_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the resource manager token specified in the call is not in a valid state to issue the service call. The resource manager must be in set state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
756 CTX_AUTH_FAILURE	**Meaning**: Program error. The resource manager is PKM 8–15 problem state and specified a resource manager token that does not belong to a PKM 8–15 problem state resource manager registered in the home address space. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
F00 CTX_MAX_CTXT_EXCEEDED	**Meaning**: Environment error. The resource manager is PKM 8–15 problem state and attempted to create more than the allowable number of active contexts. The system rejects the service call. **Action**: Either allow unauthorized resource managers to own additional contexts or change your program so less contexts are required.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to create a private context. Storage for the call parameters has been allocated.

```
⋮
RM_TOKEN = REG_TOKEN
CALL CTXBEGC(RC,RM_TOKEN,C_TOKEN)
IF RC ≠ CTX_OK THEN
    /* handle error */
⋮
```

Delete_Context_Interest (CTXDINT, CTX4DINT)

- CTXDINT is for AMODE(31) callers.
- CTX4DINT is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Delete_Context_Interest service to delete its interest in a native context or a privately-managed context. In response to the call, context services issues a return code.

Note: If your resource manager does not issue a Delete_Context_Interest call, the system deletes the context interest when the context ends.

Environment

The requirements for the resource manager are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (CTXDINT) 64 bit (CTX4DINT)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.

Programming requirements

Either link edit the resource manager's object code with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

To call the service, the resource manager associated with the context interest token specified in the call must be in **set** state, which means it has registered and called the Set_Exit_Information service, specifying context services as the exit manager.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CTXDINT	(return_code ,context_interest_token)

CALL CTX4DINT	(return_code ,context_interest_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Delete_Context_Interest service.

,context_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the context interest token that identifies the context interest to be deleted. Your resource manager received the token from the Express_Context_Interest service.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00120000' or X'00120001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 CTX_OK	**Meaning**: Successful completion. **Action**: None.
103 CTX_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports context services. Then rerun the resource manager.
365 CTX_CI_TOKEN_INV	**Meaning**: Program error. The context interest token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to delete an interest in a context. Storage for the call parameters has been allocated.

```
⋮
CI_TOKEN = CONTEXT_INTEREST_TOKEN
CALL CTXDINT(RC,CI_TOKEN)
IF RC ≠ CTX_OK THEN
```

```
    /* handle error */
⋮
```

End_Context (CTXENDC, CTX4ENDC)

- CTXENDC is for AMODE(31) callers.
- CTX4ENDC is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the End_Context service to end a privately-managed context or a dispatchable unit native context. A native context has a fixed association with a single dispatchable unit.

When an application program ends processing, your resource manager should call the End_Context service to end any context associated with the application. RRS default actions are to commit on normal context termination and backout on abnormal context termination.

In response to the call, the system issues a return code. After the call, the context token associated with the ended context is no longer valid.

Next current context: A call to the End_Context service ends the context specified in the *context_token* parameter. If the call specifies a privately-managed context, the native context becomes the current context. If the call specifies a native context, which is also the current context, a new native context becomes the current context.

Ending a privately-managed context: If a call to the End_Context service specifies a privately-managed context not associated with a unit of work, the system gives control to the CONTEXT_SWITCH exit routines of all resource managers interested in the context. If any CONTEXT_SWITCH exit routine disallows the context end, the call does not end the context. However, CONTEXT_SWITCH exit routines cannot stop the context end if one of the following is true:

- The address space of the resource manager that owns the privately-managed context is terminating.
- The End_Context service forces the ending.

The End_Context service might fail because a CONTEXT_SWITCH exit routine disallows ending the context. To override the decision when there is no other way to resolve the problem, the privately-managed context owner can force an end to the context by specifying a *completion_type* of CTX_FORCED_END_OF_CONTEXT on the End_Context call.

After the CONTEXT_SWITCH exit routines have completed, if the privately-managed context is still ending, the system invokes the END_CONTEXT exit routines associated with the context.

Environment

The requirements for the resource manager are:

Minimum authorization: None
Dispatchable unit mode: Task
Cross memory mode: Any PASN, any HASN, any SASN

AMODE:	
	31 bit (CTXENDC)
	64 bit (CTX4ENDC)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.

Programming requirements

Either link edit the resource manager's object code with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

To call the service, the resource manager associated with the context specified in the call must be in **set** state, which means it has registered and called the Set_Exit_Information service, specifying context services as the exit manager.

When calling End_Context to end a native context:

- The context must be the current context.
- Your resource manager must be running under the work unit associated with the native context.

When calling End_Context to end a privately-managed context associated with the work unit:

- Your resource manager must be running under the work unit associated with the privately-managed context.

If you are coding an RRS exit routine, do not call this service to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter.

Resource managers that are PKM 8–15 problem state must register using the Register_Resource_Manager service from the home address space before invoking this service. They can only end native contexts and privately managed contexts obtained by a key 8–15 problem state resource manager that registered from the home address space.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CTXENDC	(return_code ,context_token ,completion_type)

CALL CTX4ENDC	(return_code ,context_token ,completion_type)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the End_Context service.

,context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token for the context that the resource manager wants to end, as follows:

- 0: Binary zeros specify the current context associated with the application's task or SRB.
- token: The context token of a privately-managed context.

For a privately-managed context, your resource manager received the *context_token* from the Begin_Context service.

,completion_type
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates the type of completion for the context. The value is passed to the END_CONTEXT exit routine. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	**Resource manager specifies the *completion_type* because:**
0 (0) CTX_NORMAL_TERMINATION	The context is ending normally.
1 (1) CTX_ABNORMAL_TERMINATION	The context is ending abnormally.
3 (3) CTX_FORCED_END_OF_CONTEXT	The context must be forced to end. A SWITCH_CONTEXT exit routine cannot prevent the context from ending.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00140000' or X'00140001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 CTX_OK	**Meaning**: Successful completion. **Action**: None.
103 CTX_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
104 CTX_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports context services. Then rerun the resource manager.
360 CTX_COMPLETION_TYPE_INV	**Meaning**: Program error. The *completion_type* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
361 CTX_CONTEXT_TOKEN_INV	**Meaning**: Program error. The context token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
363 CTX_OTHER_WU_NATIVE	**Meaning**: Program error. The native context specified in the call is the native context of another unit of work. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
366 CTX_PRIVATE_OTHER_WU	**Meaning**: Program error. The privately-managed context specified in the call is the current context of another work unit. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
369 CTX_SWITCH_EXIT_PREVENTED_END	**Meaning**: Program error. A CONTEXT_SWITCH exit routine returned a code that indicated the context should not be ended. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
756 CTX_AUTH_FAILURE	**Meaning**: Program error. The resource manager is PKM 8–15 problem state and specified a context token that does not belong to a PKM 8–15 problem state resource manager registered in the home address space. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to end a context. Storage for the call parameters has been allocated.

```
⋮
COMPLETION_TYPE = CTX_NORMAL_TERMINATION
C_TOKEN = CONTEXT_1
CALL CTXENDC(RC,C_TOKEN,COMPETION_TYPE)
IF RC ≠ CTX_OK THEN
```

```
    /* Handle error */
⋮
```

Express_Context_Interest (CTXEINT, CTXEINT1, CTX4EINT)

A resource manager calls the Express_Context_Interest service to express an interest in a privately-managed context or a dispatchable unit native context. A native context has a fixed association with a single dispatchable unit. There are three versions of Express_Context_Interest, each with different parameters.

- CTXEINT is for AMODE(31) callers and is the basic version of the service.
- CTXEINT1 is for AMODE(31) callers and adds work manager name support.
- CTX4EINTis for AMODE(64) callers, allows parameters in 64 bit addressable storage, and includes work manager name support.

Code your resource manager to call the version that includes the support you need.

In response to the call, context services returns:

- A return code.
- The context token of the current context, if requested.
- The context interest token. You need the context interest token for calls to the following services: Delete_Context_Interest, Retrieve_Context_Interest_Data, or Set_Context_Interest_Data.
- For CTXEINT1 and CTX4EINT callers, the work manager name of the resource manager that owns the context associated with the expression of interest.

If your resource manager already has an interest in the context, a call to the Express_Context_Interest service can do one of the following, depending on the *multiple_interest_option* parameter you supply:

- Return the context interest token and the context interest data for the existing interest

  If your resource manager already has several interests in a context, there is no way to predict which one the service will return.
- Create a new interest in the context and provide a new context interest token for the new interest

Expressing interest: Expressing interest in a context tells the system to invoke your resource manager's exit routines for this context interest. A resource manager can express interest in **any** context in **any** address space. A resource manager can make the call multiple times to create multiple context interests.

Expressing interest in a context has no connection with expressing interest in a unit of recovery (UR).

Context interest data: In the call, your resource manager provides context interest data. The system passes this data to your resource manager's exit routines invoked for this context interest. This data can contain an anchor for the resource manager's data structures for the context. Your resource manager can issue:

- A call to the Retrieve_Context_Interest_Data service
- A call to the Set_Context_Interest_Data service to specify the data, if it is not specified in the Express_Context_Interest call
- One or more calls to the Set_Context_Interest_Data service to change this data

Context end: The context abnormally ends if the application program abnormally ends processing or if the application's address space abnormally ends. Other conditions that can abnormally end a context are:

- The End_Context service specifies an abnormal condition.
- The owner of a disassociated privately-managed context ends.

Depending on how the context ends and on the *memterm_option* parameter in the Express_Context_Interest call, the system might give control to your resource manager's END_CONTEXT exit routine.

Environment

The requirements for the resource manager are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (CTXEINT, CTXEINT1)
	64 bit (CTX4EINT)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.

Programming requirements

Either link edit the resource manager's object code with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL Definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

To call the service, the resource manager associated with the context specified in the call must be in **set** state, which means it has registered and called the set exit routine service, specifying context services as the exit manager. must be in **run** state.

When the resource manager issues the call in SRB mode, the call cannot specify a *context_token* of 0, indicating the current context.

If you are coding an RRS exit routine, do not call this service to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the appropriate call as shown in the syntax diagrams. You must code the parameters in the CALL statement as shown.

CALL CTXEINT	(return_code ,resource_manager_token ,context_token ,memterm_option ,context_interest_data ,current_context_token ,context_interest_token ,returned_context_interest_data ,multiple_interest_option)

CALL CTXEINT1	(return_code ,resource_manager_token ,context_token ,memterm_option ,context_interest_data ,current_context_token ,context_interest_token ,returned_context_interest_data ,multiple_interest_option ,work_manager_name)

CALL CTX4EINT	(return_code ,resource_manager_token ,context_token ,memterm_option ,context_interest_data ,current_context_token ,context_interest_token ,returned_context_interest_data ,multiple_interest_option ,work_manager_name)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Express_Context_Interest service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

,context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token for the context in which the resource manager is expressing an interest, as follows:

- 0: Binary zeros specify the current context associated with the application's task.
- token: The context token of a privately-managed context.

For a privately-managed context, your resource manager received the *context_token* from the Begin_Context service.

,memterm_option

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates whether or not the resource manager's END_CONTEXT exit routine should receive control if the context abnormally ends. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) CTX_ALL_TERMINATIONS	**All endings:** The END_CONTEXT exit routine receives control at all endings, including memory termination.
1 (1) CTX_NOT_MEMTERM	**All endings, except memory termination:** The END_CONTEXT exit routine receives control at all endings except memory termination.

,context_interest_data

Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the context interest data that the system is to associate with this context interest. The Retrieve_Context_Interest_Data service can retrieve the context interest data.

,current_context_token

Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives the following from the service:

- The token of the current context, if the call specifies zeros in the *context_token* parameter. The token is a 16-byte character string.
- Undefined, if *context_token* specifies a token.

,context_interest_token

Returned parameter

- Type: Character string
- Character Set: No restriction

- Length: 16 bytes

Receives the context interest token from the service. The context interest token uniquely identifies your resource manager's interest in the context. If you specified CTX_CONDITIONAL on *multiple_interest_option*, the context interest token represents an existing interest, if there is one. Otherwise, the context interest token represents the newly created interest.

,returned_context_interest_data

Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

If *multiple_interest_option* specified CTX_CONDITIONAL and the return code from the service is CTX_RM_ALREADY_HAS_INTEREST, this field receives the context interest data from the service. The data comes from an already existing interest that the resource manager has in the context. If the resource manager does not have an existing interest, the service returns binary zeros.

Otherwise, this field is undefined.

,multiple_interest_option

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates whether or not the service is to create a new interest when the resource manager already has an interest in the context. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) CTX_UNCONDITIONAL	**Unconditional:** The service should create a new interest, even when the resource manager already has an interest in the context.
1 (1) CTX_CONDITIONAL	**Conditional:** The service should **not** create a new interest when the resource manager already has an interest in the context.

When you specify CTX_CONDITIONAL and the resource manager has an existing interest in the specified context, the values in *memterm_option* and *context_interest_data* are ignored.

,work_manager_name

Returned parameter

- Type: Character string
- Character Set: See Note
- Length: 32 bytes

For CTXEINT1 callers, this field receives the work manager name from the service. The work manager name is the 32–byte name of the resource manager that owns the privately-managed context this expression of interest pertains to.

If the expression of interest is for a dispatchable unit native context, the work manager name returned is a concatenation of the following strings:

- SystemName
- Period (.)
- JobName
- Period (.)
- ASID (4 bytes readable hexadecimal)
- Blanks (padded to 32 bytes)

Note: The work manager name can consist of the following printable characters:

- Alphanumeric characters: A–Z and 0–9
- National characters: $ (X'5B'), # (X'7B'), and @ (X'7C')
- The period (.)
- The underscore (_)
- The trailing blank characters needed to fill the 32–byte field

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00130000' or X'00130001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 CTX_OK	**Meaning**: Successful completion. **Action**: None.
8 CTX_RM_ALREADY_HAS_INTEREST	**Meaning**: The resource manager already has an expression of interest in this context. The system returns the *context_interest_token* and *returned_context_interest_data* for an existing expression of interest in the context by the resource manager. **Action**: Process the returned information.
103 CTX_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
104 CTX_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.
301 CTX_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
361 CTX_CONTEXT_TOKEN_INV	**Meaning**: Program error. The context token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
364 CTX_MEMTERM_INV	**Meaning**: Program error. The *memterm_option* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
367 CTX_MULTIPLE_INTEREST_ OPTION_INV	**Meaning**: Program error. The *multiple_interest_option* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
36A CTX_DU_TERMINATING	**Meaning**: Environmental error. The application's task or SRB associated with the specified context is abnormally ending. The system rejects the service call. **Action**: None.
701 CTX_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the context specified in the call is not in a valid state to issue the service call. The resource manager must be in set state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to express an interest in a context. Storage for the call parameters has been allocated.

```
⋮
RM_TOKEN = REG_TOKEN
C_TOKEN = CONTEXT_1
MEMTERM_OPT = CTX_ALL_TERMINATIONS
CI_DATA = CONTEXT_1_DATA
CALL CTXEINT1(RC,RM_TOKEN,C_TOKEN,MEMTERM_OPT,CI_DATA,
            CUR_C_TOKEN,CI_TOKEN,
            RETURNED_CONTEXT_INTEREST_DATA,
            MULTIPLE_INTEREST_OPTION,
            WORK_MANAGER_NAME)
IF RC = CTX_OK THEN
  DO
   CONTEXT_INTEREST_TOKEN = CI_TOKEN
   MYWMNAME=WORK_MANAGER_NAME
  END DO
⋮
```

Retrieve_Context_Data (CTXRDTA, CTX4RDTA)

- CTXRDTA is for AMODE(31) callers.
- CTX4RDTA is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Retrieve_Context_Data service to retrieve the data associated with a particular context. The data must have previously been set by a call to Set_Context_Data. The resource manager specifies a key which identifies the data that is to be retrieved.

Environment

The requirements for the resource manager are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (CTXRDTA) 64 bit (CTX4RDTA)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Programming requirements

Either link edit the resource manager's object code with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL Definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

The caller must be in Task mode when invoking Retrieve_Context_Data for the current dispatchable unit's context.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CTXRDTA	(return_code ,context_token ,context_key ,context_bufferlength ,context_datalength ,context_data_buffer)

CALL CTX4RDTA	(return_code ,context_token ,context_key ,context_bufferlength ,context_datalength ,context_data_buffer)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Provides the return code for the call.

,context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

The context_token identifies the context with which the data is associated. If a value of "binary zeros" is supplied in this field, then the context will be the currently active context of this dispatchable unit of work. A context token may be obtained via the Begin_Context, Express_Context_Interest, or Retrieve_Current_Context_Token services.

,context_key
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 32 bytes

The context_key is the identifier which was supplied on the earlier Set_Context_Data, and identifies the data which is to be retrieved.

If context_key is set to CTX.OWNER_INFO.IBM, the type and authorization of the context specified by the context_token will be returned in the context_data_buffer.

,context_bufferlength
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

The context_bufferlength is the length of the area specified by the context_data_buffer keyword for the return of the data. The minimum length is 1. If the length of the buffer is less than the length of the saved data, then only as much of the data will be returned as will fit in the buffer. In this case, the actual length of the data will be returned in the context_datalength keyword.

,context_datalength
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

The context_datalength is the actual length of the data specified by the context_data_buffer keyword. A value of zero indicates that no data was returned. This value may be larger than context_bufferlength if the return code is CTX_PARTIAL_DATA.

,context_data_buffer
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 1 to 4096 bytes

The context_data_buffer is the area to which the saved data will be returned.

If context_key is set to CTX.OWNER_INFO.IBM, the following 6–word data string will be returned:

Word	Contents
0	**0**: The target work context is a DU native context. **1**: The target work context is a privately managed context.
1	**0**: The target work context is owned by an authorized resource manager. **1**: The target work context is owned by an unauthorized resource manager.
2–5	These words are reserved for future use. They contain random contents.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00210000' or X'00210001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 CTX_OK	**Meaning**: Successful completion. **Note:** If there is no data associated with the specified key, the context datalength returned will be `0`. **Action**: None.
5 CTX_PARTIAL_DATA	**Meaning**: Program error. Partial data was returned. The buffer is not long enough to hold all of the data. The service completes successfully, but returns only partial data. The actual length of the data associated with the input context_key is returned in context_datalength. **Action**: Check program logic for probable coding error. Correct the problem and reissue service.
103 CTX_INTERRUPT_INV	**Meaning**: Program error. The caller is disabled. The system rejects the service call. **Action**: Check program logic for probable coding error. Correct the problem and reissue service.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
104 CTX_MODE_INV	**Meaning**: Program error. The caller is not in task mode and specified 0 for context_token. The system rejects the service call. **Action**: Check program logic for probable coding error. Correct the problem and reissue service.
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks may be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: The release of MVS does not support this service. The service stub has been linked on a system that does not support the correct level of Context Services. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports the correct level of Context Services. Then rerun the resource manager.
361 CTX_CONTEXT_TOKEN_INV	**Meaning**: Program error. The context interest token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
36D CTX_BUFFER_LENGTH_INV	**Meaning**: Program error. The Buffer length specified via the CTXRDTA invocation is invalid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to retrieve the data associated with the key `FRED` for the context associated with the current dispatchable unit.

```
⋮
C_TOKEN = ''B;
DATA_KEY = 'FRED';
DATA_LEN = 0;
BUFFER_LEN = 20;
DATA = '';
CALL CTXRDTA(RC,C_TOKEN,DATA_KEY,BUFFER_LEN,DATA_LEN,DATA);
IF RC ¬= CTX_OK THEN
    /* handle error situation */
⋮
IF DATA_LEN = 0 THEN
    /* handle no data associated with FRED */
⋮
```

Retrieve_Context_Interest_Data (CTXRCID, CTX4RCID)

- CTXRCID is for AMODE(31) callers.
- CTX4RCID is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Retrieve_Context_Interest_Data service to retrieve the context interest data supplied by the resource manager in:

- A call to the Express_Context_Interest service
- A call to the Set_Context_Interest_Data service

In response to the call, context services also returns a return code.

A resource manager can express interest in a context multiple times. The particular interest for this call is identified by the context interest token your resource manager received from the Express_Context_Interest service.

Environment

The requirements for the resource manager are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (CTXRCID) 64 bit (CTX4RCID)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.

Programming requirements

Either link edit the resource manager's object code with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

To call the service, the resource manager associated with the context interest token specified in the call must be in **set** state, which means it has registered and called the Set_Exit_Information service, specifying context services as the exit manager.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CTXRCID	(return_code ,context_interest_token ,context_interest_data)

CALL CTX4RCID	(return_code ,context_interest_token ,context_interest_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Retrieve_Context_Interest_Data service.

,context_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the context interest token that identifies your resource manager's interest in the context. Your resource manager received the token from the Express_Context_Interest service.

,context_interest_data
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives the context interest data associated with this context interest. The resource manager supplied the data on the most recent call to the Express_Context_Interest or Set_Context_Interest_Data service. If no earlier call has supplied context interest data, the field contains hexadecimal zeros.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00160000' or X'00160001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 CTX_OK	**Meaning**: Successful completion. **Action**: None.
103 CTX_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.
365 CTX_CI_TOKEN_INV	**Meaning**: Program error. The context interest token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to retrieve context interest data. Storage for the call parameters has been allocated.

```
⋮
CI_TOKEN = CONTEXT_INTEREST_TOKEN
CALL CTXRCID(RC,CI_TOKEN,CI_DATA)
IF RC = CTX_OK THEN
```

```
  CONTEXT_I_DATA = CI_DATA
⋮
```

Retrieve_Current_Context_Token (CTXRCC, CTX4RCC)

- CTXRCC is for AMODE(31) callers.
- CTX4RCC is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Retrieve_Current_Context_Token service to obtain the context token for the currently active context. The token can be used in subsequent calls to other context services to identify that context.

Note: The Retrieve_Current_Context_Token service is intended to be used to obtain the context token of the active context of the current dispatchable unit of work.

Environment

The requirements for the resource manager are:

Minimum authorization:	None
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (CTXRCC) 64 bit (CTX4RCC)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.

Programming requirements

Either link edit the resource manager's object code with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL Definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

None.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CTXRCC	(return_code ,context_token)

CALL CTX4RCC	(return_code ,context_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A

- Length: 4 bytes

Provides the return code for the call.

,context_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

The context_token identifies the context that is currently active.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00220000' or X'00220001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 CTX_OK	**Meaning**: Successful completion. **Action**: None.
103 CTX_INTERRUPT_INV	**Meaning**: Program error. The caller is disabled. The system rejects the service call. **Action**: Check program logic for probable coding error. Correct the problem and reissue service.
104 CTX_MODE_INV	**Meaning**: Program error. The caller is not in task mode. The system rejects the service call. **Action**: Check program logic for probable coding error. Correct the problem and reissue service.
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: The release of MVS does not support this service. The service stub has been linked on a system that does not support the correct level of Context Services. **Action**: Remove the resource manager from the system, and install it on a system that supports the correct level of Context Services. Then rerun the resource manager.
36A CTX_DU_TERMINATING	**Meaning**: Program error. The dispatchable unit associated with the specified context is terminating. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to retrieve the context token of the current context. Storage for the call parameters has been allocated.

```
⋮
C_TOKEN = ''B;
CALL CTXRCC(RC,C_TOKEN);
IF RC ¬= CTX_OK THEN
    /* handle error situation */
⋮
```

Set_Context_Data (CTXSDTA, CTX4SDTA)

- CTXSDTA is for AMODE(31) callers.
- CTX4SDTA is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Set_Context_Data service to save data to be associated with a specific context and to identify it with a specified key. This key can be used to identify the data in subsequent calls to Set_Context_Data, or to the Retrieve_Context_Data service. In response to the call, context services issues a return code.

Note:

1. The Set_Context_Data service can be used to change or delete the data specified on a previous invocation of Set_Context_Data. The data may be retrieved via the Retrieve_Context_Data service.
2. The data set by calling Set_Context_Data can only be set by a program running with a PKM of 0–7 or in supervisor state; however, the data can be retrieved by any program.

Environment

The requirements for the resource manager are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (CTXSDTA) 64 bit (CTX4SDTA)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.

Programming requirements

Either link edit the resource manager's object code with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

The caller must be in Task mode when invoking Set_Context_Data for the current dispatchable unit's context.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system

15 Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CTXSDTA	(return_code ,context_token ,context_key ,context_datalength ,context_data)

CALL CTX4SDTA	(return_code ,context_token ,context_key ,context_datalength ,context_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Set_Context_Data service.

,context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

The context_token identifies the context with which the data is to be associated. If a value of "binary zeros" is supplied in this field, then the context will be the currently active context of this dispatchable unit of work. A context token may be obtained via the Begin_Context, Express_Context_Interest, or Retrieve_Current_Context_Token services.

,context_key
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 32 bytes

The context_key is an identifier used to identify the data to be saved, and by which the data can be changed or deleted by later calls to Set_Context_Data, or retrieved by a call to Retrieve_Context_Data.

You may code almost any key to identify data. However, a specific key, CTX.OWNER_INFO.IBM, is reserved by IBM to return special data. You may not set context_key to CTX.OWNER_INFO.IBM with the Set_Context_Data service.

,context_datalength
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

The context_datalength is the length of the data specified by the context_data keyword. The maximum value is 4096 bytes of data. If a length of zero is specified, the previously saved data identified by context_datakey will be deleted.

,context_data
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 1 to 4096 bytes

The context_data is the data to be saved. The contents of this parameter are ignored if the context_datalength is 0.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00200000' or X'00200001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 CTX_OK	**Meaning**: Successful completion. **Action**: None.
103 CTX_INTERRUPT_INV	**Meaning**: Program error. The caller is disabled. The system rejects the service call. **Action**: Check program logic for probable coding error. Correct the problem and reissue service.
104 CTX_MODE_INV	**Meaning**: Program error. The caller is not in task mode. The system rejects the service call. **Action**: Check program logic for probable coding error. Correct the problem and reissue service.
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks may be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: Environment error. The release of MVS does not support this service. The service stub has been linked on a system that does not support the correct level of Context Services. **Action**: Remove the resource manager from the system, and install it on a system that supports the correct level of Context Services. Then rerun the resource manager.
309 CTX_RESERVED_NAME	**Meaning**: Program error. The value specified in context_key is reserved for Context Services. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
361 CTX_CONTEXT_TOKEN_INV	**Meaning**: Program error. The context interest token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
36B CTX_DATA_LENGTH_INV	**Meaning**: Program error. The data length specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
36C CTX_DATA_KEY_NOTFOUND	**Meaning**: Program error. The data key value specified in the call is not found. The system rejects the service call. This code is only returned when the specified data length is 0. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
36E CTX_STORAGE_UNAVAILABLE	**Meaning**: Environment error. There is no storage available from the necessary subpool to save the data. The system rejects the service call. **Action**: Determine why the system ran out of subpool 247. Correct the problem and reissue service.
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to associate data for the key FRED with the current context. Storage for the call has been allocated.

```
⋮
C_TOKEN = ''B;
DATA_KEY = 'FRED';
DATA_LEN = 6;
DATA = 'MYDATA';
CALL CTXSDTA(RC,C_TOKEN,DATA_KEY,DATA_LEN,DATA);
IF RC ¬= CTX_OK THEN
    /* handle error situation */
⋮
```

Set_Context_Interest_Data (CTXSCID, CTXSCID2, CTX4SCID)

A resource manager calls the Set_Context_Interest_Data service to provide or change context interest data. The system passes the current context interest data to the resource manager's exit routines associated with the context interest. Your resource manager can issue:

- A call to the Set_Context_Interest_Data service to specify the data, if it is not specified in a call to the Express_Context_Interest service

- One or more calls to the Set_Context_Interest_Data service to change this data
- A call to the Retrieve_Context_Interest_Data service to retrieve the data

There are three versions of Set_Context_Interest_Data, each with different parameters.

- CTXSCID is for AMODE(31) callers and is the basic version of the service.
- CTXSCID2 is for AMODE(31) callers and adds current context interest data support.
- CTX4SCID is for AMODE(64) callers, allows parameters in 64 bit addressable storage, and adds current context interest data support.

Code your resource manager to call the version that includes the support you need.

For CTXSCID2 and CTX4SCID callers, to enable serialized update of the context interest data, you must provide the expected current value of the context interest data when you make the call to Set_Context_Interest_Data. If the provided data does not match the real data, the set request will fail and the actual current data will be returned.

Environment

The requirements for the resource manager are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (CTXSCID, CTXSCID2) 64 bit (CTX4SCID)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.

Programming requirements

Either link edit the resource manager's object code with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL Definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

To call the service, the resource manager associated with the context interest token specified in the call must be in **set** state, which means it has registered and called the set exit routine service, specifying context services as the exit manager.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the appropriate call as shown in the syntax diagrams. You must code the parameters in the CALL statement as shown.

CALL CTXSCID	(return_code ,context_interest_token ,context_interest_data)

CALL CTXSCID2	(return_code ,context_interest_token ,context_interest_data ,current_context_interest_data)

CALL CTX4SCID	(return_code ,context_interest_token ,context_interest_data ,current_context_interest_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Set_Context_Interest_Data service.

,context_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the context interest token that identifies your resource manager's interest in the context. Your resource manager received the token from the Express_Context_Interest service.

,context_interest_data
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the context interest data the system is to associate with this context interest. The Express_Context_Interest service can also specify this data. If context interest data already exists, the system replaces it with the context interest data you supply.

,current_context_interest_data
Supplied and returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For CTXSCID2 callers, specifies the current context interest data the system is to compare with context interest data. As part of a compare and swap, if current_context_interest_data matches the context interest data, then the context interest data is set to context_interest_data. Otherwise, current_context_interest_data returns the context interest data.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00170000' or X'00170001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 CTX_OK	**Meaning**: Successful completion. **Action**: None.
8 CTX_CUR_CI_DATA_MISMATCH	**Meaning**: Program error. The context interest data parameter no longer contains the value that was passed on input to Set_Context_Interest_Data. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
103 CTX_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.
365 CTX_CI_TOKEN_INV	**Meaning**: Program error. The context interest token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
701 CTX_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the context specified in the call is not in a valid state to issue the service call. The resource manager must be in set state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager retrieves the current context interest data value for an interest identified by *context_interest_token* and updates that value to *new_ci_data*. Storage for the call parameters has already been allocated.

```
⋮
DONE = FALSE
CI_TOKEN = CONTEXT_INTEREST_TOKEN
/* RETRIEVE CURRENT VERSION OF CI_DATA */
CALL CTXRCID(RC,CI_TOKEN,CURRENT_CI_DATA)
/* BUILD NEW DATA */
DO WHILE(¬DONE)
      CI_DATA = NEW_CI_DATA
      /* UPDATE CI_DATA IF IT HAS NOT CHANGED SINCE LAST TIME I LOOKED */
      CALL CTXSCID2(RC,CI_TOKEN,CI_DATA,CURRENT_CI_DATA)
      IF RC ¬= CTX_CUR_CI_DATA_MISMATCH THEN
            DONE = TRUE
END DO
⋮
```

Switch_Context (CTXSWCH, CTX4SWCH)

- CTXSWCH is for AMODE(31) callers.
- CTX4SWCH is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Switch_Context service to switch the context associated with the application's task to another context. In response to the call, the system returns a return code.

A context can be associated with only one task at a time. A context represents the resources for a work request; a context consists of the application program requesting the work and the protected resources involved in the work.

A native context exists when an application requests work. A resource manager can associate a privately-managed context with an application by calling the Switch_Context service.

Possible context switches: A call to the Switch_Context service can switch the context for the current task:

- From one privately-managed context to another privately managed context
- From the native context to a privately-managed context
- From a privately-managed context to the native context

The call cannot be used to switch from one native context to another.

Results of context switches: The results of using the call to associate the current application's task with a different context depend on the type of the previously current context:

- If the previously current context was a native context, it will still be associated with the task, but it will no longer be the current context.
- If the previously current context was a privately-managed context, it will be disassociated from the task. If the call specifies a new privately-managed context, the new context becomes the current context. Otherwise, the native context becomes the current context for the task.

When it processes the Switch_Context service, the system invokes each CONTEXT_SWITCH exit routine set by a resource manager with an interest in the context. Any CONTEXT_SWITCH exit routine can disallow the context switch.

Environment

The requirements for the resource manager are:

Minimum authorization:	None
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (CTXSWCH) 64 bit (CTX4SWCH)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the resource manager.

Programming requirements

Either link edit the resource manager's object code with the linkable stub routine CTXCSS (31 bit) or CTX4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the service. The high level language (HLL) definitions for the callable service are:

HLL Definition	Description
CTXASM	390 Assembler declarations
CTXC	C/390 declarations
FOMUTXC	z/OS HFS header files

Restrictions

To call the service, the resource manager associated with the context specified in the call must be in **set** state, which means it has registered and called the Set_Exit_Information service, specifying context services as the exit manager.

The context to be associated with the current application's task must not be already associated with another task.

If you are coding an RRS exit routine, do not call this service to process the context associated with the UR passed to the exit routine in the *ur_interest_token* parameter.

Resource managers that are PKM 8–15 problem state must register using the Register_Resource_Manager service from the home address space before invoking this service. They can only switch contexts obtained by a key 8–15 problem state resource manager which registered from the home address space.

Note: A PKM 8–15 problem state resource manager can switch to and from the native context.

Input register information

Before issuing the call, the resource manager does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the resource manager, the GPRs contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the resource manager, the ARs contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some resource managers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the resource manager depends, the resource manager must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL CTXSWCH	(return_code ,context_token ,disassociated_context_token)

CALL CTX4SWCH	(return_code ,context_token ,disassociated_context_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Switch_Context service.

,context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token for the context to be associated with the current application's task:

- 0: Binary zeros specify the native context. The call switches the task from a privately-managed context to the native context.
- token: Specifies the context token of a privately-managed context. The call switches the task from its current context to the specified privately-managed context. The current context can be a privately-managed context or a native context.

,disassociated_context_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token for the privately-managed context that was disassociated by the switch.

- 0: Binary zeros, if the previous current context was the native context.

- token: The disassociated context token. It identifies the privately-managed context that has been disassociated from the current task.

ABEND codes

The call might result in an abend X'AC7' with a reason code of either X'00150000' or X'00150001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 CTX_OK	**Meaning**: Successful completion. **Action**: None.
103 CTX_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
104 CTX_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 CTX_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 CTX_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
361 CTX_CONTEXT_TOKEN_INV	**Meaning**: Program error. The context token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
362 CTX_PRIVATE_CURRENT	**Meaning**: Program error. The privately-managed context specified in the *context_token* parameter in the call is already the current context. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
363 CTX_OTHER_WU_NATIVE	**Meaning**: Program error. The context specified in the *context_token* parameter in the call is the native context for another task or SRB. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
366 CTX_PRIVATE_OTHER_WU	**Meaning**: Program error. The privately-managed context specified in the call is the current context of another work unit. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
368 CTX_CURRENT_WU_NATIVE	**Meaning**: Program error. The context specified in the *context_token* parameter in the call is the native context and is already the current context. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
36A CTX_DU_TERMINATING	**Meaning**: Environmental error. The task or SRB associated with or to be associated with the context specified in the *context_token* parameter in the call is terminating. The system rejects the service call. **Action**: None.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
756 CTX_AUTH_FAILURE	**Meaning**: Program error. The resource manager is PKM 8–15 problem state and specified a context token that does not belong to a PKM 8–15 problem state resource manager registered in the home address space. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
800 CTX_DISALLOW_SWITCH	**Meaning**: Program error. A CONTEXT_SWITCH exit routine disallowed the context switch requested in the call. The system rejects the service call. **Action**: Check the resource manager for a probable coding or environmental error. Correct the resource manager and rerun it.
801 CTX_DISALLOW_SWITCH_WU	**Meaning**: Program error. A CONTEXT_SWITCH exit routine disallowed the context switch requested in the call because the calling resource manager is running under the wrong work unit. The system rejects the service call. **Action**: Check the resource manager for a probable coding or environmental error. Correct the resource manager and rerun it.
FFF CTX_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to switch an application's task or SRB to another context. Storage for the call parameters has been allocated.

```
⋮
C_TOKEN = CONTEXT_1
CALL CTXSWCH(RC,C_TOKEN,DISASSOC_C_TOKEN)
IF RC ≠ CTX_OK THEN
⋮
```

Chapter 7. Callable resource recovery services

This section describes the callable services that an authorized resource manager can use to request resource recovery services. The chapter lists the services in alphabetical order by descriptive name.

Backout_Agent_UR (ATRABAK, ATR4ABAK)

- ATRABAK is for AMODE(31) callers.
- ATR4ABAK is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager that has taken the server distributed syncpoint resource manager (SDSRM) role calls the Backout_Agent_UR service to tell RRS to back out the unit of recovery (UR) associated with the specified UR interest.

Your resource manager can invoke this service either to initiate a backout operation for an **in-flight** UR or to resolve an **in-doubt** UR to **in-backout**. Backout_Agent_UR changes the unit of recovery state to **in-forget** or **forgotten**.

If a resource manager with an interest in a UR has taken the SDSRM role, RRS will implicitly change the *log_option* to ATR_DEFER_EXPLICIT under any of the following conditions:

- When the application backs out the UR through a call to the Backout_UR service or the Application_Backout_UR service.
- When an RRS panel or the ATRSRV macro is used to resolve an **in-doubt** UR.
- When RRS recreates a committed or backed out UR during restart processing.

If any of these conditions has occurred, RRS returns the ATR_UR_STATE_ERROR return code. The UR might be in any state, but, once it reaches **in-forget**, it will remain in that state until the Forget_Agent_UR service is called. RRS waits for Forget_Agent_UR to ensure that the resource manager that has taken the SDSRM role is always informed of the results of the UR and allows the resource manager to safely prevote its BACKOUT and COMMIT exits.

Environment

Authorization:	Supervisor state, or PKM allowing keys 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (ATRABAK) 64 bit (ATR4ABAK)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	All parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

Table 34. Backout_Agent_UR (ATRABAK, ATR4ABAK) Programming requirements

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

To use the service:

- The resource manager state must be **run**.
- The unit of recovery state must be **in-flight** or **in-doubt**.

CAUTION:
The resource manager must ensure that no application can be updating protected resources for the unit of recovery being backed out. This is necessary to ensure that no resource manager taking part in the unit of recovery sees updates being made on behalf of a unit of recovery at the same time as they are executing syncpoint processing.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller

depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

CALL ATRABAK	(return_code ,UR_interest_token ,log_option)

CALL ATR4ABAK	(return_code ,UR_interest_token ,log_option)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Backout_Agent_UR service.

UR_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that uniquely represents an instance of the resource manager's interest in the particular UR.

log_option
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies how RRS is to process log entries for the unit of recovery. Code one of the following values:

Table 35. Backout_Agent_UR (ATRABAK, ATR4ABAK) Parameters

Constant in Hexadecimal (Decimal) Equate Symbol	Description
X'0' (0) ATR_DEFER_IMPLICIT	**Meaning:** RRS is to delete the log record when the UR state changes to **forgotten**. The deletion is an unforced deletion. Your resource manager will not call the Forget_Agent_UR_Interest service.
X'1' (1) ATR_DEFER_EXPLICIT	**Meaning:** RRS must keep the log record for the unit of recovery until your resource manager calls the Forget_Agent_UR_Interest service. The *log_option* specified on Forget_Agent_UR_Interest then determines how RRS is to process the log entry. Your resource manager will call the Forget_Agent_UR_Interest service.
X'2' (2) ATR_IMMEDIATE	**Meaning:** RRS is to immediately delete from the log the resource manager's interest in the UR. RRS hardens a new log record without the interest before driving any additional exit routines.

Abend codes

The call might result in an abend X'5C4' with a reason code of either X'001A0000' or X'001A0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the program, GPR 15 and *return_code* contain a hexadecimal return code.

Table 36. Backout_Agent_UR (ATRABAK, ATR4ABAK)Return codes

Return Codes in Hexadecimal Equate Symbol	Description
0 ATR_OK	**Meaning:** The backout operation completed successfully. All protected resources have been returned to the previous state of consistency. If the resource manager has taken the SDSRM role, and the *log_option* is ATR_IMMEDIATE, RRS will have deleted its interest before returning control from the service. **Action:** Continue normal processing.

Table 36. Backout_Agent_UR (ATRABAK, ATR4ABAK)Return codes (continued)

Return Codes in Hexadecimal Equate Symbol	Description
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The calling program is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The calling program is holding one or more locks; no locks can be held. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system level does not support this service. The system rejects the service call. **Action**: Remove the calling program from the system, and install it on a system that supports RRS. Then rerun the calling program.
12D ATR_BACKED_OUT_OUTCOME_ PENDING	**Meaning**: The backout operation completed. However, the state of one or more of the protected resources is not known. **Action**: Continue normal processing for a backed out unit of recovery.
12E ATR_BACKED_OUT_OUTCOME_ MIXED	**Meaning**: The backout operation completed. However, at least one of the protected resources has advanced to a new synchronization state. **Action**: Report the error to the other transactional participants.

Table 36. Backout_Agent_UR (ATRABAK, ATR4ABAK)Return codes (continued)

Return Codes in Hexadecimal Equate Symbol	Description
370 ATR_URI_TOKEN_INV	**Meaning:** The specified *UR_interest_token* does not represent a valid expression of interest. This condition can occur after RRS has terminated and restarted. **Action:** The system rejects this service request. Check program logic for probable coding error.
395 ATR_LOG_OPT_INV	**Meaning:** The specified *log_option* value is not valid. **Action:** The system rejects this service request. Check program logic for probable coding error.
701 ATR_RM_STATE_ERROR	**Meaning:** The resource manager state is not valid for this request. **Action:** The system rejects this service request. Check program logic for probable coding error.
702 ATR_RM_EXITS_UNSET	**Meaning:** RRS has unset the RRS exit routines for this resource manager. **Action:** The system rejects this service request. The resource manager must reset its RRS exit routine information and begin restart processing with RRS.
731 ATR_UR_STATE_ERROR	**Meaning**: If this service was called to resolve an **in_doubt** UR, the **in-doubt** condition might have already been resolved by operator action. If the service was called to back out an **in-flight** UR, the application might have already requested backout. Call Retrieve_UR_Data or Retrieve_Side_Information to obtain information about the state of the UR. If you receive this return code, you must call Forget_Agent_UR to complete processing for the UR. **Action:** Call Forget_Agent_UR to complete the processing of this UR.
74A ATR_NOT_SERVER_DSRM	**Meaning:** The resource manager does not have the server distributed syncpoint resource manager role for the unit of recovery. **Action:** The system rejects this service request. Check program logic for probable coding error.

Table 36. Backout_Agent_UR (ATRABAK, ATR4ABAK)Return codes (continued)

Return Codes in Hexadecimal Equate Symbol	Description
750 ATR_RESPOND_CONTINUE_REQUIRED	**Meaning:** The resource manager must call Respond_to_Retrieved_Interest before it can call Backout_Agent_UR for this interest. **Action:** The system rejects this service request. Call Respond_to_Retrieved_Interest, then call Backout_Agent_UR for this interest.
F00 ATR_NOT_AVAILABLE	**Meaning:** RRS is not available. **Action:** The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F04 ATR_UNEXPECTED_UR_ERROR	**Meaning:** System error. While processing the UR, RRS has encountered an unexpected error that might have damaged the UR. The system rejects the service call. **Action:** Contact the system programmer who maintains RRS at your installation. Manual intervention might be needed to restore consistent resources.
FFF ATR_UNEXPECTED_ERROR	**Meaning:** This service routine encountered an unexpected error. **Action:** The system rejects this service request. Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager needs to back out a unit of recovery. Storage for the call parameters has been allocated.

```
⋮
URI_TOKEN = MY_URI_TOKEN
FTOPT=ATR_DEFER_IMPLICIT
CALL ATRABAK(RC,URI_TOKEN,FTOPT)
⋮
```

Backout_UR (ATRBACK, ATR4BACK)

- ATRBACK is for AMODE(31) callers.
- ATR4BACK is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager or application program calls the Backout_UR service to indicate that the changes for the unit of recovery (UR) are not to be made. In

response, RRS requests that the resource managers return their resources to the values they had before the UR was processed, then issues a return code to the caller.

This call performs the same services as the Application_Backout_UR (SRRBACK) service, with one exception: Backout_UR provides return codes for many error conditions that cause Application_Backout_UR to abnormally end the calling program with abend code X'5C4'. For a description of Application_Backout_UR, see *z/OS MVS Programming: Callable Services for High-Level Languages*.

Environment

The requirements for the caller are:

Minimum authorization:	None
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRBACK)
	64 bit (ATR4BACK)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

Table 37. Backout_UR (ATRBACK, ATR4BACK) Programming requirements

HLL Definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The UR state must be **in-reset** or **in-flight**.

The UR must not be in local transaction mode.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRBACK	(return_code)

CALL ATR4BACK	(return_code)

Parameters

The parameters are explained as follows:

return_code
Returned parameter:

- Type: Integer
- Character set: N/A
- Length: Full word

Contains the return code from the Backout_UR service.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00170000' or X'00170001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and **return_code** contain a hexadecimal return code.

Table 38. Backout_UR (ATRBACK, ATR4BACK) Return Codes

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The calling program is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The calling program is holding one or more locks; no locks can be held. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system level does not support this service. The system rejects the service call. **Action**: Remove the calling program from the system, and install it on a system that supports RRS. Then rerun the calling program.

Table 38. Backout_UR (ATRBACK, ATR4BACK) Return Codes (continued)

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
12D ATR_BACKED_OUT_OUTCOME_PENDING	**Meaning**: Environmental error. RRS requested that the resource managers back out the changes to the resources. The backout was not completed. The reason might be that: • A resource manager failed with a protected interest in an incomplete UR. • RRS failed before UR processing is completed. RRS cannot determine if one or more of the protected resources was backed out. **Action**: The action by the calling program depends on the system environment. Some possible actions are: • Display a warning message to the end user. • Write an exception entry into an output log. • Abnormally end the application because the resource manager will not allow any further changes to the resource until the situation is resolved.
12E ATR_BACKED_OUT_OUTCOME_MIXED	**Meaning**: Environmental error. The requested backout of changes was completed; however, one or more of the protected resources were changed. **Action**: Same as the action for return code 12D.
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The UR is not in a valid state or a valid transaction mode for the service call. The UR state must be **in-reset** or **in-flight**. The transaction mode must not be local. The system rejects the service call. **Action**: Check the calling program for a probable coding error or an application environment configuration error. Correct the calling program and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning:** RRS is not available. **Action:** The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Table 38. Backout_UR (ATRBACK, ATR4BACK) Return Codes (continued)

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F04 ATR_UNEXPECTED_UR_ERROR	**Meaning:** System error. While processing the UR, RRS has encountered an unexpected error that might have damaged the UR. The system rejects the service call. **Action:** Contact the system programmer who maintains RRS at your installation. Manual intervention might be needed to restore consistent resources.
F05 ATR_UNEXPECTED_CTX_ERROR	**Meaning**: Environmental error. The service call encountered an unexpected error from a context services service. The system rejects the service call. **Action**: Examine the dump from context services and correct the problem, then rerun the calling program.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the calling program backs out a UR.

```
⋮
CALL ATRBACK (RC)
⋮
```

Begin_Restart (ATRIBRS, ATR4IBRS)

- ATRIBRS is for AMODE(31) callers.
- ATR4IBRS is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Begin_Restart service to begin resource manager restart. Before calling Begin_Restart, your resource manager should call the Retrieve_Log_Name service to check log names and tokens to make sure the restart logs are the same as the previous logs. After calling Begin_Restart, your resource manager should call the Retrieve_UR_Interest service repetitively to obtain your interests in incomplete units of recovery (URs).

In response to the Begin_Restart call, RRS issues a return code. If the code is `ATR_HARDENED_DATA_LOST`, some hardened data, which is stored in an RRS log on nonvolatile external storage, has been lost; the calls to the Retrieve_UR_Interest service might not return all of your interests in incomplete URs. In this case, your resource manager should not back out any URs that the Retrieve_UR_Interest service does not return.

Cold Starts: If hardened data was lost from an RRS log, this restart might appear as a cold start. Your resource manager can recognize a cold start when its call to the Retrieve_Log_Name service does not return the name of its previous resource manager log.

Your resource manager's first start is always a cold start.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRIBRS) 64 bit (ATR4IBRS)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The resource manager associated with the resource manager token specified in the call must be in **set** state, which means it has registered and set its exit routines with RRS. After a successful Begin_Restart call, the resource manager enters the **restart** state.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system

2-13 Unchanged

14 Used as a work register by the system

15 Return code

When control returns to the caller, the ARs contain:

Register
Contents

0-1 Used as work registers by the system

2-13 Unchanged

14-15 Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

Calls to the Begin_Restart service can significantly affect performance because two events must be serialized across the sysplex:

- RRS initialization
- Begin_Restart calls from other resource managers

Also, while processing the call, RRS might need to read one or more logs.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRIBRS	(return_code ,resource_manager_token)

CALL ATR4IBRS	(return_code ,resource_manager_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter
Type: Integer
Character Set: N/A
Length: 4 bytes

Is a fullword that receives an integer return code from the service.

,resource_manager_token
Supplied parameter

Type: Character string

Character Set: No restrictions

Length: 16 bytes

Is the resource manager token. The token is a 16-byte character string that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00030000' or X'00030001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and **return_code** contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
301 ATR_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the resource manager token specified in the call is not in **set** state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
738 ATR_RM_ATTR_INC	**Meaning**: Program error. For the incomplete UR interests, the resource manager had not issued Set_Exit_Information calls to set all of the required exit routines. For example, if the resource manager has a distributed syncpoint role, a DISTRIBUTED_SYNCPOINT exit routine is required. However, the resource manager has not called Set_Exit_Information to set a DISTRIBUTED_SYNCPOINT exit routine. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning:** RRS is not available. **Action:** The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F01 ATR_HARDENED_DATA_LOST	**Meaning**: System error. RRS has lost some hardened data. The system accepts the Begin_Restart call, but following calls to the Retrieve_UR_Interest service might not return all of the incomplete UR interests. **Action**: The resource manager should process the incomplete URs obtained from the Retrieve_UR_Interest service. The resource manager should not, however, back out any URs that Retrieve_UR_Interest does not return.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F02 ATR_RESTART_WRONG_SYSTEM	**Meaning**: Environmental error. The resource manager is not restarting on the correct system. The resource manager's incomplete URs are on another system in the sysplex. The system rejects the service call. **Action**: Restart the resource manager on the correct system.
F07 ATR_RM_GROUP_RRS_DOWNLEVEL	**Meaning**: Environmental error. The restarting Resource Manager belongs to an RM group which has utilized an RRS function that is not supported by this version of RRS. The RRS on this system is downlevel and cannot honor the request to restart. The system rejects the service call. **Action**: Restart the resource manager on the correct system.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to begin its restart.

```
⋮
RM_TOKEN = MY_RM_TOKEN
CALL ATRIBRS (RC,RM_TOKEN)
⋮
```

Begin_Transaction (ATRBEG, ATR4BEG)

- ATRBEG is for AMODE(31) callers.
- ATR4BEG is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A work manager calls the Begin_Transaction service to begin a transaction and set the transaction mode. Begin_Transaction allows an application to clearly mark the beginning boundary of a transaction. The transaction mode, which affects resource managers within the boundaries of this transaction, can be local or global, as follows:

- In local mode, resource managers within the transaction boundaries must behave independently: the application can use RM-specific functions to commit and roll back any connections it might have to resource managers. If the application has multiple connections to the same resource manager, the resource manager treats the resources affected via each connection as completely separate for the purposes of syncpoint processing. Committing or backing out the resources affected by one connection does not affect the resources affected by another

connection. RRS prevents the use of any global commit functions that would act upon a UR that is in local transaction mode. Local URs begun with Begin_Transaction must be ended with a call to the End_Transaction service.

- In global mode, resource managers are to accept two-phase commit cues from RRS, and all updates made through RRS-compliant resource managers are made atomically. In global mode, an application cannot determine two different outcomes for any of its connections under a global unit of recovery; the resource managers prevent the application from using RM-specific local commit functions.

Environment

The requirements for the caller are:

Minimum authorization:	Any
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRBEG) 64 bit (ATR4BEG)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL Definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The current UR state must be **in-reset**.

The current default environment setting for transaction mode must not be hybrid-global.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRBEG	(return_code ,diag_area ,transaction_mode ,ur_token ,ur_identifier)

CALL ATR4BEG	(return_code ,diag_area ,transaction_mode ,ur_token ,ur_identifier)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Begin_Transaction service.

,diag_area
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 32 bytes

Contains diagnostic data from Begin_Transaction to help IBM Service determine the cause of a Begin_Transaction failure. Be sure to log this data when recording any information about a Begin_Transaction failure.

,transaction_mode
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates the transaction mode of the transaction to be started. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
1 (1) ATR_GLOBAL_MODE	Set the transaction mode for the current UR to global, and set the UR state to **In-Flight**.
2 (2) ATR_LOCAL_MODE	Set the transaction mode for the current UR to local, and set the UR state to **In-Flight**.

,ur_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives the token that uniquely represents the new UR. UR tokens do not persist across restarts of the resource manager, RRS, or the system.

,ur_identifier
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives a UR identifier (URID) from the service. The URID uniquely identifies the UR. URIDs returned for global URs persist across restarts of the resource

manager, RRS, or the system. URIDs are also returned for local URs, but they do not persist across failures because RRS does not harden any information about local URs.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00023000' or X'00023001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that is running a version of RRS that supports this service call. Then rerun the resource manager.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
363 ATR_TRAN_MODE_INV	**Meaning**: Program error. The transaction mode specified in the call was not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
36A ATR_DU_TERMINATING	**Meaning**: Environmental error. The task associated with the context specified in the call is ending. The system rejects the service call. **Action**: None.
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The UR is not in a valid state for the service call. The UR must be in the **in-reset** state. RRS does not support nested transactions. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
804 ATR_HYBRID_GLOBAL_MODE_ERROR	**Meaning**: Program error. The current RRS default for transaction mode is ATR_HYBRID_GLOBAL_MODE; hybrid-global mode is not valid for this service. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the work manager issues a call to begin a local transaction.

```
⋮
TRAN_MODE = ATR_LOCAL_MODE
CALL ATRBEG(RC,DIAG_DATA,TRAN_MODE,UR_TOKEN,URID)
```

```
IF RC = ATR_OK THEN
:
```

Change_Interest_Type (ATRSIT, ATR4SIT)

- ATRSIT is for AMODE(31) callers.
- ATR4SIT is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Change_Interest_Type service to change an interest in a unit of recovery (UR) from unprotected to protected. The resource manager can change the interest type until the UR enters the **in-prepare** state.

On the call, you also specify action for a resource manager failure and provide persistent interest data.

In response to the call, resource recovery services/MVS (RRS) returns:

- A return code
- A UR identifier (URID)

Action for Resource Manager Failure: On the Change_Interest_Type call, you can specify how RRS should process requests to commit the UR if your resource manager becomes:

- **Unregistered:** Your resource manager is no longer registered as a resource manager. See "Register_Resource_Manager (CRGGRM, CRG4GRM)" on page 137 for a description of how a resource manager can become unregistered.
- **Unset:** Your resource manager's exit routines are no longer set with RRS.

RRS reacts to a resource manager failure as follows:

- **Standard processing:** RRS backs out this UR, if the state of the UR is **in-reset**, **in-flight**, **in-state-check**, or **in-prepare**.

Persistent interest data: In the Change_Interest_Type call, your resource manager can provide persistent interest data for the protected interest. When hardening information for the interest in an RRS log, RRS records the persistent interest data. Because the data is hardened, it will be available if your resource manager restarts or if RRS restarts, forcing your resource manager to restart.

Your resource manager can also provide persistent interest data in a call to: Express_UR_Interest, Set_Persistent_Interest_Data, or Retain_Interest. Your resource manager can retrieve persistent interest data in a call to: Retrieve_UR_Interest or Retrieve_Interest_Data.

URID: Save the returned UR identifier (URID) with the information about the UR in your resource manager log. During restart processing after your resource manager, RRS, or the system fails, your resource manager obtains the URID for an incomplete UR from a Retrieve_UR_Interest call. Compare the URID from Retrieve_UR_Interest with the URIDs in your resource manager log to find the data for the incomplete UR.

Your resource manager can also obtain the URID from a call to: Express_UR_Interest, Retrieve_UR_Interest, Retrieve_UR_Data, or Retain_Interest.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRSIT) 64 bit (ATR4SIT)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions

Programming Requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The UR state must be **in-flight** or **in-state-check**.

The state of the resource manager associated with the UR interest token specified in the call must be **run**, which means it has registered, set its exit routines with RRS, and completed restart.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRSIT	(return_code ,ur_interest_token ,ur_identifier ,interest_type ,failure_action ,persistent_interest_data_length ,persistent_interest_data)

CALL ATR4SIT	(return_code ,ur_interest_token ,ur_identifier ,interest_type ,failure_action ,persistent_interest_data_length ,persistent_interest_data)

Parameters

The parameters are explained as follows:

`return_code`
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Change_Interest_Type service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that uniquely represents an instance of the resource manager's interest in the UR. Your resource manager received the token from the Express_Interest service or the Retain_Interest service.

,ur_identifier
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives a UR identifier (URID) from the service. The URID uniquely identifies the UR.

,interest_type
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies the new type of interest the resource manager has in the UR. Specify the interest type as follows:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
1 (1) ATR_PROTECTED	**Protected:** The resource manager is now expressing a protected interest in the UR.

,failure_action
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies how RRS should process commit requests if your resource manager becomes unregistered or unset. Specify the failure action as follows:

Constant in: Hexadecimal (Decimal) Equate Symbol	Action
0 (0) ATR_FAIL_STANDARD	Standard processing

,persistent_interest_data_length
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies the length in bytes of the persistent interest data. The length can be 0 through 4096.

,persistent_interest_data
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified in *persistent_interest_data_length*

The persistent interest data for your resource manager's interest in the UR. RRS records this data in an RRS log.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'000F0000' or X'000F0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call is not one of the currently valid interests. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
371 ATR_INTEREST_TYPE_INV	**Meaning**: Program error. The *interest_type* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
372 ATR_FAILURE_ACTION_INV	**Meaning**: Program error. The *failure_action* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
376 ATR_PERSISTENT_DATA_LEN_INV	**Meaning**: Program error. The length specified in the *persistent_interest_data_len* parameter in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager must be in **run** state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: The resource manager must reset its RRS exits and begin restart processing with RRS.
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The UR is not in a valid state for the service call. The UR state must be **in-reset** or **in-flight**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
739 ATR_PROTECTED_INTEREST	**Meaning**: Program error. The URI_TOKEN specified in the call represents an interest that is already protected. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
749 ATR_MAX_UR_LOG_DATA_EXCEEDED	**Meaning**: Environmental error. This request will exceed the maximum amount of data that RRS can log for a UR. The system rejects the service call. **Action**: Fail the client program request or back out the UR. Verify that the space set up for logging is adequate.
F00 ATR_NOT_AVAILABLE	**Meaning:** RRS is not available. **Action:** The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Change_Interest_Type was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to change one of its interests in a UR from unprotected to protected.

```
⋮
URI_TOKEN = MY_URI_TOKEN
P_DATA_LEN = LENGTH(MY_P_DATA)
P_DATA = MY_P_DATA
INT_TYPE = ATR_PROTECTED
FAIL_ACT = ATR_FAIL_STANDARD
CALL ATRSIT(RC,URI_TOKEN,URID,INT_TYPE,FAIL_ACT,
            P_DATA_LEN,P_DATA)
IF RC ≠ ATR_OK THEN
   /* Handle error */
⋮
```

Commit_Agent_UR (ATRACMT, ATR4ACMT)

- ATRACMT is for AMODE(31) callers.

- ATR4ACMT is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager that has taken the server distributed syncpoint resource manager (SDSRM) role calls Commit_Agent_UR to tell RRS to commit the unit of recovery (UR) associated with the specified UR interest. The SDSRM can invoke this service to resolve an **in_doubt** unit of recovery to **in_commit**.

Commit_Agent_UR changes the unit of recovery state to **in_forget** or **forgotten**.

If a resource manager with an interest in a UR has taken the SDSRM role, RRS will implicitly change the *log_option* to ATR_DEFER_EXPLICIT under any of the following conditions:

- When an RRS panel or the ATRSRV macro is used to resolve an **in-doubt** UR.
- When RRS re-creates a committed or backed out UR during restart processing.

If any of these conditions has occurred, RRS returns the ATR_UR_STATE_ERROR return code. The UR might be in any state, but, once it reaches **in-forget**, it will remain in that state until the Forget_Agent_UR service is called. RRS waits for Forget_Agent_UR to ensure that the resource manager that has taken the SDSRM role is always informed of the results of the UR and allows the resource manager to safely prevote its BACKOUT and COMMIT exits.

Environment

Authorization:	Supervisor state, or PKM allowing keys 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRACMT) 64 bit (ATR4ACMT)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	All parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

Table 39. Commit_Agent_UR (ATRACMT, ATR4ACMT) Programming requirements

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

To use the service:

- The resource manager state must be **run**.

- The unit of recovery state must be **in-doubt**.

CAUTION:
The resource manager must ensure that no application can be updating protected resources for the unit of recovery being committed. This is necessary to ensure that no resource manager taking part in the unit of recovery sees updates being made on behalf of a unit of recovery at the same time as they are executing syncpoint processing.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

CALL ATRACMT	(return_code ,UR_interest_token ,log_option)

CALL ATR4ACMT	(return_code ,UR_interest_token ,log_option)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code for the Commit_Agent_UR service.

UR_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that uniquely represents an instance of the resource manager's interest in the particular UR. The resource manager received the token from: Express_UR_Interest, Retrieve_UR_Interest, or Retain_Interest.

log_option
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies how RRS is to process log entries for the unit of recovery. Code one of the following values:

Table 40. Commit_Agent_UR (ATRACMT, ATR4ACMT)Parameters

Constant in Hexadecimal (Decimal) Equate Symbol	Description
X'0' (0) ATR_DEFER_IMPLICIT	**Meaning:** RRS is to logically delete the log record when the unit of recovery state changes to **Forgotten**. Your resource manager will not call the Forget_Agent_UR_Interest service.

Table 40. Commit_Agent_UR (ATRACMT, ATR4ACMT)Parameters (continued)

Constant in Hexadecimal (Decimal) Equate Symbol	Description
X'1' (1) ATR_DEFER_EXPLICIT	**Meaning:** RRS must keep the log record for the unit of recovery until your resource manager calls the Forget_Agent_UR_Interest service. The *log_option* specified on Forget_Agent_UR_Interest then determines how RRS processes the log entry. Your resource manager will call the Forget_Agent_UR_Interest service.
X'2' (2) ATR_IMMEDIATE	**Meaning:** RRS is to immediately delete from the log the interest the server distributed syncpoint manager (SDSRM) has in the UR. RRS hardens a new log record without the interest before driving any additional exit routines.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'001B0000' or X'001B0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and **return_code** contain a hexadecimal return code.

Table 41. Commit_Agent_UR (ATRACMT, ATR4ACMT) Return codes

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: The commit operation completed successfully. All protected resources have advanced to a consistent state. **Action**: Continue normal processing.
65 ATR_COMMITTED_OUTCOME_ PENDING	**Meaning**: The commit operation completed. The RRS decision was to advance to a consistent state. However, the state of one or more of the protected resources is not known. **Action**: Continue normal processing for a committed unit of recovery.
66 ATR_COMMITTED_OUTCOME_ MIXED	**Meaning**: The commit operation completed. The RRS decision was to advance to a consistent state. However, the state of one or more of the protected resources has been returned to the previous consistent state. **Action**: Report the error to the other transactional participants.

Table 41. Commit_Agent_UR (ATRACMT, ATR4ACMT) Return codes (continued)

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: The caller is disabled. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: The caller is holding one or more locks. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: The system level does not support this service. The system rejects this service request. **Action**: Remove the calling program from the system, and install it on a system that supports RRS. Then rerun the calling program.
370 ATR_URI_TOKEN_INV	**Meaning**: The specified *UR_interest_token* does not represent a valid expression of interest. This condition can occur after RRS has terminated and restarted. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
395 ATR_LOG_OPT_INV	**Meaning**: The specified *log_option* value is not valid. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
701 ATR_RM_STATE_ERROR	**Meaning**: The resource manager state is not valid for this request. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
702 ATR_RM_EXITS_UNSET	**Meaning**: RRS has unset the RRS exit routines for this resource manager. The system rejects this service request. **Action**: The resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Table 41. Commit_Agent_UR (ATRACMT, ATR4ACMT) Return codes (continued)

Return Code in: Hexadecimal Equate Symbol	Meaning and action
731 ATR_UR_STATE_ERROR	**Meaning**: The UR state is not valid for this request. If this service was called to resolve an **in_doubt** UR, the **in-doubt** condition might have already been resolved by operator action. Call Retrieve_UR_Data or Retrieve_Side_Information to obtain information about the state of the UR. If you receive this return code, you must call Forget_Agent_UR to complete processing for the UR. **Action**: Call Forget_Agent_UR to complete the processing of this UR.
74A ATR_NOT_SERVER_DSRM	**Meaning**: The resource manager does not have the server distributed syncpoint resource manager role for the unit of recovery. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
750 ATR_RESPOND_CONTINUE_ REQUIRED	**Meaning**: The resource manager must call Respond_to_Retrieved_Interest before it can call Commit_Agent_UR for this interest. **Action**: The system rejects this service request. Call Respond_to_Retrieved_Interest, then call Commit_Agent_UR for this interest.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F04 ATR_UNEXPECTED_UR_ERROR	**Meaning**: System error. While processing the UR, RRS has encountered an unexpected error that might have damaged the UR. The system rejects the service call. **Action**: Contact the system programmer who maintains RRS at your installation. Manual intervention might be needed to restore consistent resources.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: This service routine encountered an unexpected error. The system rejects this service request. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager wants to commit the unit of recovery. Storage for the call parameters has been allocated.

```
⋮
URI_TOKEN = MY_URI_TOKEN
FTOPT=ATR_DEFER_IMPLICIT
CALL ATRACMT(RC,URI_TOKEN,FTOPT)
⋮
```

Commit_UR (ATRCMIT, ATR4CMIT)

- ATRCMIT is for AMODE(31) callers.
- ATR4CMIT is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager or application program calls the Commit_UR service to indicate that the changes for the unit of recovery (UR) are to be made permanent. To process the call, RRS requests that the resource managers make the changes permanent, then issues a return code to the calling program.

This call performs the same services as the Application_Commit_UR call (SRRCMIT), but provides return codes for many error conditions that cause Application_Commit_UR to abnormally end the calling program with abend code X'5C4'. For a description of Application_Commit_UR, see *z/OS MVS Programming: Callable Services for High-Level Languages*.

Environment

The requirements for the caller are:

Minimum authorization:	None
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRCMIT) 64 bit (ATR4CMIT)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	MVS standard linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The UR state must be **in-reset** or **in-flight**.

The UR must not be in local transaction mode.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRCMIT	(return_code)

CALL ATR4CMIT	(return_code)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Commit_UR service.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00180000' or X'00180001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
65 ATR_COMMITTED_OUTCOME_PENDING	**Meaning**: Environmental error. The commit completed. However, RRS cannot determine if all of the protected resources were changed. **Action**: The action by the calling program depends on the system environment. Some possible actions are: • Display a warning message to the end user. • Write an exception entry into an output log. • Abnormally end the application because the resource manager will not allow any further changes to the resource until the situation is resolved.
66 ATR_COMMITTED_OUTCOME_ MIXED	**Meaning**: Environmental error. The commit completed. However, one or more of the protected resources were not changed. **Action**: Same as the action for return code 65.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
C8 ATR_PROGRAM_STATE_CHECK	**Meaning**: Environmental error. The commit failed. The resource managers may have rejected the commit because one of the following occurred: • A communications interface conversation that is a protected resource is not in a required state: **send**, **send pending**, **defer receive**, **defer allocate**, **sync_point**, **sync_point send**, or **sync_point deallocate**. • A protected communications interface conversation is in **send** state. The program started sending the basic conversation logical record, but did not finish sending it. • A resource on the same system as the application is not in the proper state for the commit. **Action**: Initiate an action by a resource manager to get its resource to a committable state. Issue the call again.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The calling program is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The calling program is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the calling program from the system, and install it on a system that supports RRS. Then rerun the calling program.
12C ATR_BACKED_OUT	**Meaning**: Environmental error. The commit failed. The resource managers backed out the changes, returning the resources to the values they had before the UR was processed. **Action**: Same as the action for return code 65.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
12D ATR_BACKED_OUT_OUTCOME_ PENDING	**Meaning**: Environmental error. The commit failed. RRS requested that the resource managers back out the changes to the resources. However, RRS cannot determine if one or more of the protected resource was backed out. **Action**: Same as the action for return code 65.
12E ATR_BACKED_OUT_OUTCOME_ MIXED	**Meaning**: Environmental error. The commit failed. RRS requested that the resource managers back out the changes to the resources. One or more resources were backed out, but one or more were changed. **Action**: Same as the action for return code 65.
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The UR is not in a valid state or a valid transaction mode for the service call. The UR state must be **in-reset** or **in-flight**. The transaction mode cannot be local. The system rejects the service call. **Action**: Check the calling program for a probable coding error or an application environment configuration error. Correct the calling program and rerun it.
74C ATR_SDSRM_DISALLOWS_COMMIT	**Meaning**: Program error. The commit failed. Another resource manager involved in this UR has taken the server distributed syncpoint resource manager (SDSRM) role. Only the SDSRM can initiate the syncpoint operation for this UR. The system rejects this service request. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
756 ATR_CASCADED_UR_DISALLOWS_COMMIT	**Meaning**: Program error. The commit failed. The current UR is a child cascaded-UR. Only the top-level UR of a cascaded-UR family can be committed. The system rejects this service request. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F04 ATR_UNEXPECTED_UR_ERROR	**Meaning**: System error. While processing the UR, RRS has encountered an unexpected error that might have damaged the UR. The system rejects the service call. **Action**: Contact the system programmer who maintains RRS at your installation. Manual intervention might be needed to restore consistent resources.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F05 ATR_UNEXPECTED_CTX_ERROR	**Meaning**: Environmental error. The service call encountered an unexpected error from a context services service. The system rejects the service call. **Action**: Examine the dump from context services and correct the problem, then rerun the calling program.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the calling program commits a UR.

```
⋮
CALL ATRCMIT (RC)
⋮
```

Create_Cascaded_UR (ATRCCUR2, ATRCCUR3, ATR4CCUR)

A work manager calls the Create_Cascaded_UR service to create a UR that is associated with an existing UR. Typically, a work manager would do this when a single work request involves multiple work managers. The initial work manager would obtain the initial work context that represented the work request. When the work request moved from the execution environment of the original work manager into another work manager's environment, the latter work manager could obtain a new work context to represent the work request, allowing it to manipulate the work context as it needs. There are three versions of Create_Cascaded_UR, each with different parameters.

- ATRCCUR2 is for AMODE(31) callers and is the basic version of the service.
- ATRCCUR3 is for AMODE(31) callers and adds create options support.
- ATR4CCUR is for AMODE(64) callers, allows parameters in 64 bit addressable storage, and adds create options support.

Code your resource manager to call the version that includes the support you need.

By using the Create_Cascaded_UR service to create a new UR for the new work context that is cascaded from the original UR, the second work manager ensures that the resources changed by the work request when it is running in its new execution environment are committed or backed out atomically along with those resources changed while the work request was running in other environments.

A work manager that creates a cascaded UR is responsible for informing RRS when the application running under the UR is application execution complete by calling the Set_Side_Information service specifying the ATR_APPL_COMPLETE information identifier.

Express_UR_Interest (ATREINT2, ATREINT4, ATREINT5 or ATR4EINT) can also be used to create a cascaded UR, if the creator also wants to express interest in the UR.

Multisystem cascaded transactions: When one work manager requests another work manager, residing on a different system, to become part of an existing work request, the requesting work manager is responsible for transferring all of the data needed by the new work manager, including the UR token representing the work request. The new work manager may then use the Create_Cascaded_UR service to create a new UR, associated with a new work context, to be cascaded from the received UR token.

As with normal (non-multisystem) cascaded transactions, a work manager that creates a multisystem cascaded transaction is responsible for informing RRS when the part of the application executing under a multisystem cascaded UR is complete by using the Set_Side_Information service to mark the UR as application-complete.

See "Cascaded transactions" on page 69 for more information about cascaded transactions, including "Multisystem cascaded transactions" on page 70 for more information about multisystem cascaded transactions.

Environment

The requirements for the caller are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (ATRCCUR2, ATRCCUR3) 64 bit (ATR4CCUR)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	MVS standard linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

For the call, the child UR state must be **in-reset** and the parent UR must be **in-reset** or **in-flight**.

The parent UR must not be in local transaction mode.

When the resource manager issues the call in SRB mode:

- The call cannot specify a *child_context_token* of 0.
- The call cannot specify a *parent_UR_token* of 0.

A caller that is PKM 8–15 problem state must specify a *parent_UR_token* for a UR that is associated with a DU native context associated with a task in the current home address space, or a private context owned by a PKM 8–15 problem state resource manager registered in the home address space.

A caller that is PKM 8–15 problem state must specify a *child_context_token* for a context that is either a DU native context associated with a task in the current home address space, or a private context owned by a PKM 8–15 problem state resource manager registered in the home address space.

Note: A call to the Retrieve_UR_Data service that does not specify ATR_EXTENDED_STATES for the *states_option* could cause a UR to go into an **in-flight** state. Once a UR has gone **in-flight**, it can no longer be made into a cascaded UR.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the appropriate call as shown in the syntax diagrams. You must code the parameters in the CALL statement as shown.

CALL ATRCCUR2	(return_code ,parent_UR_token ,child_context_token ,child_UR_token ,child_UR_identifier)

CALL ATRCCUR3	(return_code ,parent_UR_token ,child_context_token ,child_UR_token ,child_UR_identifier ,create_options)

CALL ATR4CCUR	(return_code ,parent_UR_token ,child_context_token ,child_UR_token ,child_UR_identifier ,create_options)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Create_Cascaded_UR service.

,parent_UR_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token of the UR that is to be the parent of the UR specified by the *child_context_token*:

- 0: Binary zero specifies the current UR associated with the application's task active on the current system. Binary zero may be specified for either the *parent_UR_token* or the *child_context_token*, but not both.
- *token*: The UR token of a particular UR. The UR token may be from another system in the same logging group.

Your resource manager may have received the *parent_UR_token* from the Retrieve_UR_Data service or from a work manager from another system. If the UR token was received from another system, RRS will associate the new child UR, which has a context specified by *child_context_token*, with the top-level UR of the cascaded UR family that resides on the system where the work request originated.

,child_context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token of the context associated with the UR that is to be the child of the UR specified by the *parent_UR_token*:

- 0: Binary zero specifies the current UR associated with the application's task. Binary zero may be specified for either the *parent_UR_token* or the *child_context_token*, but not both.
- *token*: The token of the context associated with a particular UR.

Your resource manager may have received the *parent_UR_token* from the Retrieve_UR_Data service.

,child_UR_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives the token that uniquely represents the new unit of recovery.

Note: UR tokens do not persist across restarts of the resource manager, RRS, or the system.

,child_UR_identifier
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives a UR identifier (URID) from the service. The URID uniquely identifies the UR.

Note: Unlike UR tokens, URIDs are persistent across restarts of the resource manager, RRS, and the system.

,create_options
Supplied parameter

- Type: Bit string
- Character Set: N/A
- Length: 4bytes

On ATRCCUR3 calls, specifies various options that determine how RRS will process this interest. Each of the bits in *create_options* is either reserved or has a specific meaning. Each reserved bit must be specified as zero. Each other bit can be specified as either zero or one. The bit specifications are:

Bit Positions	Constant in: Hexadecimal Equate Symbol	Description
0–22	0	Reserved
23	00000000 ATR_DONT_END_ CONTEXT_MASK 00000100 ATR_END_ CONTEXT_MASK	**Auto context termination** A resource manager specifies zero when it does not want RRS to end the work context associated with the UR in which interest is being expressed when the UR completes. A resource manager specifies one when it wants RRS to end the work context associated with the UR in which interest is being expressed when the UR completes. **Note:** IBM strongly recommends that one only be specified by the resource manager that owns the work context.
24–31	0	Reserved

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'001F0000' or X'001F0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The calling program is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
104 ATR_MODE_INV	**Meaning**: Program error. The calling program specified 0 in *parent_UR_token* or *child_context_token*, but the calling program is not in task mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The calling program is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that is running a version of RRS that supports this service call. Then rerun the resource manager.
39A ATR_PARENT_UR_TOKEN_INV	**Meaning**: Program or environmental error. The UR token specified in the *parent_UR_token* parameter is not valid because one of the following is true: • It was coded incorrectly • The parent transaction failed • The system it resided on failed • The coordinator system failed • The RRS running on that system failed • The parent UR belongs to a system that is not in the same RRS logging group as this system If any of these conditions occurs, the system rejects the service call. **Action**: Check the calling program for a probable coding error. • If it's a program error, correct the program and rerun it. • If the work manager was creating a multisystem cascaded UR, the work manager must communicate the failure to the work manager originating the request. **Installation action**: Ensure that the originating work manager and this work manager are in the same RRS logging group.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
39B ATR_CHILD_CONTEXT_TOKEN_INV	**Meaning**: Program error. The context token specified in the *child_context_token* parameter is not valid. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
39E ATR_PARENT_DU_TERMINATING	**Meaning**: Environmental error. The task associated with the UR represented by the *parent_UR_token* parameter is ending. The system rejects the service call. **Action**: None.
39F ATR_CHILD_DU_TERMINATING	**Meaning**: Environmental error. The task associated with the context specified by the *child_context_token* parameter is ending. The system rejects the service call. **Action**: None.
3A0 ATR_SAME_CURRENT_CONTEXT_INV	**Meaning**: Program error. The UR represented by the *parent_UR_token* and the UR associated with the context represented by the *child_context_token* are both 0. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A1 ATR_SAME_PARENT_ CONTEXT_INV	**Meaning**: Program error. The UR represented by the *parent_UR_token*, which may have been specified with a 0, and the UR associated with the context represented by the *child_context_token* are the same UR. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A2 ATR_SAME_CHILD_CONTEXT_INV	**Meaning**: Program error. The UR represented by the *parent_UR_token* and the UR associated with the context represented by the *child_context_token*, specified with a 0, are the same UR. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
3A4 ATR_PARENT_AUTH_FAILURE	**Meaning**: Program error. The caller is PKM 8–15 problem state, but specified a *parent_UR_token* of a UR associated with a context which: • Does not belong to a PKM 8–15 problem state resource manager registered in the home address space. • Is not a native context in the home address space. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A5 ATR_CHILD_AUTH_FAILURE	**Meaning**: Program error. The caller is PKM 8–15 problem state, but specified a *child_context_token* of a context which: • Does not belong to a PKM 8–15 problem state resource manager registered in the home address space. • Is not a native context in the home address space. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3AD ATR_CREATE_OPTIONS_INV	**Meaning**: Program error. The *create_options* value specified on the call is not valid. Either reserved bits were nonzero or an unacceptable selection of options and parameters was specified. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
743 ATR_PARENT_UR_ STATE_ERROR	**Meaning**: Program error. The UR specified by the *parent_UR_token* is not in a valid state for the service call. The UR must be in an **in-reset** or **in-flight** state. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
744 ATR_CHILD_UR_STATE_ERROR	**Meaning**: Program error. The UR associated with the specified *child_context_token* is not in a valid state for the service call. The UR must be in an **in-reset** state. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
763 ATR_PARENT_LOCAL_TRAN_ MODE_INV	**Meaning**: Program error. The parent UR is in local transaction mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error or an application environment configuration error. Correct the calling program and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Create_Cascaded_UR was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the calling program attempts to create a parent-child relationship between the current unit of recovery and another unit of recovery that is to be the parent UR of the current UR. Storage for the call parameters has already been allocated.

```
⋮
PARENT_URTOKEN = TOPURTOKEN
CHILD_CTOKEN = 0
COPTS = ATR_DONT_END_CONTEXT_MASK
CALL ATRCCUR3(RC, PARENT_URTOKEN, CHILD_CTOKEN, CHILDURTOKEN,
              CHILDURID, COPTS)
IF RC=0 THEN
    NEW_URTOKEN = CHILDURTOKEN
    NEW_URID = CHILDURID

⋮
```

Delegate_Commit_Agent_UR (ATRADCT, ATRADCT1, ATR4ADCT)

A resource manager that has taken the server distributed syncpoint resource manager (SDSRM) role calls Delegate_Commit_Agent_UR to tell RRS to initiate and complete a syncpoint operation for the unit of recovery (UR) associated with the specified UR interest. This single call replaces a call to Prepare_Agent_UR, possibly followed by a call to either Commit_Agent_UR or Backout_Agent_UR.

There are three versions of Delegate_Commit_Agent_UR.

- ATRADCT is for AMODE(31) callers and is the basic version of this service.
- ATRADCT1 is for AMODE(31) callers and allows the specification of the commit_options mask option. This option allows RRS to delete the SDSRM interest prior to driving syncpoint processing, freeing RRS to perform exit optimization.
- ATR4DCT is for AMODE(64) callers, allows parameters in 64 bit addressable storage, and allows the specification of the commit_options mask option. This option allows RRS to delete the SDSRM interest prior to driving syncpoint processing, freeing RRS to perform exit optimization.

When the SDSRM calls ATRADCT, RRS takes responsibility for making the commit or backout decision for the UR based on the collective prepare vote. The UR will bypass the **in-doubt** state and transition directly into **in-commit** or **in-backout**.

When the SDSRM calls ATRADCT1 or ATR4ADCT, if *commit_options* indicated that the SDSRM's interest is to be removed, RRS deletes the SDSRM's interest and lets other resource manager(s) take responsibility for making the commit or backout

decision. If there is only one other Resource Manager with a single expression of interest, and it provides an Only_Agent exit, RRS will drive its Only_Agent exit. When the Only_Agent exit returns control, RRS considers the UR processing to be complete. If there is more than one Resource Manager or the only Resource Manager did not provide an Only_Agent exit, RRS will perform a two-phase commit processing for the UR, and commit or backout the UR based on the collective prepare vote. In this case, UR will bypass the **in-doubt** state and transition directly into **in-commit** or **in-backout**.

A successful call to Delegate_Commit_Agent_UR changes the UR state to either **in-forget** or **forgotten**, depending on the value of *log_option* specified on the call. If the return code is ATR_PROGRAM_STATE_CHECK, the UR state will remain unchanged.

RRS will implicitly change the *log_option* to ATR_DEFER_EXPLICIT under any of the following conditions:

- When the application backs out the UR through a call to the Backout_UR service or the Application_Backout_UR service.
- When RRS recreates a committed or backed-out UR during restart processing.

If any of these conditions has occurred, RRS returns the ATR_UR_STATE_ERROR return code. The UR might be in any state, but once it reaches **in-forget**, it will remain in that state until the Forget_Agent_UR service is called.

RRS waits for Forget_Agent_UR to ensure that the resource manager that has taken the SDSRM role is always informed of the results of the UR. This allows the resource manager to safely prevote its BACKOUT and COMMIT exits.

With ATRADCT1 or ATR4ADCT, if the Delete Interest option has been requested and a State Check Exit returns the ATRX_STATE_INCORRECT return code, the syncpoint will be backed out because the SDSRM's interest was deleted prior to the beginning of the syncpoint processing.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRADCT, ATRADCT1) 64 bit (ATR4ADCT)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	All parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL Definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

To use this service:

- The resource manager state must be **run**, which means that the UR interest token specified in the call has registered, set its exit routines with RRS, and completed restart.
- The unit of recovery state must be **in-flight**. With ATRADCT, after the call, subsequent references to the UR interest token will cause a logic error, unless the *log_option* was specified or changed to ATR_DEFER_EXPLICIT.

Caution: The resource manager must ensure that no application can be updating protected resources for the unit of recovery being committed. This is necessary to ensure that no resource manager taking part in the unit of recovery sees updates being made on behalf of a unit of recovery at the same time the updates are executing syncpoint processing.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

CALL ATRADCT	(return_code ,ur_interest_token ,log_option);

CALL ATRADCT1	(return_code ,ur_interest_token ,log_option ,commit_options);

CALL ATR4ADCT	(return_code ,ur_interest_token ,log_option ,commit_options);

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Delegate_Commit_Agent_UR service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that uniquely represents an instance of the resource manager's interest in the particular UR. This must be an interest in which the resource manager has taken the SDSRM role for the UR. The resource manager received the token from the Express_UR_Interest service or the Retain_Interest service.

,log_option
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates how RRS is to process log entries for the UR.

With ATRADCT1, if *commit_options* indicates to remove the SDSRM's interest, RRS ignores the *log_option* parameter.

Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
X'0' (0) ATR_DEFER_IMPLICIT	**Meaning:** RRS is to logically delete the log record when the UR state changes to **forgotten**. Your resource manager will not call the Forget_Agent_UR service.
X'1' (1) ATR_DEFER_EXPLICIT	**Meaning:** If the UR is committed, RRS must keep the log record for the UR until your resource manager calls the Forget_Agent_UR_Interest service. The *log_option* specified on Forget_Agent_UR_Interest then determines how RRS processes the log entry for committed URs. The Delegate_Commit_Agent_UR return codes document which return codes require your resource manager to call Forget_Agent_UR, as indicated by the final state of the UR being **in-forget**.

,commit_options

Supplied parameter

- Type: Bit string
- Character Set: N/A
- Length: 4 bytes

Specifies various options which determine how RRS is to perform the *delegated_commit* request. Each of the bits or set of bits in *commit_options* is either reserved or has a specific meaning. Each reserved bit must be specified as zero. Each of the other bits can be specified as zero or one.

The bit specifications are:

Bit Position	Constant in: Hexadecimal Equate Symbol	Description
0	00000000 ATR_STANDARD_COMMIT_MASK	When zero is specified, the SDSRM wants RRS to perform a normal delegated commit processing.

Bit Position	Constant in: Hexadecimal Equate Symbol	Description
0	10000000 ATR_REMOVE_SDSRM_INTEREST_MASK	When one is specified, the SDSRM wants RRS to remove its interest in the UR and let other resource manager(s) take responsibility for making the commit or backout decision. In this case, RRS ignores the log_option value.
1-31	0 (None)	Reserved

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00220000' or X'00220001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. All protected resources advanced to a consistent state. The UR state is now **in-forget** if you specified a *log_option* of ATR_DEFER_EXPLICIT, or **forgotten** if you specified a *log_option* of ATR_DEFER_IMPLICIT. **Action**: Continue normal processing.
8 ATR_FORGET	**Meaning**: The commit operation completed successfully. The collective prepare vote allows the unit of recovery to be completed, and all resource managers voted to abstain or forget. The UR state is now **forgotten**. **Action**: Continue normal processing. **Note:** This return code is not valid when the *Delete_SDSRM_Interest* option has been specified.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
65 ATR_COMMITTED_OUTCOME_PENDING	**Meaning**: The commit operation completed. The RRS decision was to advance to a consistent state. However, the state of one or more of the protected resources is not known. The UR state is now **in-forget** if you specified a *log_option* of ATR_DEFER_EXPLICIT, or **forgotten** if you specified a *log_option* of ATR_DEFER_IMPLICIT. **Action**: Continue normal processing for a committed UR.
66 ATR_COMMITTED_OUTCOME_MIXED	**Meaning**: The commit operation completed. The RRS decision was to advance to a consistent state. However, the state of one or more of the protected resources has been returned to the previous consistent state. The UR state is now **in-forget** if you specified a *log_option* of ATR_DEFER_EXPLICIT, or **forgotten** if you specified a *log_option* of ATR_DEFER_IMPLICIT. **Action**: Report the error to the other transactional participants.
C8 ATR_PROGRAM_STATE_CHECK	**Meaning**: The commit operation failed. The consistency state of the protected resources has not been altered. This return code indicates one of the following conditions has occurred: • A protected resource, specifically a communications Interface conversation, is not in **Send, Send Pending, Defer Receive, Defer Allocate, Sync_Point, Sync_Point Send,** or **Sync_Point Deallocate** state. • A protected resource, specifically a Communications Interface conversation, is in **Send** state, and the program started but did not finish sending a basic conversation logical record. • A protected resource, specifically a local resource, is not in the proper state for a commit. The UR state is unchanged. **Action**: If possible, initiate a resource manager action to get the resources to a committable state and then invoke the Delegate_Commit_Agent_UR service again. Otherwise, issue Backout_Agent_UR to back out the transaction.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: The caller is disabled. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: The caller is holding one or more locks. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: The system level does not support this service. The system rejects this service request. **Action**: Remove the calling program from the system, and install it on a system that supports RRS. Then rerun the calling program.
12C ATR_BACKED_OUT	**Meaning**: The commit operation failed. All protected resources have been returned to the previous consistent state. The UR state is now **forgotten**. **Action**: Continue normal processing for a backed out unit of recovery.
12D ATR_BACKED_OUT_OUTCOME_ PENDING	**Meaning**: The commit operation failed. The RRS decision was to return to the previous consistent state. However, the state of one or more of the protected resources is not known. The UR state is now **forgotten**. **Action**: Continue normal processing for a backed out unit of recovery.
12E ATR_BACKED_OUT_OUTCOME_ MIXED	**Meaning**: The commit operation failed. However, one or more of the protected resources has advanced to a new synchronization state. The UR state is now **forgotten**. **Action**: Report the error to the other transactional participants.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
370 ATR_URI_TOKEN_INV	**Meaning**: The specified *UR_interest_token* does not represent a valid expression of interest. This condition can occur after RRS has terminated and restarted. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
395 ATR_LOG_OPT_INV	**Meaning**: The specified *log_option* value is not valid. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
3AE ATR_COMMIT_OPTIONS_INV	**Meaning**: The specified *commit_options* value is not valid. Either reserved bits were nonzero or an unacceptable selection of options and parameters was specified. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
701 ATR_RM_STATE_ERROR	**Meaning**: The resource manager state is not valid for this request. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
702 ATR_RM_EXITS_UNSET	**Meaning**: RRS has unset the RRS exit routines for this resource manager. The system rejects this service request. **Action**: The resource manager must reset its RRS exit routine information and begin restart processing with RRS.
731 ATR_UR_STATE_ERROR	**Meaning**: The UR state is not valid for this service request. The system rejects the request. The application might have already requested backout. Call Retrieve_UR_Data or Retrieve_Side_Information to obtain information about the state of the UR. If you receive this return code, you must call Forget_Agent_UR to complete processing for the UR. **Action**: Call Forget_Agent_UR to complete the processing of this UR.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
74A ATR_NOT_SERVER_DSRM	**Meaning**: The resource manager does not have the server distributed syncpoint resource manager role for the unit of recovery. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
762 ATR_PRESUMED_NOTHING_INVALID	**Meaning**: The specified UR interest has an invalid two-phase commit protocol selected. PRESUMED_NOTHING is not allowed. **Action**: Check the calling program for a probable coding error.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. The system rejects the service request. **Action**: Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F04 ATR_UNEXPECTED_UR_ERROR	**Meaning**: System error. While processing the UR, RRS has encountered an unexpected error that might have damaged the UR. The system rejects the service call. **Action**: Contact the system programmer who maintains RRS at your installation. Manual intervention might be needed to restore consistent resources.
F06 ATR_WAS_NOT_AVAILABLE	**Action**: Refer to the documentation of this error under the description of the *Delete_UR_Interest* service.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: This service routine encountered an unexpected error. The system rejects this service request. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to tell RRS to initiate and complete a syncpoint operation for a UR. Storage for the call parameters has been allocated.

```
⋮
URI_TOKEN = MY_URI_TOKEN
LOGOPT = ATR_DEFER_IMPLICIT
```

```
CALL ATRADCT(RC,URI_TOKEN,LOGOPT)
⋮
```

Delete_Post_Sync_PET (ATRDPSP2, ATR4DPSP)

- ATRDPSP2 is for AMODE(31) callers.
- ATR4DPSP is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A work manager calls the Delete_Post_Sync_PET service to inform RRS that it no longer needs to know about syncpoint completion through a pause element token (PET) set by the Set_Post_Sync_PET service.

Environment

The requirements for the caller are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRDPSP2) 64 bit (ATR4DPSP)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	MVS standard linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

For the call, the UR state must be **in-reset** or **in-flight**.

When the resource manager issues the call in SRB mode, the call cannot specify a *ur_token* of 0, indicating information for the current UR.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRDPSP2	(return_code ,UR_token ,pause_element_token)

CALL ATR4DPSP	(return_code ,UR_token ,pause_element_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Delete_Post_Sync_PET service.

,UR_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token of the UR to which the PET specified by *pause_element_token* is associated:

- 0: Binary zero specifies the current UR associated with the application's task.
- *token*: The UR token of a particular UR.

,pause_element_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the pause element token to be disassociated from the UR specified by the *UR_token*.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00210000' or X'00210001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The calling program is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
104 ATR_MODE_INV	**Meaning**: Program error. The calling program specified 0 in *UR_token*, indicating the context associated with the current UR, but the calling program is not in task mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The calling program is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that is running a version of RRS that supports this service call. Then rerun the resource manager.
3A3 ATR_UR_TOKEN_INV	**Meaning**: Program error. The UR token specified in the *UR_token* parameter is not valid. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A6 ATR_PET_INV	**Meaning**: Program error. The pause element token specified in the *pause_element_token* parameter is not valid. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A7 ATR_PET_OUTDATED	**Meaning**: Program error. The pause element token specified in the *pause_element_token* parameter is outdated. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
3A8 ATR_PET_AUTH_FAILURE	**Meaning**: Program error. The pause element token specified in the *pause_element_token* parameter was allocated with *auth_level*=IEA_AUTHORIZED, and the caller is unauthorized. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A9 ATR_PET_SPACE_FAILURE	**Meaning**: Program error. The pause element token specified in the *pause_element_token* parameter represents a pause element belonging to another address space, and the caller is unauthorized. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3AA ATR_PET_NOT_ASSOCIATED	**Meaning**: Program error. The pause element token specified in the *pause_element_token* parameter does not represent a pause element associated with the specified UR. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The UR specified by the *UR_token* is not in a valid state for the service call. The UR must be in an **in-reset** or **in-flight** state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Delete_Post_Sync_PET was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the calling program attempts to disassociate a UR from its pause element token. Storage for the call parameters has already been allocated.

```
⋮
CALL ATRDPSP2(RC, URTOKEN, PETOKEN)
⋮
```

Delete_UR_Interest (ATRDINT, ATR4DINT)

- ATRDINT is for AMODE(31) callers.
- ATR4DINT is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Delete_UR_Interest service to delete an interest in a unit of recovery (UR). Note that RRS normally deletes the interest when the UR is complete.

In response to the Delete_UR_Interest call, RRS issues a return code.

The call deletes only one interest in the UR; if there are other interests, they continue to exist. For example, multiple resource managers might have expressed interest in the UR, or your resource manager might have issued the Express_UR_Interest call multiple times for this UR.

If the expression of interest to be deleted is the last expression of interest in a UR that is in local transaction mode, and the transaction was implicitly started (no Begin_Transaction was issued to demarcate the beginning of the local transaction), then the unit of recovery is completed and returned to **in-reset** state. When the transaction is completed, any post sync PETs that were set for this UR are released.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRDINT) 64 bit (ATR4DINT)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The UR state must be **in-flight**. After the call, any subsequent reference to the UR interest token causes a logic error.

The resource manager associated with the UR interest token specified in the call must be in **run** state, which means it has registered, set its exit routines with RRS, and completed restart.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRDINT	(return_code ,ur_interest_token)

CALL ATR4DINT	(return_code ,ur_interest_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Delete_UR_Interest service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that uniquely represents an instance of the resource manager's interest in the particular UR. Your resource manager received the token from the Express_UR_Interest service or the Retain_Interest service.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00010000' or X'00010001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call is not for one of the currently valid interests. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager must be in run state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: The resource manager must reset its RRS exits and begin restart processing with RRS.
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The UR is not in a valid state for the service call. The UR state must be **in-flight**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Delete_UR_Interest was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to delete its interest in a UR.

```
⋮
URI_TOKEN = MY_URI_TOKEN
CALL ATRDINT(RC,URI_TOKEN)
⋮
```

End_Restart (ATRIERS, ATR4IERS)

- ATRIERS is for AMODE(31) callers.
- ATR4IERS is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the End_Restart service to complete resource manager restart. Before calling End_Restart, your resource manager calls the

Retrieve_UR_Interest service repetitively until your resource manager obtains all of its interests in incomplete protected units of recovery (URs).

In response to the call, RRS issues a return code.

After the call, your resource manager can express interest in URs and take part in processing URs.

The call to End_Restart notifies RRS that it can begin to invoke exit routines for your resource manager.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRIERS)
	64 bit (ATR4IERS)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

To call the service, the resource manager associated with the resource manager token specified in the call must be in **restart** state. After a successful call, the resource manager is in **run** state.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRIERS	(return_code ,resource_manager_token)

CALL ATR4IERS	(return_code ,resource_manager_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the End_Restart service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00040000' or X'00040001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
301 ATR_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the resource manager token specified in the call is not in a valid state to issue the service call. The resource manager must be in **restart** state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: The resource manager must reset its RRS exits and begin restart processing with RRS.
73A ATR_RESTART_INCOMPLETE	**Meaning**: Program error. The resource manager has not obtained all of its incomplete protected URs. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. The resource manager must call the Retrieve_UR_Interest service repetitively until it obtains all incomplete URs that it had expressed interest in. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to end its restart. Storage for the call parameters has been allocated.

```
⋮
RM_TOKEN = MY_RM_TOKEN
CALL ATRIERS(RC,RM_TOKEN)
⋮
```

End_Transaction (ATREND, ATR4END)

- ATREND is for AMODE(31) callers.
- ATR4END is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

End_Transaction commits or rolls back (backs out) the current transaction. An application program calls the End_Transaction service to indicate that the changes for the UR are either to be made permanent (committed) or undone (rolled back). To process the call, RRS informs the resource managers about the specified action, then issues a return code to the calling program.

End_Transaction performs the same service as the following services:

- Application_Commit_UR
- Commit_UR
- Application_Backout_UR
- Backout_UR

End_Transaction, however, provides return codes for many error conditions that cause Application_Commit_UR and Application_Backout_UR to abnormally end the calling program with ABEND code X'5C4'. For a description of Application_Commit_UR and Application_Backout_UR, see *z/OS MVS Programming: Callable Services for High-Level Languages*.

Typically, End_Transaction is called in local transaction mode when an application or work manager needs to ensure that all uncommitted local resources are placed in a consistent state, with all changes either committed or backed out, but not necessarily in an atomic manner with respect to each other. When the specified action is commit, and the transaction mode of the current UR is global or hybrid-global, then this service performs identically to the Commit_UR service. Similarly, when the specified action is rollback, and the transaction mode of the current UR is global, then this service performs identically to the Backout_UR service.

Environment

The requirements for the caller are:

Minimum authorization:	Any
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATREND) 64 bit (ATR4END)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The UR state must be **in-reset** or **in-flight**.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATREND	(return_code ,diag_area ,action ,current_ur_token)

CALL ATR4END	(return_code ,diag_area ,action ,current_ur_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the End_Transaction service.

,diag_area
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 32 bytes

Contains diagnostic data from End_Transaction to help IBM Service determine the cause of an End_Transaction failure. Be sure to log this data when recording any information about an End_Transaction failure.

,action
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates whether the uncommitted resources associated with the current UR are to be committed or rolled back. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
1 (1) ATR_COMMIT_ACTION	The resource managers are to commit any uncommitted resources associated with the current UR.

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
2 (2) ATR_ROLLBACK_ACTION	The resource managers are to roll back any uncommitted resources associated with the current UR.

,current_ur_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR token that uniquely represents the current UR. Specify this token when you want RRS to verify that the UR specified is the current UR before performing any operation against it.

Specify binary zeros to indicate that RRS is to perform the requested action against the current UR, regardless of what UR is current.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00240000' or X'00240001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. All protected resources advanced to a consistent state. If you specified an *action* of ATR_COMMIT, all protected resources have been successfully committed. If you specified an *action* of ATR_ROLLBACK, all protected resources have been successfully rolled back. **Action**: Continue normal processing.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
65 ATR_COMMITTED_OUTCOME_PENDING	**Meaning**: Environmental error. The commit operation completed. However, RRS cannot determine if all of the protected resources were changed. **Action**: The action by the calling program depends on the system environment. Some possible actions are: • Display a warning message to the end user. • Write an exception entry into an output log. • Abnormally end the application because the resource manager will not allow any further changes to the resource until the situation is resolved.
66 ATR_COMMITTED_OUTCOME_MIXED	**Meaning**: Environmental error. The commit operation completed. However, one or more of the protected resources were not changed. **Action**: The action by the calling program depends on the system environment. Some possible actions are: • Display a warning message to the end user. • Write an exception entry into an output log. • Abnormally end the application because the resource manager will not allow any further changes to the resource until the situation is resolved.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
C8 ATR_PROGRAM_STATE_CHECK	**Meaning**: Environmental error. The commit operation failed. The resource managers may have rejected the commit because one of the following occurred: • A communications interface conversation that is a protected resource is not in a required state: **send, send pending, defer receive, defer allocate, sync_point, sync_point send,** or **sync_point deallocate** state. • A protected communications interface conversation is in **send** state. The program started sending the basic conversation logical record, but did not finish sending it. • A resource on the same system as the application is not in a proper state for the commit. **Action**: Initiate an action by a resource manager to get its resource to a commitable state. Issue the call again. **Note:** This code is returned only when transaction mode is global or hybrid-global.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enable for I/O and external interrupts. The system rejects this service request. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The caller is holding one or more locks; no locks must be held. The system rejects this service request. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system level does not support this service. The system rejects this service request. **Action**: Remove the calling program from the system, and install it on a system that supports this level of RRS. Then rerun the calling program.
12C ATR_BACKED_OUT	**Meaning**: Environmental error. The commit operation failed. All protected resources have been returned to the previous consistent state. **Action**: The action by the calling program depends on the system environment. Some possible actions are: • Display a warning message to the end user. • Write an exception entry into an output log. • Abnormally end the application because the resource manager will not allow any further changes to the resource until the situation is resolved.
12D ATR_BACKED_OUT_OUTCOME_ PENDING	**Meaning**: Environmental error. The commit or rollback operation failed. The RRS decision was to return to the previous consistent state. However, the state of one or more of the protected resources is not known. **Action**: The action by the calling program depends on the system environment. Some possible actions are: • Display a warning message to the end user. • Write an exception entry into an output log. • Abnormally end the application because the resource manager will not allow any further changes to the resource until the situation is resolved.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
12E ATR_BACKED_OUT_OUTCOME_ MIXED	**Meaning**: Environmental error. The commit or rollback operation failed. The RRS decision was to return to the previous consistent state. However, one or more of the protected resources has changed. **Action**: The action by the calling program depends on the system environment. Some possible actions are: • Display a warning message to the end user. • Write an exception entry into an output log. • Abnormally end the application because the resource manager will not allow any further changes to the resource until the situation is resolved.
3A3 ATR_UR_TOKEN_INV	**Meaning**: Program error. The specified *current_UR_token* does not identify a valid UR. The system rejects this service request. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
36B ATR_ACTION_INV	**Meaning**: Program error. The specified *action* value is not valid. The system rejects this service request. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The UR for the current task is not in a valid state for this service request. The UR state must be **in-reset** or **in-flight**. The system rejects the request. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
805 ATR_CUR_UR_TOKEN_ NOT_CURRENT	**Meaning**: Program error. The current UR token specified in the call does not match the token of the current UR. **Note:** The system takes no action against the current UR, so it is therefore possible that the current UR can be committed at a later time. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. The system rejects the service request. **Action**: Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F04 ATR_UNEXPECTED_UR_ERROR	**Meaning**: System error. While processing the UR, RRS has encountered an unexpected error that might have damaged the UR. The system rejects the service call. **Action**: Contact the system programmer who maintains RRS at your installation. Manual intervention might be needed to restore consistent resources.
F05 ATR_UNEXPECTED_CTX_ERROR	**Meaning**: Environmental error. The service call encountered an unexpected error from a context services service. The system rejects the service call. **Action**: Examine the dump from context services and correct the problem, then rerun the calling program.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: This service routine encountered an unexpected error. The system rejects this service request. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, an application issues an End_Transaction call to tell RRS to commit a transaction. Storage for the call parameters has been allocated.

```
⋮
ACT = ATR_COMMIT
CUR = MY_CURRENT_UR
CALL ATREND(RC,DIAG_DATA,ACT,CUR)
⋮
```

Express_UR_Interest (ATREINT, ATREINT1, ATREINT2, ATREINT3, ATREINT4, ATREINT5, ATR4EINT)

A resource manager calls the Express_UR_Interest service to express an interest, either protected or unprotected, in a unit of recovery (UR). There are seven versions of Express_UR_Interest, each with different parameters.

- ATREINT is for AMODE(31) callers and is the basic version of the service.
- ATREINT1 is for AMODE(31) callers and adds support for XIDs.
- ATREINT2 is for AMODE(31) callers and supports XIDs and cascaded transactions.

- ATREINT3 is for AMODE(31) callers, supports XIDs and returns transaction mode information.
- ATREINT4 is for AMODE(31) callers, supports XIDs, cascaded transactions, COMMIT exit tier priority and returns transaction mode information.
- ATREINT5 is for AMODE(31) callers, supports XIDs, cascaded transactions, COMMIT exit tier priority and returns transaction mode information. ATREINT5 uses the *interest_options* parameter to specify various options that determine how RRS will process an interest. Earlier versions of the service use multiple parameters to specify specific options.
- ATR4EINT is for AMODE(64) callers, allows parameters in 64 bit addressable storage, supports XIDs, cascaded transactions, returns transaction mode information, and COMMIT exit tier priority. ATREINT5 uses the *interest_options* parameter to specify various options that determine how RRS will process an interest. Earlier versions of the service use multiple parameters to specify specific options.

Code your resource manager to call the version that includes the support you need.

In response to the different versions of the call, RRS can return:

- A return code.
- A UR interest token for the interest. You need this token on many other RRS calls to identify a specific UR.
- The current context token, if you specified binary zeros on *context_token* to indicate that RRS is to use the current context associated with the current dispatchable unit.
- A UR identifier (URID).
- A UR token. You need this token if you want to create a UR cascaded from the UR in which you are expressing interest. You can also use this token as input for some services instead of using a URI token.
- If you make a conditional request and your resource manager already has an interest in the UR, the current nonpersistent interest data.
- An indicator of the type of transaction in which interest has been expressed. The indicator will indicate one of the following:
  - Local transaction mode: The transaction in which interest has been expressed is in local transaction mode. The resource manager should commit the resources it is managing for the transaction when asked to do so explicitly by the transaction or when told to do so by RRS.

    Note: A resource manager will only be allowed to express interest in a transaction that is in local transaction mode if it has indicated, through Set_Exit_Information, that it can participate in local mode transactions.
  - Global transaction mode: The transaction in which interest has been expressed is in global transaction mode. The resource manager may commit its resources without involving RRS as long as it is the only resource manager involved in the transaction.
  - Hybrid-global transaction mode: The transaction in which interest has been expressed is in hybrid-global transaction mode. The resource manager may commit its resources without involving RRS as long as it is the only resource manager involved in the transaction.

Note: Hybrid-global transaction mode is the mode that all RRS managed transactions ran under prior to RRS support for local and global modes. See "Local transactions" on page 73 for more information about local and global transaction modes.

When a UR involves changes to multiple databases, communications, or both, then multiple resource managers might be interested in the UR and issue Express_UR_Interest calls for the same UR.

A single resource manager can also issue multiple Express_UR_Interest calls for the same UR, perhaps one for each of the resource manager's databases or one for each conversation being handled for the UR by a communication resource manager. Note, however, that expressing multiple interests in a UR causes RRS to invoke multiple exit routines. You can provide a shorter path length by expressing only one interest in a UR and keeping track of the resources for the UR in a control block that the resource manager maintains. In contrast, a communication resource manager might find multiple interests more useful, outweighing the overhead of multiple exit routine invocations.

To avoid creating multiple interests in the same UR, your resource manager can issue Express_UR_Interest as a conditional request; RRS creates a UR interest only when one does not already exist for this resource manager.

Protected and unprotected interests: The call can express a protected or unprotected interest in a UR. For a protected interest, RRS or a resource manager coordinates changes to the resources, so that all changes are made or no changes are made. Resources that can be protected are a database, a conversation between two communications managers, or a product-specific resource.

If an interest is a protected interest and the system, RRS, or the resource manager fails, RRS will inform the resource manager about the UR, if incomplete, when the resource manager restarts. The resource manager can then finish processing the UR.

Action for resource manager failure: On the Express_UR_Interest call, you can specify how RRS should process requests to commit the UR if your resource manager becomes:

- **Unregistered:** Your resource manager is no longer registered as a resource manager. See "Register_Resource_Manager (CRGGRM, CRG4GRM)" on page 137 for a description of how a resource manager can become unregistered.
- **Unset:** Your resource manager's exit routines are no longer set with RRS.

RRS should react to a resource manager failure as follows:

- **Standard processing:** RRS is to back out this UR, if the state of the UR is **in-reset**, **in-flight**, **in-state-check**, or **in-prepare**.
- **Forget interest:** RRS is to delete the resource manager's interest in the UR. You can specify this value only when *interest_type* or *interest_options* indicates that the interest is unprotected.

Action for subordinate system failure: On the Express_UR_Interest call, you can specify whether RRS should notify the coordinator UR of a sysplex cascaded transaction in the event of a failure of either RRS, any resource manager on the subordinate system, or the subordinate system itself:

- **Notify:** RRS will drive the SUBORDINATE_FAILED exit to notify the resource manager that there is a breakage on the subordinate system for which the

resource manager is the coordinator. RRS only drives this exit when the sysplex cascaded transaction was in-flight at the time of the failure.

- **Ignore:** SUBORDINATE_FAILED exit will not be driven.

Two-phase commit protocol: An Express_UR_Interest call can specify the type of two-phase commit protocol to be used for the UR if the resource manager is restarting:

- **Presumed nothing**: For a presumed nothing expression of interest in a protected UR, RRS hardens an **in-prepare** record, including the persistent interest data, in the RRS log before it invokes the PREPARE exit routines. If the last log record for a UR was an **in-prepare** record, RRS returns the UR as **in_backout** in response to a Retrieve_UR_Interest call during resource manager restart.

  If one protected interest in a UR is presumed nothing, RRS uses the presumed nothing protocol. If there is only one presumed nothing protected interest in a UR and this interest is by a distributed syncpoint resource manager, RRS does not log an **in-prepare** record.
- **Presumed abort**: When the UR state is **in-prepare**, RRS does not harden an **in-prepare** record in the RRS log. During restart, RRS cannot return such a UR in response to a Retrieve_UR_Interest call. The resource manager presumes the UR was backed out.

Automatic context termination: An Express_UR_Interest call can specify how RRS should process the work context associated with the UR when the UR is forgotten:

- **Standard processing**: No changes are made to the work context.
- **End processing**: RRS will end the work context when the UR is forgotten.

  Note: IBM strongly recommends that end processing only be specified by the resource manager that owns the work context.

XID processing: A resource manager can provide an XID for the UR in which interest is being expressed as long as the UR does not already have an XID assigned. The resource manager can tell RRS to use the XID with either:

- Standard Processing, or
- Use BQUAL without checking, and/or
- Use FormatID without checking

The possible results are as follows:

Table 42. XID Processing results

Standard processing:		
Interest being expressed in	**XID specified**	**RRS action**
An existing non-cascaded UR	No	The UR keeps the XID that it already has (if any).
An existing cascaded UR	No	The UR keeps the XID that it already has.
A new non-cascaded UR	No	The UR will not have any XID associated with it.
A new cascaded UR	No	The UR will be given an XID that has the same FormatID and GTRID as its parent and a new, unique BQUAL.

Table 42. XID Processing results (continued)

Standard processing:		
An existing non-cascaded UR	Yes	If the UR does not already have an XID, the specified XID will be associated with the UR. If the UR already has an XID, the call to Express_UR_Interest will be considered invalid and no expression of interest will be created.
An existing cascaded UR	Yes	The call to Express_UR_Interest will be considered invalid and no expression of interest will be created.
A new non-cascaded UR	Yes	The specified XID will be associated with the UR.
A new cascaded UR	Yes	If the specified XID has a FormatID and GTRID that are the same as its parent, the specified XID will be associated with the UR unless the caller request RRS not to check the specified FormatID. In this case, only the GTRID component will be validated.
Use BQUAL without checking:		
Interest being expressed in	**XID specified**	**RRS action**
A new cascaded UR	Yes	RRS will assign the new cascaded UR an XID that has the same FormatID and GTRID as its parent. The BQUAL portion of the XID will be taken from the XID specified. The FormatID and GTRID portions of the XID will be ignored.

Persistent interest data: In the Express_UR_Interest call, your resource manager can provide persistent interest data for the protected interest. When hardening information for the interest in an RRS log, RRS records the persistent interest data. Because the data is hardened, it will be available if your resource manager restarts or if RRS restarts, forcing your resource manager to restart.

Your resource manager can also provide persistent interest data in a call to: Change _Interest_Type, Set_Persistent_Interest_Data, or Retain_Interest. Your resource manager can retrieve persistent interest data in a call to: Retrieve_UR_Interest or Retrieve_Interest_Data.

Nonpersistent Interest Data: The Express_UR_Interest call can also provide nonpersistent interest data. RRS passes nonpersistent data to each resource manager exit routine it invokes for this interest. This data is not recorded in nonvolatile storage and is not available at subsequent restarts.

One use of this data is to pass the exit routines the locations of resource manager structures that represent the resources being changed.

Your resource manager can also provide nonpersistent interest data in a call to: Respond_to_Retrieved_Interest or Retain_Interest. Your resource manager can retrieve it by calling the Retrieve_UR_Interest_Data service.

URID: Save the returned UR identifier (URID) with the information about the UR in your resource manager log. During restart processing after your resource manager, RRS, or the system fails, your resource manager obtains the URID for an incomplete UR from a Retrieve_UR_Interest call. Compare the URID from Retrieve_UR_Interest with the URIDs in your resource manager log to find the data for the incomplete UR.

Your resource manager can also obtain the URID from a call to: Change_Interest_Type, Retrieve_UR_Interest, Retrieve_UR_Data, or Retain_Interest.

XID: If the UR does not already have an XID, the resource manager can specify an XID. If specified, the XID will be used as a work identifier for the UR. XIDs are not supported by the ATREINT version of Express_UR_Interest. You must call ATREINT1, ATREINT2, ATREINT3, ATREINT4, ATREINT5, or ATR4EINT to specify an XID.

Creating cascaded units of recovery: A work manager may use Express_UR_Interest to make a cascaded UR. A work manager would do this when a single work request involves multiple work managers that require separate contexts, or when separate transactional pieces of an application need to execute in parallel. By using the Express_UR_Interest service to make a new UR associated with a new work context and cascading it from the original UR, a second work manager ensures that all of the resources changed while it was executing in its original environment. Cascaded URs are not supported by the ATREINT, ATREINT1, and ATREINT3 versions of Express_UR_Interest. You must call ATREINT2, ATREINT4, ATREINT5 or ATR4EINT to make a cascaded UR while expressing interest. For additional information about cascaded URs, see "Create_Cascaded_UR (ATRCCUR2, ATRCCUR3, ATR4CCUR)" on page 268.

Multisystem cascaded units of recovery: When one work manager requests another work manager, residing on a different system, to become part of an existing work request, the requesting work manager is responsible for transferring all of the data needed by the new work manager, including the UR token representing the work request. The new work manager may then use the Express_UR_Interest service to create a new UR, associated with a new work context, to be cascaded from the received UR token.

As with normal (non-multisystem) cascaded transactions, a work manager that creates a multisystem cascaded transaction is responsible for informing RRS when the part of the application executing under a multisystem cascaded UR is complete by using the Set_Side_Information service to mark the UR as application-complete.

For additional information about multisystem cascaded transactions, see "Multisystem cascaded transactions" on page 70.

Local transactions: A resource manager may use Express_UR_Interest to express interest in a UR that is in local transaction mode. A resource manager must indicate that it supports local transactions via Set_Exit_Information before it can express interest in a local transaction mode UR. Calls to ATREINT3, ATREINT4,

ATREINT5, or ATR4EINT return information about the UR's transaction mode. For additional information about local transactions, see “Local transactions” on page 73.

Commit exit tier priority: On the Express_UR_Interest call, you can specify the tier priority at which RRS should invoke your COMMIT exit:

- **Tier one priority:** RRS will invoke the resource manager's COMMIT exit before other resource managers. If multiple resource managers request the tier one priority, the commit exits will be driven in the order in which they expressed interest in the UR.
- **No priority:** The resource manager's exit will be driven after tier one resource managers' commit exits, if any.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (ATREINT, ATREINT1, ATREINT2, ATREINT3, ATREINT4, ATREINT5) 64 bit (ATR4EINT)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

For the call, the UR must be in an **in-reset** or **in-flight** state. If the UR state is **in-reset**, a successful call changes the UR state to **in-flight**.

To call the service, the resource manager associated with the resource manager token specified in the call must be in **run** state.

When the resource manager issues the call in SRB mode, the call cannot specify a *context_token* of 0, indicating the current context.

Do not express interests in a given UR asynchronously. An asynchronous thread may be used to express interest in a given UR, but the contextual thread should be suspended until the completion of the service. Asynchronous expressions of

interest can cause work managers to make incorrect decisions about optimized commit flows, causing some resources to be committed and some left uncommitted.

Cascaded transaction restrictions:

The following restrictions apply when using ATREINT2, ATREINT4 or ATREINT5 to work with cascaded transactions:

- If either *UR_family_option* or *interest_options* indicates that a cascaded UR is to be created, the UR must be in the **in-reset** state.
- When the resource manager issues the call in SRB mode, the call must not specify binary zero for *context_token* or *parent_UR_token*. When either *UR_family_option* or *interest_options* indicates that a cascaded UR is to be created, the call can specify binary zero for either *context_token* or *parent_UR_token*, but not both.
- If either *UR_family_option* or *interest_options* indicates that a cascaded UR is to be created, an XID may be specified, but the specified XID must have the same FormatID and GTRID as the XID of the UR specified by *parent_UR_token*.
- To express an interest that creates a cascaded UR, the parent UR state must be **in-reset** or **in-flight** and the parent UR must not be in local transaction mode. After successfully expressing interest in a cascaded UR, both the parent and the child UR are in **in-flight** state and in global transaction mode.
- For multisystem cascaded transactions, *parent_UR_token* must specify a token from a system in the same logging group as this system.

Note: A call to the Retrieve_UR_Data service that does not specify ATR_EXTENDED_STATES for the *states_option* could cause a UR to go into an **in-flight** state. Once a UR has gone **in-flight**, it can no longer be made into a cascaded UR.

Local transaction restrictions

The following restrictions apply when working with local transactions:

- Before Express_UR_Interest can set a UR to local transaction mode, the resource manager must indicate to RRS, on a call to Set_Exit_Information, that the resource manager supports local transaction mode.
- URs in local transaction mode cannot participate in cascaded transactions.
- You cannot set an XID for a UR that is in local transaction mode.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged

14 Used as a work register by the system

15 Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

If possible, resource managers should use presume abort logging protocols in order to minimize log update activity.

Syntax

Write the appropriate call as shown in the syntax diagrams. You must code the parameters in the CALL statements as shown.

CALL ATREINT	(return_code ,resource_manager_token ,context_token ,ur_interest_token ,current_context_token ,ur_identifier ,multiple_interest_option ,interest_type ,failure_action ,two_phase_protocol ,nonpersistent_interest_data ,current_nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data)

CALL ATREINT1	(return_code ,resource_manager_token ,context_token ,ur_interest_token ,current_context_token ,ur_identifier ,multiple_interest_option ,interest_type ,failure_action ,two_phase_protocol ,nonpersistent_interest_data ,current_nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data ,xid_length ,xid)

CALL ATREINT2	(return_code ,resource_manager_token ,context_token ,ur_interest_token ,current_context_token ,ur_identifier ,multiple_interest_option ,interest_type ,failure_action ,two_phase_protocol ,nonpersistent_interest_data ,current_nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data ,xid_length ,xid ,ur_family_option ,parent_ur_token)

CALL ATREINT3	(return_code ,resource_manager_token ,context_token ,ur_interest_token ,current_context_token ,ur_identifier ,multiple_interest_option ,interest_type ,failure_action ,two_phase_protocol ,nonpersistent_interest_data ,current_nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data ,xid_length ,xid ,diag_area ,transaction_mode)

CALL ATREINT4	(return_code ,resource_manager_token ,context_token ,ur_interest_token ,current_context_token ,ur_identifier ,multiple_interest_option ,interest_type ,failure_action ,two_phase_protocol ,nonpersistent_interest_data ,current_nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data ,xid_length ,xid ,ur_family_option ,parent_ur_token ,diag_area ,transaction_mode)

CALL ATREINT5	(return_code ,resource_manager_token ,context_token ,ur_interest_token ,ur_token ,current_context_token ,ur_identifier ,interest_options ,nonpersistent_interest_data ,current_nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data ,xid_length ,xid ,parent_ur_token ,diag_area ,transaction_mode)

CALL ATR4EINT	(return_code ,resource_manager_token ,context_token ,ur_interest_token ,ur_token ,current_context_token ,ur_identifier ,interest_options ,nonpersistent_interest_data ,current_nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data ,xid_length ,xid ,parent_ur_token ,diag_area ,transaction_mode)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Express_UR_Interest service.

,resource_manager_token
Supplied parameter

- Type: Character string

- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

,context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token for the context associated with the UR:

- 0: Binary zeros specify the current context associated with the application's task. Binary zero may be specified for either *parent_UR_token* or *context_token*, but not both.
- token: The context token of a particular context.

Your resource manager might have received the *context_token* from the Begin_Context or Retrieve_Current_Context_Token services.

,ur_interest_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives the UR interest token that uniquely represents this instance of the resource manager's interest in the UR.

If you specified conditional interest via *multiple_interest_option* or *interest_options* and the return code from the service is `ATR_RM_ALREADY_HAS_INTEREST`, then the token returned represents the resource manager's existing interest in the UR. If there are multiple existing interests, the one returned is not predictable.

,ur_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

On ATREINT5 calls, receives the UR token that uniquely identifies the unit of recovery in which interest has been expressed.

If you specified conditional interest via *interest_options* and the return code from the service is `ATR_RM_ALREADY_HAS_INTEREST`, then the token returned represents the unit of recovery for which the URI token is returned in *URI_token*.

,current_context_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives the following from the service:

- The token of the current context, if the call specifies zeros in the *context_token* parameter. The token is a 16-byte character string.
- Undefined, if *context_token* specifies a token.

,ur_identifier
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives a UR identifier (URID) from the service. The URID uniquely identifies the UR.

,interest_options
Supplied parameter

- Type: Bit string
- Character Set: N/A
- Length: 4bytes

On ATREINT5 calls, specifies various options that determine how RRS will process this interest. Each of the bits in *interest_options* is either reserved or has a specific meaning. Each reserved bit must be specified as zero. Each other bit can be specified as either zero or one. The bit specifications are:

Bit positions	Constant in: Hexadecimal Equate Symbol	Description
0–2	0	Reserved
3	00000000 ATR_UNCOND_ INT_MASK 10000000 ATR_CONDITIONAL_ INT_MASK	**Multiple interests** A resource manager specifies zero to express unconditional interest in the UR. RRS is to create a new interest, even if the resource manager already has an interest in the UR. A resource manager specifies one to express conditional interest in the UR. RRS is not to create a new interest if the resource manager already has an interest in the UR. Instead, RRS should return information about the resource manager's existing interest. **Note:** If the resource manager has more than one interest, information about only one interest will be returned.
4–6	0	Reserved
7	00000000 ATR_UNPROT_ INT_MASK 01000000 ATR_PROTECTED_ INT_MASK	**Interest type** A resource manager specifies zero to express an unprotected interest in the UR. A resource manager specifies one to express a protected interest in the UR.
8–10	0	Reserved

Bit positions	Constant in: Hexadecimal Equate Symbol	Description
11	00000000 ATR_STANDARD_ FAIL_MASK 00100000 ATR_REMOVE_INT_ ON_FAIL_MASK	**Failure action** A resource manager specifies zero when it wants RRS to do its standard processing if the resource manager fails. A resource manager specifies one when it wants RRS to remove its interest in the UR if the resource manager fails. **Note:** One can only be specified if the resource manager is expressing an unprotected interest in the UR.
12–13	0	Reserved
14	00000000 ATR_COMMIT_ NO_PRIORITY 00020000 ATR_COMMIT_ TIER_ONE_ PRIORITY	**Commit exit tier priority** When zero is specified, the resource manager does not require RRS to drive its COMMIT exit at a higher priority with regards to other resource managers in the same UR. When one is specified, the resource manager wants RRS to drive its COMMIT exit first with respect to other resource managers' exits.
15	00000000 ATR_PRESUME_ NOTHING_MASK 00010000 ATR_PRESUME_ ABORT_MASK	**Two phase protocol** A resource manager specifies zero when it wants RRS to treat this expression of interest as needing **presume nothing** logging. A resource manager specifies one when it wants RRS to treat this expression of interest as needing **presume abort** logging.
16–18	0	Reserved
19	00000000 ATR_CREATE_ STANDARD_ UR_MASK 00001000 ATR_CREATE_ CASCADED_ UR_MASK	**UR family type** A resource manager specifies zero when it wants RRS to create a normal (non-cascaded) UR. A resource manager specifies one when it wants RRS to create a cascaded UR. *parent_UR_token* specifies the UR token of the parent of the new cascaded UR. **Note:** To specify one, the UR in which interest is being expressed must be in **in-reset** state.
20–22	0	Reserved

Bit positions	Constant in: Hexadecimal Equate Symbol	Description
23	00000000 ATR_DONT_END_ CONTEXT_MASK 00000100 ATR_END_ CONTEXT_MASK	**Auto context termination** A resource manager specifies zero when it does not want RRS to end the work context associated with the UR in which interest is being expressed when the UR completes. A resource manager specifies one when it wants RRS to end the work context associated with the UR in which interest is being expressed when the UR completes. **Note:** IBM strongly recommends that one only be specified by the resource manager that owns the work context.
24–26	0	Reserved
27	00000000 ATR_STANDARD_ XID_MASK 00000010 ATR_USE_ BQUAL_MASK	**XID processing** A resource manager specifies zero when it wants RRS to do its standard XID processing. A resource manager specifies one it wants RRS to assign the new UR an XID with the same FormatID and GTRID as its parent and a BQUAL equal to the BQUAL of the XID specified. **Note:** To specify one, the interest must be in a new cascaded UR and *XID* must be specified. RRS will only validate and use the BQUAL and BQUAL length fields in the specified XID.
28	00000000 ATR_STANDARD_ XID_MASK 00000008 ATR_USE_ FORMATID_MASK	**XID processing** A resource manager specifies zero when it wants RRS to do its standard XID processing. A Resource manager specifies one when it wants RRS to assign the new UR an XID with the same GTRID as its parent and a FormatID equal to the FormatID of the XID specified. **Note:** To specify one, the interest must be in a new cascaded UR and *XID* must be specified. RRS will use the FormatID in the specified XID.
29-30	0	Reserved.

Bit positions	Constant in: Hexadecimal Equate Symbol	Description
31	00000000 ATR_IGNORE_ SUBORDINATE_ FAILURE_MASK	Subordinate Failure Action When zero is specified, the resource manager does not want RRS to drive the SUBORDINATE_FAILED exit.
	00200000 ATR_NOTIFY_ SUBORDINATE_ FAILURE_MASK	When 1 is specified, the resource manager wants RRS to drive its SUBORDINATE_FAILED exit to inform the resource manager that RRS or a resource manager on its subordinate system has failed or the subordinate system itself has terminated. The sysplex cascaded transaction for which this resource manager is the coordinator was in-flight at the time of the failure.

,multiple_interest_option

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates whether or not RRS is to create a new interest when the resource manager already has an interest in the UR. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) ATR_UNCONDITIONAL	**Unconditional:** RRS is to create a new interest, even when the resource manager already has an interest in the UR.
1 (1) ATR_CONDITIONAL	**Conditional:** RRS is **not** to create a new interest when the resource manager already has an interest in the UR.

,interest_type

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates the type of interest the resource manager has in the UR. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) ATR_UNPROTECTED	**Unprotected:** The resource manager is expressing an unprotected interest in the UR.
1 (1) ATR_PROTECTED	**Protected:** The resource manager is expressing a protected interest in the UR.

When expressing interest in a UR in local transaction mode, RRS will not harden any data for a local transaction; therefore, the processing for both ATR_UNPROTECTED and ATR_PROTECTED is identical. You can specify either.

,failure_action
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Defines how RRS is to process commit requests for the UR if the resource manager becomes unregistered or unset. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Action
0 (0) ATR_FAIL_STANDARD	Standard processing
2 (2) ATR_FAIL_FORGET	Forget interest

,two_phase_protocol
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Identifies the two-phase commit protocol RRS is to use for this UR. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Protocol
0 (0) ATR_PRESUMED_NOTHING	Presumed nothing
1 (1) ATR_PRESUMED_ABORT	Presumed abort

When expressing interest in a UR in local transaction mode, RRS will not harden any data for a local transaction; therefore, the syncpoint processing for both ATR_PRESUMED_NOTHING and ATR_PRESUMED_ABORT is identical. You can specify either.

,nonpersistent_interest_data
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the nonpersistent interest data for your resource manager's interest. RRS does not record this data in nonvolatile storage.

,current_nonpersistent_interest_data
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives from the service the current nonpersistent interest data when both of the following conditions occur together:

- You specified conditional interest via *multiple_interest_option* or *interest_options*.
- The return code is ATR_RM_ALREADY_HAS_INTEREST.

Otherwise, the field contains binary zeros.

The nonpersistent interest data is for the resource manager's interest represented by the UR interest token returned by the service.

,persistent_interest_data_length
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies, in hexadecimal, the length of the persistent interest data. Specify X'0' - X'1000' (0-4096) bytes. If either *interest_options* or *interest_type* indicates that the interest is unprotected, then this field must be binary zeros.

,persistent_interest_data
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified in *persistent_interest_data_length*

The persistent interest data for your resource manager's interest in the UR. RRS records this data in an RRS log. If *persistent_interest_data_length* is binary zeros, RRS ignores this parameter.

When expressing interest in a UR in local transaction mode, RRS will not harden any data for a local transaction, including persistent interest data. If the caller knows a UR is in local transaction mode, it should set *persistent_interest_data_length* to binary zeros. If the calling program does not know the transaction mode of the UR, however, it should specify all of the data needed for any mode. It is not considered an error if a nonzero value is specified in local transaction mode.

,xid_length
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

On ATREINT1, ATREINT2, ATREINT3, ATREINT4 and ATREINT5 calls, specifies the length of the XID specified by *xid* as follows:

- 0: Binary zero specifies that no XID is provided. RRS ignores the contents of the *xid* parameter.
- non-zero: The length of the value in the *xid* parameter. The value must be between ATR_MIN_XID_LENGTH (13) and ATR_MAX_XID_LENGTH (140), inclusive. This parameter is normally set to a non-zero value only by a communications resource manager that has received an inbound transactional request with an XID provided. To specify an XID, the UR state must be **in-reset**. After a resource manager successfully uses ATREINT1, ATREINT2, ATREINT3 or ATREINT4 to set an XID, the transaction mode is set to global.

,xid
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified by the *xid_length* parameter

On ATREINT1, ATREINT2, ATREINT3, ATREINT4 and ATREINT5 calls, specifies the XID your resource manager wants to set for this unit of recovery. If *xid_length* is zero, the contents of this parameter are ignored.

An XID has the following format:

FORMATID
4–byte format identifier

GTRID_length
4–byte GTRID length

BQUAL_length
4–byte BQUAL length

ID 1–128 byte ID transaction identifier

The 1–128 byte ID field has the following format:

GTRID
1–64 byte GTRID

BQUAL
0–64 byte BQUAL

,ur_family_option
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

On ATREINT2 and ATREINT4 calls, indicates whether or not RRS is to make the UR in which the resource manager expresses interest into a cascaded UR. The parent UR is specified by the *parent_UR_token*. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) ATR_NO_FAMILY	**No family**: The UR is not a cascaded UR.
1 (1) ATR_CASCADED	**Cascaded**: The UR is a cascaded UR. The parent UR is specified by *parent_UR_token*. This must be the first expression of interest in the UR.

,parent_UR_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

On ATREINT2, ATREINT4 and ATREINT5 calls, specifies the token of the UR that is to be the parent of the UR specified by the *context_token*:

- 0: Binary zero specifies the current UR associated with the application's task active on the current system. Binary zero may be specified for either the *parent_UR_token* or the *child_context_token*, but not both.
- *token*: The UR token of a particular UR. The UR token may be from another system in the same logging group.

Unless *UR_family_option* is ATR_CASCADED or *interest_options* indicates that the interest being expressed is being used to create a new cascaded UR, RRS ignores this parameter.

Your resource manager may have received the *parent_UR_token* from the Retrieve_UR_Data service or from a work manager from another system. If the UR token was received from another system, RRS will associate the new child UR, specified by *child_context_token*, with the top-level UR of the cascaded UR family that resides on the system where the work request originated.

,diag_area
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 32 bytes

On ATREINT3, ATREINT4 and ATREINT5 calls, contains diagnostic data from Express_UR_Interest to help IBM service determine the cause of an Express_UR_Interest failure. Be sure to log this data when recording any information about an Express_UR_Interest failure.

,transaction_mode
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

On ATREINT3 and ATREINT4 calls, indicates the transaction mode of the UR that the resource manager is expressing interest in. RRS returns one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	**Description**
1 (1) ATR_GLOBAL_MODE	The transaction mode for the UR is global.
2 (2) ATR_LOCAL_MODE	The transaction mode for the UR is local.
3 (3) ATR_HYBRID_GLOBAL_MODE	The transaction mode for the UR is hybrid-global.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00020000' or X'00020001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
8 ATR_RM_ALREADY_HAS_INTEREST	**Meaning**: For a conditional request, the resource manager already has an interest in the UR. The system accepts the service call and returns: • The UR interest token • The UR identifier • The nonpersistent interest data for the existing interest The system might also return the current context token. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program specified 0 in *context_token* or *parent_UR_token*, but the calling program is not in task mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that is running a version of RRS that supports this service call. Then rerun the resource manager.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
301 ATR_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
361 ATR_CONTEXT_TOKEN_INV	**Meaning**: Program error. The context token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
36A ATR_DU_TERMINATING	**Meaning**: Environmental error. The task or SRB associated with the context specified in the call is ending. The system rejects the service call. **Action**: None.
371 ATR_INTEREST_TYPE_INV	**Meaning**: Program error. The *interest_type* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
372 ATR_FAILURE_ACTION_INV	**Meaning**: Program error. The *failure_action* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
375 ATR_TWO_PHASE_PROTOCOL_INV	**Meaning**: Program error. The *two_phase_protocol* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
376 ATR_PERSISTENT_DATA_LEN_INV	**Meaning**: Program error. The length specified in the *persistent_interest_data_len* parameter in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
386 ATR_FAILURE_ACTION_ INCORRECT	**Meaning**: Program error. The failure action specified in the call is not valid with the specified interest type. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
389 ATR_PERSISTENT_DATA_NOT_ ALLOWED	**Meaning**: Program error. The persistent interest data length specified in the call is not zero; zero is the only value valid when either *interest_type* or *interest_options* indicates unprotected interest is being requested. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
391 ATR_MULTIPLE_INTEREST_ OPTION_INV	**Meaning**: Program error. The multiple interest option specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
397 ATR_XID_DATA_INV	**Meaning**: Program error. The computed length of the XID does not match the specified length passed via the *xid_length* parameter. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
399 ATR_UR_FAMILY_ OPTION_INV	**Meaning**: Program error. The UR family option specified in the call is not valid. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
39A ATR_PARENT_UR_TOKEN_INV	**Meaning**: Program or environmental error. The UR token specified in the *parent_UR_token* parameter is not valid because one of the following is true: • It was coded incorrectly • The parent transaction failed • The system it resided on failed • The coordinator system failed • The RRS running on that system failed • The parent UR belongs to a system that is not in the same RRS logging group as this system If any of these conditions occurs, the system rejects the service call. **Action**: Check the calling program for a probable coding error. • If it's a program error, correct the program and rerun it. • If the work manager was creating a multisystem cascaded UR, the work manager must communicate the failure to the work manager originating the request. **Installation Action**: Ensure that the originating work manager and this work manager are in the same RRS logging group.
39C ATR_XID_LENGTH_INV	**Meaning**: Program error. The XID length specified in the call is not valid. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
39D ATR_XID_INV	**Meaning**: Program error. The XID specified in the call is not valid. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
39E ATR_PARENT_DU_ TERMINATING	**Meaning**: Environmental error. The task associated with the UR family specified by the *parent_UR_token* parameter is ending. The system rejects the service call. **Action**: None.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
3A0 ATR_SAME_CURRENT_CONTEXT_INV	**Meaning**: Program error. The UR represented by the *parent_UR_token* and the UR associated with the context represented by the *child_context_token* are both 0. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A1 ATR_SAME_PARENT_ CONTEXT_INV	**Meaning**: Program error. The UR represented by the *parent_UR_token*, which may have been specified with a 0, and the UR associated with the context represented by the *child_context_token* are the same UR. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A2 ATR_SAME_CHILD_CONTEXT_INV	**Meaning**: Program error. The UR represented by the *parent_UR_token* and the UR associated with the context represented by the *child_context_token*, specified with a 0, are the same UR. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A3 ATR_UR_TOKEN_INV	**Meaning**: Program error. The UR token specified in the call does not identify a valid UR. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
3A4 ATR_PARENT_AUTH_FAILURE	**Meaning**: Program error. The caller is PKM8-15 problem state, but specified a *parent_UR_token* of a UR associated with a context which: • Does not belong to a PKM 8-15 problem state resource manager registered in the home address space. • Is not a native context in the home address space. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
3A5 ATR_CHILD_AUTH_FAILURE	**Meaning**: Program error. The caller is PKM 8-15 problem state, but specified a *child_context_token* of a context which: • Does not belong to a PKM 8-15 problem state resource manager registered in the home address space. • Is not a native context in the home address space. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3AC ATR_INTEREST_OPTIONS_INV	**Meaning**: Program error. The *interest_options* value specified on the call is not valid. Either reserved bits were nonzero or an unacceptable selection of options and parameters was specified. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3B0 ATR_XID_EXISTS	**Meaning**: Program error. The resource manager specified an XID, but the UR already has an XID. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3B1 ATR_SUBORDINATE_FAILED_ EXIT_NOT_DEFINED	**Meaning**: Program error. The resource manager requested to be notified, through the SUBORDINATE_FAILED exit, in the event of either RRS, RM, or system failure on a subordinate system. However, a SUBORDINATE_FAILED exit was not provided by the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the program and rerun it.
3B2 ATR_DRV_SUBORDINATE_FAILED_ EXIT_INV	**Meaning**: Program error. The resource manager requested to be notified, through the SUBORDINATE_FAILED exit, in the event of subordinate failure, but the UR in which it is expressing interest is a cascaded UR. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
3B3 ATR_COMMIT_TIER_ONE_SRB_INV	**Meaning**: Program error. The resource manager specified a tier one request for an SRB Commit Exit routine. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the program and rerun it.
3B7 ATR_COMMIT_TIER_ONE_MISMATCH	**Meaning**: Program error. The resource manager expressed interest conditionally and an expression of interest already exists. The tier level specified by the RM does not match the tier level already set in that interest. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the program and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the resource manager token specified in the call is not in a valid state to issue the service call. The resource manager must be in run state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: RRS/MVS has unset the RRS/MVS exit routines for this resource manager. **Action**: The system rejects this service request. The resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. Either the UR state or the transaction mode is not valid for the service call. You cannot call Express_UR_Interest when: • Your resource manager is trying to set an XID, but the *xid_length* parameter specified is 0 • The UR state is any other state but **in-reset** when either *UR_family_option* or *interest_options* indicates a new cascaded UR was to be created or the resource manager attempted to set an XID • The UR state is beyond **in-flight**. A new expression of interest cannot be created if the UR state is beyond **in-flight**. • The UR is in local transaction mode, or this expression of interest would change the UR transaction mode to local, but the resource manager has not called Set_Exit_Information to inform RRS that it supports local transaction mode. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
743 ATR_PARENT_UR_ STATE_ERROR	**Meaning**: Program error. The UR specified by the *parent_UR_token* is not in a valid state for the service call. The UR must be in an **in-reset** or **in-flight** state. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
744 ATR_CHILD_UR_STATE_ERROR	**Meaning**: Program error. The UR associated with the specified *child_context_token* is not in a valid state for the service call. The UR must be in an **in-reset** state. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
749 ATR_MAX_UR_LOG_DATA_ EXCEEDED	**Meaning**: Environmental error. This request will exceed the maximum amount of data that RRS can log for a UR. The system rejects the service call. **Action**: Fail the client program request or back out the UR. Verify that the space set up for logging is adequate.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
763 ATR_PARENT_LOCAL_TRAN _MODE_INV	**Meaning**: Program error. The parent UR is in local transaction mode. This service is valid only for a parent UR in global transaction mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
769 ATR_XID_NO_GLOBAL_MATCH	**Meaning**: Program error. The XID specified in the XID parameter does not have the same FormatID, GTRID_length and GTRID values as the parent UR's XID. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS had been available to the resource manager but has gone down and come back up again. A commit or backout operation may or may not have been in progress for the context under which the Express_UR_Interest was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to express protected interest with presume abort logging in a UR associated with the current context.

```
⋮
RM_TOKEN = MY_RM_TOKEN
C_TOKEN = 0
NON_P_DATA = ANCHOR1
P_DATA_LEN = LENGTH(MY_P_DATA)
P_DATA = MY_P_DATA
INT_OPTS = ATR_PROTECTED_INT_MASK + ATR_PRESUME_ABORT_MASK
PARENT = JUNK
XID_LEN = 0
XID = JUNK
CALL ATREINT5(RC,RM_TOKEN,C_TOKEN,URI_TOKEN,UR_TOKEN,CUR_C_TOKEN,
            URID,INT_OPTS,NON_P_DATA,C_NON_P_DATA,P_DATA_LEN,
            P_DATA,XID_LEN,XID,CASCADE_OPT,PARENT,TRAN_MODE)
IF RC = ATR_OK THEN
   MY_C_TOKEN = CUR_C_TOKEN
   MY_URTOKEN = URI_TOKEN
   MY_URID = URID
   MY_URTOKEN = UR_TOKEN
   MY_TRAN_MODE = TRAN_MODE
⋮
```

Forget_Agent_UR_Interest (ATRAFGT, ATR4AFGT)

- ATRAFGT is for AMODE(31) callers.
- ATR4AFGT is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager that has taken the server distributed syncpoint resource manager (SDSRM) role calls Forget_Agent_UR_Interest to tell RRS to delete the SDSRM's interest in the specified unit of recovery (UR) and, depending on the *log_option* value, delete any log entries that exist. The *log_option* can be set in several ways, such as through the Commit_Agent_UR service or the Backout_Agent_UR service, or implicitly by RRS. The SDSRM should issue Forget_Agent_UR_Interest only when the *log_option* is ATR_DEFER_EXPLICIT.

If the call to Forget_Agent_UR_Interest specifies ATR_IMMEDIATE, then, if there is a log record, RRS deletes the SDSRM's interest and logs the UR without the SDSRM's interest to ensure that, during any subsequent restart processing, it never returns the UR to the SDSRM.

If the UR state is **in-forget**, there are no longer any incomplete interests in the UR; Forget_Agent_UR_Interest deletes the UR. Otherwise, Forget_Agent_UR_Interest

changes the UR state from **in-forget** to **forgotten**. The input *ur_interest_token* is no longer valid, and the SDSRM no longer has any processing obligation related to this unit of recovery. If *log_option* is not ATR_DEFER_EXPLICIT, however, the UR interest might be returned on restart.

RRS serializes Forget_Agent_UR_Interest processing so that resource manager interests are not deleted while resource manager exit routines are running.

If a resource manager with an interest in a UR has taken the SDSRM role, RRS will implicitly change the *log_option* to ATR_DEFER_EXPLICIT under any of the following conditions:

- When the application backs out the UR through a call to the Backout_UR service or the Application_Backout_UR service.
- When an RRS panel or the ATRSRV macro is used to resolve an **in-doubt** UR.
- When RRS re-creates a committed or backed out UR during restart processing.

RRS changes the *log_option* to ensure that the resource manager that has taken the SDSRM role is always informed of the results of the UR and allows the resource manager to safely prevote its BACKOUT and COMMIT exits.

Environment

Authorization:	Supervisor state, or PKM allowing keys 0-7
Dispatchable unit mode:	Task mode
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRAFGT) 64 bit (ATR4AFGT)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	All parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

To use the service:

- The resource manager state must be **run**.
- The unit of recovery state must be **in-commit**, **in-backout**, **in-end**, **in-completion**, or **in-forget**.

If you are coding an RRS exit routine, do not call this service to process the UR passed to the exit routine in the *ur_interest_token* exit routine parameter.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

CALL ATRAFGT	(return_code ,ur_interest_token, ,log_option);

CALL ATR4AFGT	(return_code ,ur_interest_token, ,log_option);

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code for the Forget_Agent_UR_Interest service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that uniquely represents an instance of the resource manager's interest in the particular UR. The resource manager received the token from: Express_UR_Interest, Retrieve_UR_Interest, or Retain_Interest.

,log_option
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates how RRS is to process log entries for the unit of recovery. Specify one of the following values:

Constant in Hexadecimal (Decimal) Equate Symbol	Description
X'0' (0) ATR_DEFER	**Meaning:** RRS is to logically delete the log record when the unit of recovery state changes to **Forgotten**.
X'2' (2) ATR_IMMEDIATE	**Meaning:** Before returning control to the SDSRM, RRS must delete the SDSRM's interest from the UR. RRS hardens a new log record without the interest.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'001C0000' or X'001C0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and **return_code** contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: The operation completed successfully. **Action**: Continue normal processing.
10 ATR_OK_NO_CONTEXT	**Meaning**: The operation completed successfully. The UR state was **in-completion** or **in-forget**, and there was no associated context. **Action**: Continue normal processing.
11 ATR_FORGET_NOT_REQUIRED	**Meaning**: The specified *ur_interest_token* represents a UR that does not require the Forget_Agent_UR service. **Action**: Continue normal processing.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: The caller is holding one or more locks. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: The system level does not support this service. The system rejects this service request. **Action**: Remove the calling program from the system, and install it on a system that supports RRS. Then rerun the calling program.
370 ATR_URI_TOKEN_INV	**Meaning**: The specified *UR_interest_token* does not represent a valid expression of interest. This condition can occur after RRS has terminated and restarted. The system rejects this service request. **Action**: Check the calling program for a probable coding error.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
395 ATR_LOG_OPT_INV	**Meaning**: The specified *log_option* value is not valid. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
701 ATR_RM_STATE_ERROR	**Meaning**: The resource manager state is not valid for this request. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
702 ATR_RM_EXITS_UNSET	**Meaning**: RRS has unset the RRS exit routines for this resource manager. The system rejects this service request. **Action**: The resource manager must reset its RRS exit routine information and begin restart processing with RRS.
731 ATR_UR_STATE_ERROR	**Meaning**: The UR state is not valid for this request. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
74A ATR_NOT_SERVER_DSRM	**Meaning**: The resource manager does not have the server distributed syncpoint resource manager role for the unit of recovery. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
750 ATR_RESPOND_CONTINUE_REQUIRED	**Meaning**: The resource manager must call Respond_to_Retrieved_Interest before it can call Forget_Agent_UR for this interest. **Action**: The system rejects this service request. Call Respond_to_Retrieved_Interest, then call Forget_Agent_UR for this interest.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F04 ATR_UNEXPECTED_UR_ERROR	**Meaning**: System error. While processing the UR, RRS has encountered an unexpected error that might have damaged the UR. The system rejects the service call. **Action**: Contact the system programmer who maintains RRS at your installation. Manual intervention might be needed to restore consistent resources.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: This service routine encountered an unexpected error. The system rejects this service request. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager tells RRS to forget the unit of recovery. Storage for the call parameters has been allocated.

```
⋮
URI_TOKEN = MY_URI_TOKEN
FTOPT=ATR_DEFER
CALL ATRAFGT(RC,URI_TOKEN,FTOPT)
⋮
```

Post_Deferred_UR_Exit (ATRPDUE, ATR4PDUE)

- ATRPDUE is for AMODE(31) callers.
- ATR4PDUE is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

Several resource manager exit routines allow the resource manager to initiate asynchronous processing, then return to RRS with a return code that indicates a deferred response. The return codes that indicate a deferred response are:

```
ATRX_LATER
ATRX_LATER_CONTINUE
```

When the asynchronous processing completes, the resource manager calls the Post_Deferred_UR_Exit service to pass to RRS a return code that reflects the results of the asynchronous processing.

In response to the call, RRS issues a return code.

Environment

The requirements for the caller are:

Minimum authorization: PKM allowing key 0-7, or supervisor state
Dispatchable unit mode: Task or SRB
Cross memory mode: Any PASN, any HASN, any SASN

AMODE:	
	31 bit (ATRPDUE)
	64 bit (ATR4PDUE)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

For the call, the UR must be in the same state as it was when the exit returned the deferred response code. The UR state cannot be **in-reset**, **in-flight**, or **in-forget**.

The exit routine's resource manager state must be either:

- **Restart**, which means it has registered, set its exit routines with RRS, begun restart, and requested incomplete UR interests
- **Run**, which means it has registered, set its exit routines with RRS, and completed restart

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents

0-1 Used as work registers by the system

2-13 Unchanged

14-15 Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRPDUE	(return_code ,ur_interest_token ,exit_number ,completion_code)

CALL ATR4PDUE	(return_code ,ur_interest_token ,exit_number ,completion_code)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Post_Deferred_UR_Exit service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that uniquely represents an instance of the resource manager's interest in the particular UR. Your resource manager received the token on a call to: Express_UR_Interest, Retrieve_UR_Interest, or Retain_Interest.

,exit_number
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies the exit number for the exit routine that has completed with the code in the *completion_code* parameter. "Set_Exit_Information (CRGSEIF, CRGSEIF1,CRG4SEIF)" on page 149 lists the exit routines and their numbers, which are assigned by RRS.

,completion_code
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies the return code from the asynchronous exit routine that has now completed processing. The code can be any valid return code for the exit routine, except `ATRX_LATER` or `ATRX_LATER_CONTINUE`.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00090000' or X'00090001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
10 ATR_OK_NO_CONTEXT	**Meaning**: The operation completed successfully. The UR state was **in-completion** or **in-forget**, and there was no associated context. **Action**: Continue normal processing.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
109 ATR_ENVIRONMENT_INV	**Meaning**: Program error. The resource manager invoked the service from an SRB suspend exit or from an SRB that was ended abnormally by the PURGEDQ service. The system rejects the call to Post_Deferred_UR_Exit. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call is not for one of the currently valid interests. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
378 ATR_EXIT_NUMBER_INV	**Meaning**: Program error. The exit number specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
379 ATR_COMP_CODE_INV	**Meaning**: Program error. The exit return code specified in the *completion_code* parameter in the call is not valid for the exit routine. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Check both the completion code and the exit number in the call. Correct the resource manager and rerun it.
381 ATR_LATER_INV	**Meaning**: Program error. The *completion_code* parameter specified in the call is `ATRX_LATER` or `ATRX_LATER_CONTINUE`. This value cannot be used for this call. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager must be in restart or run state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: The resource manager must reset its RRS exits and begin restart processing with RRS.
740 ATR_POST_NOT_PENDING	**Meaning**: Program error. For the exit number and the interest specified in the call, RRS is not expecting a return code from the exit routine. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F03 ATR_UR_RESOLVED_BY_ INSTALLATION	**Meaning**: Environmental error. The resource manager called the Post_Deferred_UR_Exit service to resolve a UR in an **in-doubt** state. However, the installation had already used panel input to resolve the UR. The system rejects the service call. **Action**: RRS processes the UR based on the installation input.
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Post_Deferred_UR_Exit was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the exit routine issues a call to supply its return code.

```
⋮
EXIT_NUM = ATR_PREPARE_EXIT
URI_TOKEN = MY_URI_TOKEN
CCODE = ATRX_OK
CALL ATRPDUE(RC,URI_TOKEN,EXIT_NUM,CCODE)
```

```
IF RC ≠ ATR_OK THEN
    /* Handle error */
⋮
```

Prepare_Agent_UR (ATRAPRP, ATR4APRP)

- ATRAPRP is for AMODE(31) callers.
- ATR4APRP is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager that has taken the server distributed syncpoint resource manager (SDSRM) role calls Prepare_Agent_UR to tell RRS to initiate the prepare phase of a syncpoint operation for the unit of recovery (UR) associated with the specified UR interest. When the SDSRM calls this service, RRS collects the local prepare votes and does one of the following:

1. If the result of the state check indicates that a state check condition exists, RRS makes no changes to the state of the UR.
2. If the collective prepare vote result was OK, RRS sets the UR state to **in-doubt**.
3. If the collective prepare vote result was forget, RRS sets the UR state to **forgotten**.
4. If the collective prepare vote result was backout, RRS backs out the unit of recovery and sets its state to **forgotten**.
5. If the application has already caused backout of the UR, RRS might have already backed out the UR. In this case, RRS returns ATR_UR_STATE_ERROR. The UR might be in any state, but, once it reaches **in-forget**, it remains in that state until the resource manager calls Forget_Agent_UR.

A successful call to Prepare_Agent_UR changes the UR state to **in-doubt** or **forgotten**.

If a resource manager with an interest in a UR has taken the SDSRM role, RRS will implicitly change the *log_option* to ATR_DEFER_EXPLICIT (see "Commit_Agent_UR (ATRACMT, ATR4ACMT)" on page 256) under any of the following conditions:

- When the application backs out the UR through a call to the Backout_UR service or the Application_Backout_UR service.
- When RRS re-creates a UR during restart processing.

If any of these conditions has occurred, RRS returns the ATR_UR_STATE_ERROR return code. The UR might be in any state, but, once it reaches **in-forget**, it will remain in that state until the Forget_Agent_UR service is called. RRS waits for Forget_Agent_UR to ensure that the resource manager that has taken the SDSRM role is always informed of the results of the UR. This allows the resource manager to safely pre-vote its BACKOUT and COMMIT exits.

Note: The SDSRM may issue Forget_Agent_UR without waiting for the UR to reach the **in-forget** state.

Environment

Authorization: Supervisor state, or PKM allowing keys 0-7
Dispatchable unit mode: Task mode
Cross memory mode: Any PASN, any HASN, any SASN

AMODE:	
	31 bit (ATRAPRP)
	64 bit (ATR4APRP)
ASC mode:	Primary or AR
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	All parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

To use the service:

- The resource manager state must be **run**.
- The unit of recovery state must be **in_flight**.

Attention: Do not invoke this service to prepare a UR with a work context associated with a task in the resource manager's address space. If you do, it might be impossible for the resource manager address space to end or for the resource manager to restart without a complete system restart.

CAUTION:
The resource manager must ensure that no application can be updating protected resources for the unit of recovery being prepared. This is necessary to ensure that no resource manager taking part in the unit of recovery sees updates being made on behalf of a unit of recovery at the same time as they are executing syncpoint processing.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged

14 Used as a work register by the system

15 Return code

When control returns to the caller, the ARs contain:

Register
Contents

0-1 Used as work registers by the system

2-13 Unchanged

14-15 Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

CALL ATRAPRP	(return_code ,ur_interest_token ,log_option);

CALL ATR4APRP	(return_code ,ur_interest_token ,log_option);

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code for the Prepare_Agent_UR service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction

- Length: 16 bytes

Specifies the UR interest token that uniquely represents an instance of the resource manager's interest in the particular UR. The resource manager received the token from a call to the Express_UR_Interest service or the Retain_Interest service.

,log_option

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates how RRS is to process log entries for the unit of recovery. This option affects processing only when the service receives a return code of ATR_FORGET. Specify one of the following:

Constant in Hexadecimal (Decimal) Equate Symbol	Description
X'0' (0) ATR_DEFER_IMPLICIT	**Meaning:** RRS is to logically delete the log record when the unit of recovery state changes to **Forgotten**. Your resource manager will not call the Forget_Agent_UR service.

A later call to the Commit_Agent_UR service or the Backout_Agent_UR service can override the log option specified here.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'001D0000' or X'001D0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and **return_code** contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: The prepare operation completed successfully. The collective prepare vote allows the unit of recovery to be committed and all resource managers did not vote to abstain or forget. The UR state is now **In_Doubt**. **Action**: Continue normal processing by determining the resolution of the **In_Doubt** condition and reporting it to RRS with Commit_Agent_UR or Backout_Agent_UR.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
8 ATR_FORGET	**Meaning**: The prepare operation completed successfully. The collective prepare vote allows the unit of recovery to be completed, and all resource managers voted to abstain or forget. The UR state is now **Forgotten**. **Action**: Continue normal processing.
C8 ATR_PROGRAM_STATE_CHECK	**Meaning**: The commit operation failed. The consistency state of the protected resources has not been altered. This return code indicates one of the following conditions has occurred: • A protected resource, specifically a communications Interface conversation, is not in **Send, Send Pending, Defer Receive, Defer Allocate, Sync_Point, Sync_Point Send,** or **Sync_Point Deallocate** state. • A protected resource, specifically a Communications Interface conversation, is in **Send** state, and the program started but did not finish sending a basic conversation logical record. • A protected resource, specifically a local resource, is not in the proper state for a commit. **Action**: If possible, initiate a resource manager action to get the resources to a committable state and then invoke the Prepare_Agent_UR service again. Otherwise, issue Backout_Agent_UR to back out the transaction.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: The caller is disabled. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, which is the required mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: The caller is holding one or more locks. The system rejects this service request. **Action**: Check the calling program for a probable coding error.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: The system level does not support this service. The system rejects this service request. **Action**: Remove the calling program from the system, and install it on a system that supports RRS. Then rerun the calling program.
12C ATR_BACKED_OUT	**Meaning**: The commit operation failed. All protected resources have been returned to the previous consistent state. **Action**: Continue normal processing for a backed out unit of recovery. The UR state is now **Forgotten**.
12D ATR_BACKED_OUT_OUTCOME_ PENDING	**Meaning**: The commit operation failed. The RRS decision was to return to the previous consistent state. However, the state of one or more of the protected resources is not known. **Action**: Continue normal processing for a backed out unit of recovery. The UR state is now **Forgotten**.
12E ATR_BACKED_OUT_OUTCOME_ MIXED	**Meaning**: The commit operation failed. However, one or more of the protected resources has advanced to a new synchronization state. **Action**: Report the error to the other transactional participants. The UR state is now **Forgotten**.
370 ATR_URI_TOKEN_INV	**Meaning**: The specified *UR_interest_token* does not represent a valid expression of interest. This condition can occur after RRS has terminated and restarted. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
395 ATR_LOG_OPT_INV	**Meaning**: The specified *log_option* value is not valid. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
701 ATR_RM_STATE_ERROR	**Meaning**: The resource manager state is not valid for this request. The system rejects this service request. **Action**: Check the calling program for a probable coding error.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
702 ATR_RM_EXITS_UNSET	**Meaning**: RRS has unset the RRS exit routines for this resource manager. The system rejects this service request. **Action**: The resource manager must reset its RRS exit routine information and begin restart processing with RRS.
731 ATR_UR_STATE_ERROR	**Meaning**: The UR state is not valid for this service request. The system rejects the request. The application might have already requested backout. Call Retrieve_UR_Data or Retrieve_Side_Information to obtain information about the state of the UR. If you receive this return code, you must call Forget_Agent_UR to complete processing for the UR. **Action**: Call Forget_Agent_UR to complete the processing of this UR.
74A ATR_NOT_SERVER_DSRM	**Meaning**: The resource manager does not have the server distributed syncpoint resource manager role for the unit of recovery. The system rejects this service request. **Action**: Check the calling program for a probable coding error.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F04 ATR_UNEXPECTED_UR_ERROR	**Meaning**: System error. While processing the UR, RRS has encountered an unexpected error that might have damaged the UR. The system rejects the service call. **Action**: Contact the system programmer who maintains RRS at your installation. Manual intervention might be needed to restore consistent resources.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: This service routine encountered an unexpected error. The system rejects this service request. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager wants to initiate the prepare phase of syncpoint for the unit of recovery. Storage for the call parameters has been allocated.

```
⋮
URI_TOKEN = MY_URI_TOKEN
FTOPT=ATR_DEFER_IMPLICIT
CALL ATRAPRP(RC,URI_TOKEN,FTOPT)
⋮
```

Respond_to_Retrieved_Interest (ATRIRRI, ATR4IRRI)

- ATRIRRI is for AMODE(31) callers.
- ATR4IRRI is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

The resource manager calls the Respond_to_Retrieved_Interest service to tell RRS how to process an interest in an incomplete unit of recovery (UR). During restart, your resource manager must retrieve its interests in repetitive calls to the Retrieve_UR_Interest service; every call that retrieves an interest must be followed by a Respond_to_Retrieved_Interest call, which tells RRS that:

- RRS should continue processing the UR by invoking the resource manager's exit routines.
- The resource manager has completed work for this interest in the UR. RRS should delete this interest.

In response to the call, RRS returns a return code.

RRS Actions for Incomplete URs: If the *response_code* you specify on the Respond_to_Retrieved_Interest call is ATR_RESPOND_CONTINUE, RRS processing is summarized in Table 43. The usual resource manager role is participant, but the Set_Syncpoint_Controls call can specify a different role. The action RRS takes for each incomplete UR depends on the UR state and the resource manager role.

Table 43. Actions for Incomplete URs

UR state	Resource manager role	Response code	RRS action for continue
In-doubt	Participant	Continue	Invokes the COMMIT or BACKOUT exit routine after the UR state is resolved
In-doubt	Distributed syncpoint resource manager	Continue	Invokes the DISTRIBUTED_SYNCPOINT exit routine to resolve the **in-doubt** UR state
In-doubt	Server distributed syncpoint resource manager	Continue	Prepares for eventual Commit_Agent_UR or Backout_Agent_UR.
In-commit	Any	Continue or complete	For continue, RRS invokes the COMMIT exit routine. For complete, it does not.
In-backout	Any	Continue or complete	For continue, RRS invokes the BACKOUT exit routine. For complete, it does not.

Table 43. Actions for Incomplete URs (continued)

UR state	Resource manager role	Response code	RRS action for continue
Note: If a resource manager calls Respond_to_Retrieved_Interest in **Restart** state and specifies ATR_RESPOND_CONTINUE, RRS does not invoke any exits for any of the resource manager's interests until the resource manager calls the End_Restart service.			

The installation cannot resolve the **in-doubt** state of a UR through RRS ISPF panels between the time when the resource manager sets its RRS exit routines and the time when a resource manager responsible for resolving the UR specifies ATR_RESPOND_CONTINUE. It is thus a good idea to design the resource manager so that this time is as short as possible.

Complete URs: Your resource manager should call the Respond_to_Retrieved_Interest service with a *response_code* of ATR_RESPOND_COMPLETE when a Retrieve_UR_Interest call returns a UR that your resource manager has completed,

Nonpersistent Interest Data: The Respond_to_Retrieved_Interest call can also provide nonpersistent interest data. RRS ignores nonpersistent interest data if the call specifies ATR_RESPOND_COMPLETE. Otherwise, RRS gives this data to each of the resource manager's exit routines that it invokes for this interest. This data is not recorded in nonvolatile storage and is not available at subsequent restarts.

Your resource manager can retrieve nonpersistent interest data in a call to the Retrieve_UR_Interest_Data service.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRIRRI) 64 bit (ATR4IRRI)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The resource manager associated with the UR interest token specified in the call must be in either **Restart** state or **Run state**.

The state of the specified UR must be **in-doubt**, **in-commit**, or **in-backout**.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRIRRI	(return_code ,ur_interest_token ,response_code ,nonpersistent_interest_data)

CALL ATR4IRRI	(return_code ,ur_interest_token ,response_code ,nonpersistent_interest_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Respond_to_Retrieved_Interest service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Is the UR interest token that identifies your resource manager's interest in the incomplete UR. Your resource manager received the token from the Retrieve_UR_Interest callable service.

,response_code
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Indicates how RRS is to respond to the UR interest. Specify one of the following:

Response Code in: Hexadecimal (Decimal) Equate Symbol	**Response**
X'0' (0) ATR_RESPOND_CONTINUE	RRS should continue processing the UR by invoking the resource manager's exit routines. If the resource manager is in **Restart** state, RRS does not invoke the exit routines until the resource manager has called the End_Restart service.

Response Code in: Hexadecimal (Decimal) Equate Symbol	Response
X'1' (1) ATR_RESPOND_COMPLETE	The resource manager has completed work for this interest in the UR. RRS should delete this interest. You cannot choose this response code for a UR that is **in-doubt**.

,nonpersistent_interest_data

Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the nonpersistent interest data for your resource manager's interest. RRS provides this data to each exit routine it invokes for the UR but does not record the data in nonvolatile storage. If you specified a *response_code* of ATR_RESPOND_COMPLETE, RRS ignores the data.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00070000' or X'00070001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and Action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and Action
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call is not one of the currently valid interests. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
384 ATR_RESPONSE_CODE_INV	**Meaning**: Program error. The *response_code* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
385 ATR_RESPONSE_CODE_ INCORRECT	**Meaning**: Program error. The *response_code* value specified in the call is not correct for the state of the UR or the role of the resource manager or both. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager must be in restart or run state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and Action
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: The resource manager must reset its RRS exits and begin restart processing with RRS.
741 ATR_NOT_RETRIEVED_INTEREST	**Meaning**: Program error. The UR interest token specified in the call is not for a retrieved interest. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
742 ATR_RESPONSE_NOT_PENDING	**Meaning**: Program error. RRS is not expecting a *process interest* call for the UR interest token specified in the call. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and Action
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Respond_to_Retrieved_Interest was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to respond to a retrieved interest. The call requests that RRS invoke the resource manager's exit routines.

```
⋮
URI_TOKEN = UR_INTEREST_TOKEN
NON_P_DATA = ANCHOR1
RESPCOD = ATR_CONTINUE
CALL ATRIRRI(RC,URI_TOKEN,RESPCOD,NON_P_DATA)
IF RC ≠ ATR_OK THEN
     /* Handle error */
⋮
```

Retain_Interest (ATRSROI, ATRSROI1, ATR4SROI)

- ATRSROI is for AMODE(31) callers.

- ATRSROI1 is for AMODE(31) callers and allows a resource manager to request to have its COMMIT exit run at a tier one priority with regards to other resource managers in the next UR.
- ATR4SROI is for AMODE(64) callers and allows parameters in 64 bit addressable storage and allows a resource manager to request to have its COMMIT exit run at a tier one priority with regards to other resource managers in the next UR.

A resource manager calls the Retain_Interest service to express interest in the next unit of recovery (UR) for the current context when the current UR completes.

While managing conversations for a work request, a communications resource manager can use the Retain_Interest call to make sure it is aware of all URs for the work request.

The new UR that the Retain_Interest service creates begins after the current UR reaches **in_completion** state.

In response to the call, RRS returns:

- A return code.
- A new UR interest token for the interest in the next UR. You need this token for many calls to RRS services.
- A UR identifier (URID) for the next UR, for both protected and unprotected expressions of interest.

Once the current UR reaches **in_completion** state, you can obtain the LUWID for the UR created by Retain_Interest. To obtain the LUWID, call the Set_Work_Identifier service:

- Specify the *new_ur_interest_token* that Retain_Interest provides.
- Request the current LUWID.

Protected and unprotected interests: The call can express a protected or unprotected interest in the next UR. For a protected interest, RRS or a resource manager coordinates changes to the resources, so that all changes are made or no changes are made. Resources that can be protected are a database, a conversation between two communications managers, or a product-specific resource.

Action for Resource Manager Failure: In the call, you can specify how RRS should process requests to commit the next UR if the resource manager becomes:

- **Unregistered**: Your resource manager is no longer registered as a resource manager. See the Register_Resource_Manager callable service for a description of the ways in which your resource manager can become unregistered.
- **Unset**: Your resource manager's exit routines are no longer set with RRS.

RRS can process requests as follows:

- **Standard processing**: RRS should back out the next UR, if the state of the UR is **in-reset**, **in-flight**, **in-state-check**, or **in-prepare**.
- **Forget interest**: RRS is to delete the resource manager's interest in the next UR. You may specify this value only if the *interest_type* is ATR_UNPROTECTED.

Persistent interest data: In the Retain_Interest call, your resource manager can provide persistent interest data if the interest is protected. When hardening information for the interest in an RRS log, RRS records the persistent interest data.

Because the data is hardened, it will be available if your resource manager restarts or if RRS restarts, forcing your resource manager to restart.

In addition to using Retain_Interest, your resource manager can also provide persistent interest data in a call to any of the following services: Express_UR_Interest, Change_Interest_Type, or Set_Persistent_Interest_Data. Your resource manager can retrieve the data in a call to the Retrieve_UR_Interest service or the Retrieve_UR_Data service.

Nonpersistent interest data: The call can also provide nonpersistent interest data. RRS gives this data to each resource manager exit routine it invokes for this interest. This data is not recorded in nonvolatile storage and is not available at subsequent restarts.

Your resource manager can also provide nonpersistent interest data for an interest in a call to the Express_UR_Interest service or the Respond_to_Retrieved_Interest service. Your resource manager can retrieve it in a call to the Retrieve_Interest_Data service.

URID: Your resource manager should save the returned URID with the data for the next UR in its resource manager log. During restart processing after the resource manager, RRS, or system fails, your resource manager obtains the URID for an incomplete UR from the Retrieve_UR_Interest service. Compare the URID from the Retrieve_UR_Interest service with URIDs in the resource manager's log to find the data for the incomplete UR.

Your resource manager can also obtain the URID from a call to the following services: Express_UR_Interest, Retrieve_UR_Interest, Retrieve_UR_Data, or Change_Interest_Type.

Commit exit tier priority: On this call, you can specify the tier priority at which RRS should invoke your COMMIT exit:

- **Tier one priority:** RRS will invoke the resource manager's COMMIT exit before other resource managers. If multiple resource managers request the tier one priority, the commits exits will be driven in the order in which they expressed interest in the UR.
- **No priority:** The resource manager's exit will be driven after tier one resource managers' commit exits, if any.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRSROI, ATRSROI1) 64 bit (ATR4SROI)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

There are restrictions related to the current UR:

- The UR state must not be **in_reset**, **in_flight**, **in_completion**, or **in_forget**.
- The current UR must not be in local transaction mode.
- The expression of interest must not be a restart expression of interest (that is, an interest returned by the Retrieve_UR_Interest service).
- No resource manager can have the server distributed syncpoint resource manager role in the UR.

The state of the resource manager associated with the specified UR interest token must be **run**, which means it has registered, set its exit routines with RRS, and completed restart.

Interest cannot be retained in a cascaded UR, a UR with an SDSRM, or a UR whose context is terminating.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRSROI	(return_code ,ur_interest_token ,new_ur_interest_token ,ur_identifier ,interest_type ,failure_action ,nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data)

CALL ATRSROI1	(return_code ,ur_interest_token ,new_ur_interest_token ,ur_identifier ,interest_options ,nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data)

CALL ATR4SROI	(return_code ,ur_interest_token ,new_ur_interest_token ,ur_identifier ,interest_options ,nonpersistent_interest_data ,persistent_interest_data_length ,persistent_interest_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Retain_Interest service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that identifies your resource manager's interest in the current UR. Your resource manager received the token from the Express_UR_Interest service.

Note that the token cannot be for a UR interest returned by a Retrieve_UR_Interest call during restart.

,new_ur_interest_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives from the service the UR interest token that identifies your resource manager's interest in the next UR.

,ur_identifier
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives from the service the UR identifier (URID) that uniquely identifies the next UR.

,interest_options
Supplied parameter

- Type: Bit string
- Character Set: N/A
- Length: 4bytes

Specifies various options that determine how RRS will process this interest. Each of the bits in *interest_options* is either reserved or has a specific meaning. Each reserved bit must be specified as zero. Each other bit can be specified as either zero or one. The bit specifications are:

Bit positions	Constant in: Hexadecimal Equate Symbol	Description
7		**Interest Type**
	00000000 ATR_UNPROT_INT_MASK	A resource manager specifies zero to express an unprotected interest in the UR.
		A resource manager specifies one to express a protected interest in the UR.

Bit positions	Constant in: Hexadecimal Equate Symbol	Description
11		**Failure action**
	00000000 ATR_STANDARD_FAIL_MASK	A resource manager specifies zero when it wants RRS to do its standard processing if the resource manager fails.
	00100000 ATR_REMOVE_INT_ON_FAIL_MASK	A resource manager specifies one when it wants RRS to remove its interest in the UR if the resource manager fails. **Note:** One can only be specified if the resource manager is expressing an unprotected interest in the next UR.
14		**Commit exit tier priority**
	00000000 ATR_COMMIT_NO_PRIORITY	When zero is specified, the resource manager does not require RRS to drive its COMMIT exit at a higher priority with regards to other resource managers in the same UR.
	00020000 ATR_COMMIT_TIER_ONE_ PRIORITY	When one is specified, the resource manager wants RRS to drive its COMMIT exit first with respect to other resource managers' exits.

,interest_type

Supplied parameter

- Type: Integer
- Length: 4 bytes

Indicates the type of interest the resource manager has in the next UR. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) ATR_UNPROTECTED	**Unprotected:** The resource manager is expressing an unprotected interest in the UR.
1 (1) ATR_PROTECTED	**Protected:** The resource manager is expressing a protected interest in the UR.

,failure_action

Supplied parameter

- Type: Integer
- Length: 4 bytes

Defines how RRS is to process commit requests for the next UR if the resource manager becomes unregistered or unset. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Action
0 (0) ATR_FAIL_STANDARD	Standard processing
2 (2) ATR_FAIL_FORGET	Forget interest

For the current UR, the failure action after this call is ATR_FAIL_STANDARD.

,nonpersistent_interest_data
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the nonpersistent interest data for your resource manager's interest. RRS does not record this data in nonvolatile storage.

,persistent_interest_data_length
Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies, in hexadecimal, the length of the persistent interest data. Specify X'0' - X'1000' (0-4096) bytes. If the interest type is ATR_UNPROTECTED, then this field must be binary zeros.

,persistent_interest_data
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified in *persistent_interest_data_length*

The persistent interest data for your resource manager's interest in the UR. RRS records this data in an RRS log. If *persistent_interest_data_length* is binary zeros, RRS ignores this parameter.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00110000' or X'00110001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call is not one of the currently valid interests. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
371 ATR_INTEREST_TYPE_INV	**Meaning**: Program error. The *interest_type* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
372 ATR_FAILURE_ACTION_INV	**Meaning**: Program error. The *failure_action* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
376 ATR_PERSISTENT_DATA_LEN_INV	**Meaning**: Program error. The length specified in the *persistent_interest_data_len* parameter in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
386 ATR_FAILURE_ACTION_INCORRECT	**Meaning**: Program error. The failure action specified in the call is not valid for the specified interest type. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
389 ATR_PERSISTENT_DATA_NOT_ ALLOWED	**Meaning**: Program error. The persistent interest data length specified in the call is not zero; zero is the only value valid with the specified interest type of ATR_UNPROTECTED. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
3B3 ATR_COMMIT_TIER_ONE_SRB_INV	**Meaning**: Program error. The resource manager specified a tier one request for an SRB Commit Exit routine. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
3B7 ATR_COMMIT_TIER_ONE_MISMATCH	**Meaning**: Program error. The resource manager expressed interest conditionally and an expression of interest already exists. The tier level specified by the RM does not match the tier level already set in that interest. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager must be in **run** state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: The resource manager must reset its RRS exits and begin restart processing with RRS.
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The current UR is not in a valid state for the service call. The UR state must not be **in_reset**, **in_flight**, **in_completion**, or **in_forget**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
736 ATR_SROI_ALREADY_DONE	**Meaning**: Program error. The resource manager has already successfully called the Retain_Interest service for this UR interest token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
73C ATR_AFTER_NEW_UR	**Meaning**: Program error. The application is already running under a new UR. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
73D ATR_INV_FOR_RESTART_ INTEREST	**Meaning**: Program error. The current UR is a restart UR. The retain interest service cannot be invoked for restart URs. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
747 ATR_TERMINATING_SYNCPOINT	**Meaning**: Program error. RRS is processing a terminating syncpoint so there cannot be any more new URs for this contact. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
748 ATR_RM_IS_THE_SDSRM	**Meaning**: Environmental error. A resource manager has taken the SDSRM role for this UR. Interest cannot be retained in a UR with an SDSRM. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
749 ATR_MAX_UR_LOG_DATA_ EXCEEDED	**Meaning**: Environmental error. This request will exceed the maximum amount of data that RRS can log for a UR. The system rejects the service call. **Action**: Fail the client program request or back out the UR. Verify that the space set up for logging is adequate.
760 ATR_CASCADED_UR	**Meaning**: Program error. The UR is a cascaded UR. Interest cannot be retained in a cascaded UR. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
764 ATR_LOCAL_TRAN_MODE_INV	**Meaning**: Program error. The current UR is in local transaction mode. This service is valid only for a UR in global transaction mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. If the caller is a resource manager, it should not unset its exits with RRS.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Retain_Interest was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to express protected interest in the next UR for the current context.

```
⋮
URI_TOKEN = MY_URI_TOKEN
NON_P_DATA = ANCHOR1
P_DATA_LEN = LENGTH(MY_P_DATA)
P_DATA = MY_P_DATA
INT_TYPE = ATR_PROTECTED
FAIL_ACT = ATR_FAIL_STANDARD
CALL ATRSROI(RC,URI_TOKEN,NEW_URI_TOKEN,URID,INT_TYPE,
            FAIL_ACT,NON_P_DATA,P_DATA_LEN,P_DATA)
IF RC = ATR_OK THEN
```

```
   MY_NEXT_URITOKEN = NEW_URI_TOKEN
⋮
```

Retrieve_Environment (ATRRENV, ATR4RENV)

- ATRRENV is for AMODE(31) callers.
- ATR4RENV is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A work manager calls the Retrieve_Environment service to retrieve the environment settings at the address space level, at the context level, or at the default level. The settings retrieved have either been set via the Set_Environment service, or defaulted to. RRS need not be available when the service is called.

Environment

The requirements for the caller are:

Minimum authorization:	Any
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (ATRRENV) 64 bit (ATR4RENV)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

SRB mode callers cannot specify a context token of 0 when trying to retrieve environment settings at the context scope (ATR_CONTEXT_SCOPE).

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRRENV	(return_code ,diag_area ,scope ,context_token ,stoken ,element_count ,environment_id ,environment_value ,environment_protection)

CALL ATR4RENV	(return_code ,diag_area ,scope ,context_token ,stoken ,element_count ,environment_id ,environment_value ,environment_protection)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Retrieve_Environment service.

,diag_area
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 32 bytes

Contains diagnostic data from Retrieve_Environment to help IBM Service determine the cause of a Retrieve_Environment failure. Be sure to log this data when recording any information about a Retrieve_Environment failure.

,scope
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies the scope at which you want to receive environment setting value(s), either address space level, context level, or at the default level, as follows:

Constant in: Hexadecimal (Decimal) Equate Symbol	**Description**
1 (1) ATR_ADDRESS_SPACE_SCOPE	Retrieve the environmental setting for each element in the *environment_id* array at the address space level.

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
2 (2) ATR_CONTEXT_SCOPE	Retrieve the environmental setting for each element in the *environment_id* array at the context level for the context represented by the *context_token* parameter.
3 (3) ATR_DEFAULT_SCOPE	Retrieve the environmental setting for each element in the *environment_id* array at the level that RRS will use for the UR associated with the work context represented by the *context_token* parameter.

,context_token

Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token of the context for which the resource manager is retrieving context scope environment settings:

- 0: Binary zero specifies either:
  - The current context (when *scope* is ATR_CONTEXT_SCOPE or ATR_DEFAULT_SCOPE)
  - No context (when *scope* is ATR_ADDRESS_SPACE_SCOPE)

  If *scope* is ATR_ADDRESS_SPACE_SCOPE, then *context_token* must be 0.
- token: Specifies a valid context token.

,stoken

Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 8 bytes

Specifies the space token (stoken) of the address space for which the resource manager is retrieving address space scope environment settings:

- 0: Binary zeros indicate the primary address space. If *scope* is ATR_CONTEXT_SCOPE or ATR_DEFAULT_SCOPE, then *stoken* must be 0.
- token: Specifies a valid address space token.

,element_count

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies the number of elements in the environment-retrieving array, which consists of the *environment_id*, *environment_value*, and *environment_protection* parameters.

The maximum number of elements is the number of possible environment settings (transaction mode and two-phase commit action) times the number of environment-retrieving parameters. The maximum number is 2.

,environment_id
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies one or more identifiers; each identifier supplies an attribute of the environment settings to be retrieved. When you specify more than one identifier, you must define an array; *element_count* indicates the number of elements in the array. The positions of the identifiers in this array define the positions of the environment settings to be returned in the *environment_id* array. The *scope* parameter specifies the scope at which these settings are to apply. Specify each identifier as one of the following:

Identifier in: Hexadecimal (Decimal) Equate Symbol	**Description**
1 (1) ATR_TRAN_MODE_SETTING	Retrieve the environmental setting for transaction mode. If *scope* is ATR_DEFAULT_SCOPE and no call was made to Set_Environment to set the transaction mode, RRS returns ATR_HYBRID_GLOBAL_MODE. If *scope* is ATR_ADDRESS_SPACE_SCOPE or ATR_CONTEXT_SCOPE, RRS returns the value of the address space or context as specified on the last applicable call to Set_Environment for the specified address space or context. If the specified environment has not been set, RRS returns ATR_ENVIRONMENT_NOT_SET.
2 (2) ATR_NORM_CTX_END_SETTING	Retrieve the environmental setting for the two-phase commit action RRS is to take for **in-flight** URs when their associated context goes through normal end processing. If *scope* is ATR_DEFAULT_SCOPE and no call was made to Set_Environment to set the normal transaction context end setting, RRS returns ATR_COMMIT_ACTION. If *scope* is ATR_ADDRESS_SPACE_SCOPE or ATR_CONTEXT_SCOPE, RRS returns the value of the address space or context as specified on the last applicable call to Set_Environment for the specified address space or context. If the specified environment has not been set, RRS returns ATR_ENVIRONMENT_NOT_SET.

,environment_value

Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Returns a value for each identifier on the *environment_id* parameter. When you specify more than one identifier, you must define an array, where *element_count* indicates the number of elements in the array. The positions of the identifiers in the *environment_id* array define the positions of the environment attributes in the *environment_value* array.

The value returned for ATR_TRAN_MODE_SETTING is one of the following:

Value in: Hexadecimal (Decimal) Equate Symbol	**Description**
0 (0) ATR_NOT_SET	The transaction mode environment setting for this *environment_id* (address space or context) has not been set. (This setting is not valid when *scope* is ATR_DEFAULT_SCOPE.)
1 (1) ATR_GLOBAL_MODE	The transaction mode is set to global for the requested scope.
2 (2) ATR_LOCAL_MODE	The transaction mode is set to local for the requested scope.
3 (3) ATR_HYBRID_GLOBAL_MODE	The transaction mode is set to hybrid-global for the requested scope. This is the same as global mode, except it allows the resource manager to exhibit proprietary connection behavior.

The value returned for ATR_NORM_CTX_END_SETTING is one of the following:

Value in: Hexadecimal (Decimal) Equate Symbol	**Description**
0 (0) ATR_NOT_SET	The two-phase commit setting at the scope specified by *environment_id* (address space or context) has not been set. If the two-phase commit environment is not set, RRS will commit the UR on normal task termination.
1 (1) ATR_COMMIT_ACTION	The action RRS will take is to commit the UR.

Value in: Hexadecimal (Decimal) Equate Symbol	Description
2 (2) ATR_ROLLBACK_ACTION	The action RRS will take is to roll back the UR.

,environment_protection
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Returns a protection value for each identifier in the *environment_id* parameter. When you specify more than one identifier, you must define an array, where *element_count* indicates the number of elements in the array. The positions of the identifiers in the *environment_id* array define the positions of the corresponding protection values in the *environment_protection* array.

The value returned is one of the following:

Value in: Hexadecimal (Decimal) Equate Symbol	Description
1 (1) ATR_UNPROTECTED_SETTING	The setting can be changed by an unauthorized caller.
2 (2) ATR_PROTECTED_SETTING	The setting can be changed only by an authorized caller.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00270000' or X'00270001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, specified a zero context token, and requested the retrieval of environment settings at a scope of ATR_CONTEXT_SCOPE. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The application is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the application for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that is running a version of RRS that supports this service call. Then rerun the resource manager.
361 ATR_CONTEXT_TOKEN_INV	**Meaning**: Program error. The context token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
362 ATR_STOKEN_INV	**Meaning**: Program error. The address space token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
364 ATR_ENV_SETTING_ID_INV	**Meaning**: Program error. A value in the *environment_id* parameter specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
366 ATR_SCOPE_INV	**Meaning**: Program error. The scope specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
392 ATR_ELEMENT_COUNT_INV	**Meaning**: Program error. The element count value in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
802 ATR_STOKEN_NOT_ZERO	**Meaning**: Program error. The stoken parameter was incorrectly specified. The stoken is not zero, but the caller specified ATR_CONTEXT_SCOPE or ATR_DEFAULT_SCOPE on the scope parameter. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
803 ATR_CTOKEN_NOT_ZERO	**Meaning**: Program error. The context token parameter was incorrectly specified. The caller specified ATR_ADDRESS_SPACE_SCOPE on the scope parameter and a non-zero value on the context token parameter. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the work manager issues a call to retrieve environmental settings at the context level.

```
⋮
SCOPE = ATR_CONTEXT_SCOPE
C_TOKEN = MY_CONTEXT_TOKEN
A_TOKEN = 0
ELE_CNT = 1
ENV_SET_ID = ATR_NORM_CTX_END_SETTING
CALL ATRRENV(RC,DIAG_DATA,SCOPE,C_TOKEN,A_TOKEN,ELE_CNT,
            ENV_SET_ID,ENV_SET,ENV_SET_PROT)
⋮
```

Retrieve_Interest_Count (ATRREIC, ATR4REIC)

- ATRREIC is for AMODE(31) callers.
- ATR4REIC is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Retrieve_Interest_Count service to determine if RRS needs to coordinate the syncpoint for a unit of recovery (UR). If multiple interests are expressed in a UR, RRS must coordinate that UR's syncpoint. If only one interest is expressed in a UR, RRS may be able to allow coordination of the syncpoint by the resource manager expressing that interest. In response to the call, RRS returns:

- A return code
- An indication of whether or not RRS must coordinate the syncpoint

  Note: The UR status may change after the call, which can affect whether or not RRS must coordinate the syncpoint.

If the UR is part of a UR family, *interest_count_info* will always return ATR_MULTIPLE_INTERESTS.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state
Dispatchable unit mode:	**Task or SRB**
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRREIC) 64 bit (ATR4REIC)
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used (but the address of a save area is not required in GPR 13).

Programming requirements

The service does not perform any error checking or have any recovery. The caller should provide recovery to handle any unexpected errors.

This service is intended for use with URs in hybrid-global transaction mode. For URs in other transaction modes or when the transaction mode is unknown, call the Retrieve_Side_Information_Fast service.

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

None.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

This service has a minimal path length.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRREIC	(return_code ,context_token ,coordinator_info)

CALL ATR4REIC	(return_code ,context_token ,coordinator_info)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Retrieve_Interest_Count service.

,context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the context token associated with the UR. Do not specify a context token of 0.

Your resource manager received the token from:

- The Begin_Context service for a privately managed context
- The Express_UR_Interest service or Express_Context_Interest service for a native context

,coordinator_info
Received parameter

- Type: Integer
- Length: 4 bytes

Receives from the service an indicator of whether or not RRS must be the syncpoint coordinator. The indicator is one of the following:

Indicator in: Hexadecimal (Decimal) Equate Symbol	Description
1 (1) ATR_NO_MORE_THAN_ONE_INTEREST	Only one resource manager has expressed only one interest in the UR. That resource manager could coordinate the syncpoint of this UR itself, or allow RRS to coordinate the syncpoint.
2 (2) ATR_MULTIPLE_INTERESTS	One or more resource managers have expressed more than one interest in the UR. RRS must coordinate the syncpoint for this UR, because only RRS has all of the information needed to properly coordinate the syncpoint.

ABEND codes

None.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Example

In the pseudocode example, the resource manager issues a call to determine if RRS must coordinate the syncpoint for the UR. Storage for the call parameters has been allocated.

```
⋮
C_TOKEN = MY_C_TOKEN
CALL ATRREIC(RC,C_TOKEN,COORD_INFO)
IF RC = ATR_OK THEN
  CURRENT_COORD_INFO = COORD_INFO
⋮
```

Retrieve_Interest_Data (ATRRID, ATR4RID)

- ATRRID is for AMODE(31) callers.
- ATR4RID is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Retrieve_Interest_Data service to retrieve data about an interest in a unit of recovery (UR). In response to the call, RRS returns:

- A return code
- The nonpersistent interest data
- The length of the persistent interest data
- The persistent interest data
- The type of interest: unprotected, protected, or protected and logged
- The type of expression of interest: new or restart
- The role the resource manager is taking in the UR interest: participant, last-agent participant, distributed syncpoint resource manager (DSRM), or server distributed syncpoint manager (SDSRM)

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRRID) 64 bit (ATR4RID)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

For the call, the UR state cannot be **in-reset**.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRRID	(return_code ,ur_interest_token ,nonpersistent_interest_data ,persistent_interest_buffer_length ,persistent_interest_data_length ,persistent_interest_data ,interest_type ,expression_of_interest_type ,role)

CALL ATR4RID	(return_code ,ur_interest_token ,nonpersistent_interest_data ,persistent_interest_buffer_length ,persistent_interest_data_length ,persistent_interest_data ,interest_type ,expression_of_interest_type ,role)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Retrieve_Interest_Data service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that identifies your resource manager's interest in the UR. Your resource manager received the token from one of the following services: Express_UR_Interest, Retrieve_UR_Interest, or Retain_Interest.

,nonpersistent_interest_data
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives from the service the nonpersistent interest data for your resource manager's interest in the UR. RRS does not record this data in nonvolatile storage.

Your resource manager provided the data in a call to one of the following services: Express_UR_Interest, Respond_to_Retrieved_Interest, or Retain_Interest.

,persistent_interest_buffer_length
Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies, in hexadecimal, the length of the buffer that your resource manager is supplying for the persistent interest data. The length in bytes can be X'0' - X'1000' (0 - 4096).

,persistent_interest_data_length
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service the actual length of the persistent interest data. The length ranges from X'0' - X'1000' (0-4096), where 0 indicates that there is no persistent interest data.

If the interest type is ATR_UNPROTECTED, then there is no persistent interest data.

,persistent_interest_data
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified in *persistent_interest_buffer_length*

Specifies a buffer that receives the persistent interest data for your resource manager's interest in the UR. If *persistent_interest_data_length* is 0, RRS ignores this parameter. Your resource manager provides this data in a call to one of the following services: Express_UR_Interest, Change_Interest_Type, Set_Persistent_Interest_Data, or Retain_Interest. Your resource manager can also retrieve persistent interest data from the Retrieve_UR_Interest service.

,interest_type
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service the interest type for the resource manager's interest in the UR. The interest type is indicated by one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	**Description**
0 (0) ATR_UNPROTECTED	**Unprotected**: The resource manager expressed an unprotected interest in the UR.

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
1 (1) ATR_PROTECTED	**Protected**: The resource manager expressed a protected interest in the UR.
2 (2) ATR_PROT_LOGGED	**Protected and logged**: The resource manager expressed a protected interest in the UR and the interest is recorded in an RRS log.

,expression_of_interest_type

Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service the expression of interest type, which indicates whether the UR is new or restart. The field contains one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) ATR_NORMAL_INTEREST	**Meaning**: The expression of interest is a normal expression (that is, not a restart expression of interest).
1 (1) ATR_RESTART_INTEREST	**Meaning**: The expression of interest is a restart expression of interest (that is, an expression of interest returned by the Retrieve_UR_Interest service).

,role

Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service the role of the resource manager in the UR interest identified in *ur_interest_token*. If your resource manager is not a participant, your resource manager specified its role through a Set_Syncpoint_Controls call. The role is indicated by one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Role
0 (0) ATR_PARTICIPANT	Participant

Constant in: Hexadecimal (Decimal) Equate Symbol	Role
1 (1) ATR_LAST_AGENT	Last-agent participant
2 (2) ATR_DSRM	Distributed syncpoint resource manager
3 (3) ATR_SDSRM	Server distributed syncpoint resource manager

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'000B0000' or X'000B0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
5 ATR_PARTIAL_PERSISTENT_DATA	**Meaning**: Program error. The *persistent_interest_buffer_length* value is less than the actual length of the persistent interest data. The system accepts the service call. RRS puts in the buffer as many characters of the data as will fit, starting at the left. **Action**: No action is required. If the result is not expected, check the resource manager for a probable coding error; correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call is not one of the currently valid interests. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
37D ATR_PERSIS_DATA_BUF_LEN_INV	**Meaning**: Program error. The length specified for the persistent interest buffer is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager state must be **restart** or **run**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Retrieve_Interest_Data was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to obtain information about an interest in a UR.

```
⋮
URI_TOKEN = MY_URI_TOKEN
PD_BUF_LEN = MY_PDATA_LEN
CALL ATRRID(RC,URI_TOKEN,NON_P_DATA,PD_BUF_LEN,PD_LEN,
     P_DATA,INT_TYPE,EI_TYPE,ROLE)
IF RC = ATR_OK THEN
  MY_P_DATA = P_DATA
  MY_INT_TYPE = INT_TYPE
  MY_ROLE = ROLE
⋮
```

Retrieve_Log_Name (ATRIRLN, ATR4IRLN)

- ATRIRLN is for AMODE(31) callers.
- ATR4IRLN is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

Before beginning restart, a resource manager should call the Retrieve_Log_Name service to obtain log names. In response to the call, RRS returns:

- A return code
- The length of the resource manager log name
- The name of the resource manager log as recorded in the RRS log
- The length of the RRS log name
- The RRS log name

The call provides the resource manager's log name only if the resource manager, when it was running before this restart, called the Set_Log_Name service to supply the log name to RRS.

Your resource manager should save the RRS log name in its resource manager log.

Comparing log names: The resource manager compares the information RRS returns with information in its own logs:

- It compares the returned resource manager log name to the name of the log the resource manager is currently using
- It compares the returned RRS log name to the RRS log name the resource manager saved when it was running before this restart

If both comparisons match, the resource manager can correlate unit of recovery (UR) data it receives from Retrieve_UR_Interest calls with data from its resource manager log. “Log name checks” on page 58 describes the comparisons and appropriate actions in more detail.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRIRLN) 64 bit (ATR4IRLN)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held

Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The resource manager associated with the specified resource manager token must be in one of the following states:

- **Set** which means it has registered and set its exit routines with RRS
- **Restart**, which means it has registered, set its exit routines with RRS, and begun restart
- **Run**, which means it has registered, set its exit routines with RRS, and completed restart

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRIRLN	(return_code ,resource_manager_token ,rm_logname_buffer_len ,rm_logname_len ,rm_logname ,rrs_logname_len ,rrs_logname)

CALL ATR4IRLN	(return_code ,resource_manager_token ,rm_logname_buffer_len ,rm_logname_len ,rm_logname ,rrs_logname_len ,rrs_logname)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Retrieve_Log_Name service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

,rm_logname_buffer_len
Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the hexadecimal length of the buffer provided in the *rm_logname* parameter. The length is X'1' - X'40' (1 - 64) bytes.

,rm_logname_len
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service:

- The actual length of the name of the resource manager log. The length ranges from X'1' - X'40' (1 - 64) bytes. The actual length is provided whether the log name fits in the *rm_logname* field or not.
- Binary zeros. The zeros indicate that the resource manager did not call the Set_Log_Name service to provide the resource manager log name to RRS.

,rm_logname
Returned parameter

- Type: Character string
- Character Set: EBCDIC
- Length: 1 - 64 bytes

Receives from the service:

- The name of the resource manager log. If the name is longer than the *rm_logname* field, RRS sets an error return code and places in the *rm_logname* field as many characters, starting at the left, as will fit.
- Blanks. The blanks indicate that the resource manager did not call the Set_Log_Name service to provide the log name to RRS.

,rrs_logname_len
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service the length of the RRS log name.

,rrs_logname
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 64 bytes

Receives from the service the RRS log name.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00050000' or X'00050001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
6 ATR_RM_LOGNAME_NOT_SET	**Meaning**: Program processing. The resource manager has not called the set log name service to provide the resource manager log name to RRS. This return code may also be issued after a successful Internal Cold Start, in response to a log stream error against the RRS RM.DATA log stream as identified by message ATR250E. In this case, RRS was not able to preserve the log name across the Internal Cold Start processing. **Action**: If this result is expected, no action is needed. If this result is not expected, check the resource manager for a probable coding error; correct the resource manager and rerun it.
9 ATR_PARTIAL_RM_LOGNAME	**Meaning**: Program error. The length of the buffer for the resource manager log name specified in the call is not long enough to contain the current resource manager log name. The system accepts the service call. RRS places in the buffer as many characters of the name as will fit, starting at the left. RRS returns the actual logname length in rm_logname_len. **Action**: No action is required. If the result is not expected, check the resource manager for a probable coding error; correct the resource manager and rerun it.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
301 ATR_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
37C ATR_RM_LOGNAME_BUF_LEN_INV	**Meaning**: Program error. The length specified for the log name buffer is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the resource manager token specified in the call is not in a valid state to issue the service call. The resource manager state must be **set**, **restart**, or **run**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to obtain its log name and the RRS log name. Storage for the call parameters has been allocated.

```
⋮
RM_TOKEN = MY_RM_TOKEN
LOGNAME_BUF_LEN = LOGNAME_BUFFER_LEN
CALL ATRIRLN(RC,RM_TOKEN,LOGNAME_BUF_LEN,LOGNAME_LEN,LOGNAME_BUFFER,
       RRS_LOGNAME_LEN,RRS_LOGNAME)
IF RC = 0 THEN
⋮
```

Retrieve_RM_Metadata (ATRRDTA, ATR4RDTA)

- ATRRDTA is for AMODE(31) callers.
- ATR4RDTA is for AMODE(64) callers, and allows parameters in 64 bit addressable storage.

A resource manager calls the Retrieve_RM Metadata service to fetch up to 8K (8192) bytes of data from RRS that the resource manager previously saved with RRS via the Set_RM_Metadata service.

If the resource manager has no metadata, a length of zero is returned.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (ATRRDTA) 64 bit (ATR4RDTA)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine *ATRRCSS* (31 bit) or *ATRR4CSS* (64 bit) from *SYS1.CSSLIB*, or *LOAD* and *CALL* the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The resource manager associated with the specified resource manager token must be in Run state, which means it has been registered, set its exit routines with RRS, and completed restart.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRRDTA	(return_code ,resource_manager_token ,rm_metadata_buffer_len ,rm_metadata_len ,rm_metadata)

CALL ATR4RDTA	(return_code ,resource_manager_token ,rm_metadata_buffer_len ,rm_metadata_len ,rm_metadata)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Set_RM_Metadata service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

,rm_metadata_buffer_len
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies the length of the buffer that your resource manager is supplying for the metadata. The length in bytes can be 1 to 8K (8192).

,rm_metadata_len
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

The actual length of the RM metadata. The length ranges from 0 to 8192 bytes. A zero length indicates that the resource manager does not have any logged metadata with RRS.

,rm_metadata
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 0-8192 byes

Specifies the buffer to contain the resource manager's metadata.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00290000' or X'00290001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
B ATR_PARTIAL_RM_METADATA	**Meaning**: Program error. The *rm_metadata_buffer_len* value is less than the actual length of the resource manager's metadata. The system accepts the service call. RRS returns in the buffer as much data as will fit. **Action**: No action is required. If the result is not expected, check the resource manager for a probable coding error. Correct the resource manager and rerun it.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
301 ATR_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not valid. The system rejects the service call. Action: Check the resource manager for a probable coding error. Correct the resource manager and rerun it. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
38B ATR_RM_METADATA_BUFFER_LEN_INV	**Meaning**: Program error. The length of the resource manager metadata buffer specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
38C ATR_RM_METADATA_LOG_ UNAVAILABLE	**Meaning**: The MetaData callable service failed since the resource manager MetaData log stream is not available. **Action**: Check SYSLOG for messages ATR132I or ATR172E that will further explain why the log is unavailable.
38D ATR_RM_8K_METADATA_NOT_ ALLOWED	**Meaning**: The resource manager did not set the *ATR_8K_RM_METADATA_REQUESTED* flag on *CRGSEIF/CRGSEIF1/CRG4SEIF* so the resource manager cannot set or retrieve 8K Meta Data. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
38E ATR_RM_METADATA_MISSING_DATA	**Meaning**: When reading from the RM Meta Data log stream, records were encountered that indicate there was a loss of data or a gap in the log stream. If Meta Data was stored for the RM, it cannot be found. **Action**: Check SYSLOG for messages ATR202D and ATR212I that will further explain the error and how to correct it.
701 ATR_RM_STATE_ERROR	**Meaning**: The resource manager associated with the resource manager token specified in the call is not in a valid state to issue the service call. The resource manager state must be run. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to retrieve its RM Metadata from RRS that it had previously set.

```
.
.
.
RM_TOKEN = MY_RM_TOKEN
BUFFER_LEN = MY_BUFFER_LEN
CALL ATRRDTA(RC,RM_TOKEN,BUFFER_LEN,META_LEN,META)
IF RC=0 THEN
.
.
.
```

Retrieve_Side_Information (ATRRUSI, ATRRUSI2, ATR4RUSI)

The resource manager calls the Retrieve_Side_Information service to retrieve side information for an interest in a unit of recovery (UR). In response to the call, RRS returns:

- A return code
- The side information

There are three versions of Retrieve_Side_Information, each with different parameters.

- ATRRUSI is for AMODE(31) callers and is the basic version of the service. It must be called specifying a UR interest token.
- ATRRUSI2 is for AMODE(31) callers and can be called specifying either a UR token or a UR interest token.

- ATR4RUSI is for AMODE(64) callers, allows parameters in 64 bit addressable storage, and can be called specifying either a UR token or a UR interest token.

Code your resource manager to call the version that includes the support you need.

Side information: The side information is set by RRS or, in a call to the Set_Side_Information service, by a resource manager that is interested in the UR. Much of the side information is set only by a resource manager that uses Systems Network Architecture (SNA) Logical Unit (LU) 6.2 sync point architecture. See the Set_Side_Information callable service for a description of side information.

Information about other resource managers: Your resource manager can use a Retrieve_Side_Information call to obtain information about another resource manager that is interested in the UR. For example, if the service returns `ATR_NEW_LUWID_PSH_UNACCEPTABLE`, your resource manager knows that an LU 6.2 communications resource manager cannot send a new LUWID on any LU 6.2 conversation that it is managing.

Parameter arrays: The *side_info_id* parameter is an input array; each position identifies side information the resource manager wants from RRS. The *side_info_state* parameter is an output array; RRS places in each position the side information requested by the corresponding position in the *side_info_id* array. The *element_count* parameter indicates the number of positions in both arrays.

For example, if the call specifies in the fourth position of *side_info_id* `ATR_BACKOUT_REQUIRED`, the fourth position of *side_info_state* will indicate if backout required is or is not set for the UR interest.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (ATRRUSI, ATRRUSI2) 64 bit (ATR4RUSI)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The state of the resource manager associated with the specified UR interest token must be:

- **Restart**, which means it has registered, set its exit routines with RRS, begun restart, and requested incomplete UR interests
- **Run**, which means it has registered, set its exit routines with RRS, and completed restart

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the appropriate call as shown in the syntax diagrams. You must code the parameters in the CALL statement as shown.

CALL ATRRUSI	(return_code ,ur_interest_token ,element_count ,side_info_id ,side_info_state)

CALL ATRRUSI2	(return_code ,ur_or_uri_token ,element_count ,side_info_id ,side_info_state)

CALL ATR4RUSI	(return_code ,ur_or_uri_token ,element_count ,side_info_id ,side_info_state)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Retrieve_Side_Information service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRRUSI callers, specifies a token that uniquely identifies your resource manager's interest in the UR whose side information you want to retrieve. Your resource manager received the token from one of the following services: Express_UR_Interest, Retrieve_Interest_Data, Retain_Interest.

,ur_or_uri_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRRUSI2 callers, specifies a token that uniquely identifies either the UR, or your resource manager's interest in the UR, whose side information you want to retrieve:

- UR token: The token for the UR.
- UR interest token: The UR interest token that identifies your resource manager's interest in the UR.

Your resource manager received the token from one of the following services: Express_UR_Interest, Retrieve_Interest_Data, Retain_Interest, Create_Cascaded_UR, or Retrieve_UR_Data.

Because you may pass two different types of tokens through this parameter, passing an invalid token can generate either a ATR_URI_TOKEN_INV or a ATR_UR_TOKEN_INV return code. For example, passing an invalid UR token might result in an ATR_URI_TOKEN_INV return code. Even though a UR token was passed, if it is invalid, then RRS may not understand what sort of token it was supposed to be. For this reason, IBM recommends callers check both return codes, even when they know what type of token they intend to pass.

,element_count

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the number of elements in the array for the *side_info_id* and *side_info_state* parameters. Both arrays must have the same number of elements. The maximum count is 13.

,side_info_id

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies one or more identifiers; each identifier requests the state of side information that RRS or a resource manager might have set. When you specify more than one identifier, you must define an array, where *element_count* indicates the number of identifiers. The positions of the identifiers in this *side_info_id* array define the positions of the side information states to be returned in the *side_info_state* array.

Specify each identifier as one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	**Identifier**
0 (0) ATR_HEURISTIC_MIX	Heuristic-mixed condition A heuristic commit occurred while the UR state was **in_backout**, a heuristic reset occurred while the UR state was **in_commit**, or a resource manager of distributed resources informed RRS of a heuristic-mixed condition. When this information is initially set through a call to Set_Side_Information, RRS will harden (or reharden) the UR state, including this identifier, immediately.

Constant in: Hexadecimal (Decimal) Equate Symbol	Identifier
1 (1) ATR_BACKOUT_REQUIRED	Backout required When this identifier is initially set, RRS will force the current UR to back out when a syncpoint occurs. If a UR state has passed beyond **in_prepare**, RRS ignores this identifier.
10 (16) ATR_BREAK_TREE	Break tree When this identifier is initially set, RRS will reset the logical unit of work identifier (LUWID) for the next UR.
11 (17) ATR_DRIVE_BACKOUT	Backout of next UR When this identifier is initially set, RRS will, if any resource manager has called Retain_Interest for the next UR, complete backout for the next UR before returning control to the application program. This processing ensures that the next UR will be backed out.
12 (18) ATR_RESYNC_IN_PROGRESS	Resync in progress A resync in progress condition has occurred. If the UR state is before **in_end** when this identifier is initially set, RRS will harden (or reharden) the UR state, including this identifier, at the next state change.
13 (19) ATR_NEW_LUWID_PSH_UNACCEPTABLE	New LUWID PSH unacceptable A communication resource manager cannot accept a new LUWID presentation header (PSH). RRS takes no action when this identifier is set.
14 (20) ATR_DRIVE_COMPLETION	Invoke completion exit(s) When this identifier is initially set, RRS will invoke all COMPLETION exit routines, if any exist, when the UR state reaches **in-completion**. RRS will harden (or reharden) the UR state, including this identifier, at the next logging point.
15 (21) ATR_SDSRM_INITIATED	Syncpoint operation initiated by resource manager When this identifier is set, the current syncpoint operation was initiated by a resource manager that has taken the SDSRM role, then called Backout_Agent_UR or Prepare_Agent_UR. RRS will harden (or reharden) this identifier whenever it normally hardens (or rehardens) the UR state.

Constant in: Hexadecimal (Decimal) Equate Symbol	Identifier
16 (22) ATR_RESOLVED_BY_INSTALLATION	Installation resolved UR When this identifier is set, an **in-doubt** UR has been resolved through the RRS panels or a program that issued the ATRSRV macro. RRS will harden (or reharden) the UR state, including this identifier, immediately.
17 (23) ATR_TERM_SYNCPOINT	Terminating syncpoint When this identifier is set, the UR is going through syncpoint processing because its context has ended. (This condition is always true for URs created by restart. When a resource manager is restarting when RRS has not failed, RRS might "reconnect" the interest returned through Retrieve_UR_Interest to an existing UR. In this case, the UR might not be marked for terminating syncpoint processing.)
18 (24) ATR_COMMITTED	Committed UR When this identifier is set, the outcome for the current UR has been determined, and the result is a commit. RRS always hardens this identifier when it hardens the commit.
20 (32) ATR_IMMEDIATE_BACKOUT	Application requested backout The backout occurred because the application, either implicitly or explicitly, requested it, not because a resource manager could not commit its resources. RRS hardens this identifier whenever it normally logs status for the UR.
21 (33) ATR_APPL_COMPLETE	Application processing is complete This identifier indicates completion of an individual UR in a cascaded UR family. RRS sets this identifier when it is informed that the application executing for this UR is complete. RRS will not commit a cascaded UR family until RRS is informed that all of the individual cascaded URs in the family are complete. **Note:** If RRS has not set this identifier, it does not necessarily mean that the application execution is incomplete; it just means RRS is unaware of the completion.

Constant in: Hexadecimal (Decimal) Equate Symbol	Identifier
23 (35) ATR_SI_LOCAL_MODE	Local transaction mode When this identifier is set, the UR is in local transaction mode and ATR_SI_GLOBAL_MODE cannot be set. When neither ATR_SI_LOCAL_MODE nor ATR_SI_GLOBAL_MODE is set, the transaction mode is hybrid-global.
24 (36) ATR_SI_GLOBAL_MODE	Implicit global transaction mode When this identifier is set, the UR is in global transaction mode and ATR_SI_LOCAL_MODE cannot be set. When neither ATR_SI_LOCAL_MODE nor ATR_SI_GLOBAL_MODE is set, the transaction mode is hybrid-global.

,side_info_state
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives one or more indicators from the service. Each indicator shows whether or not its matching identifier is set in the side information. This array must have the same number of positions as the *side_info_id* array. For each identifier, the service returns one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	State of the side information
0 (0) ATR_SIDE_VALUE_NOT_SET	**Side value not set**: The side information value is not set. Neither RRS nor a resource manager has set it, or it has been reset.
1 (1) ATR_SIDE_VALUE_SET	**Side value set**: Either RRS or a resource manager set the side information value.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'000D0000' or X'000D0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call does not identify one of the currently valid interests. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a UR token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
383 ATR_SIDE_INFO_ID_INV	**Meaning**: Program error. The identifier for a side information value in the *side_info_id* parameter specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
392 ATR_ELEMENT_COUNT_INV	**Meaning**: The specified element count is not valid. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
3A3 ATR_UR_TOKEN_INV	**Meaning**: Program error. The UR token specified in the call does not identify a valid UR. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a URI token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager state must be **restart** or **run**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F06 ATR_WAS_NOT_AVAILABLE	**Meaning:** RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Retrieve_Side_Information was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action:** The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to request side information for an interest in a UR. Storage for the call parameters has been allocated.

```
⋮
URI_TOKEN = UR_INTEREST_TOKEN
COUNT = 2
ID(1) = ATR_HEURISTIC_MIX
ID(2) = ATR_RESYNC_IN_PROGRESS
CALL ATRRUSI2(RC,URI_TOKEN,COUNT,ID,STATE)
IF RC = ATR_OK THEN
  HM = STATE(1)
```

```
  RIP = STATE(2)
⋮
```

Retrieve_Side_Information_Fast (ATRRUSF, ATRRUSF1, ATR4RUSF)

The resource manager calls the Retrieve_Side_Information_Fast service to retrieve the current settings of RRS-related environment attributes for the UR associated with the specified context. There are three versions of Retrieve_Side_Information_Fast:

- ATRRUSF is for AMODE(31) callers and is the basic version of this service.
- ATRRUSF1 is for AMODE(31) callers and adds support for work managers that need interest count data, even if an event has occurred that requires RRS to coordinate the syncpoint.
- ATR4RUSF is for AMODE(64) callers, allows parameters in 64 bit addressable storage, and adds support for work managers that need interest count data, even if an event has occurred that requires RRS to coordinate the syncpoint.

This service returns information about:

1. The mode of a UR
2. Who can or must coordinate a UR
3. How many interests exist for a UR (The interest count is only available with ATRRUSF1 and ATR4RUSF. This information is optional, and is returned only when the caller specifically requests it.)

If a condition has occurred that requires RRS to coordinate the syncpoint, such as setting an XID or a post-syncpoint PET, then RRS must coordinate the resources, even if no resource manager has expressed an interest in the UR.

Environment

The requirements for the caller are:

Minimum authorization:	Any
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRRUSF, ATRRUSF1) 64 bit (ATR4RUSF)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

This service does not perform any error checking or provide any recovery. The caller must provide recovery to handle any unexpected errors.

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

None.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

The service has a minimal path length.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRRUSF	(return_code ,context_token ,environment_info)

CALL ATRRUSF1	(return_code ,context_token ,environment_info ,side_information_options)

CALL ATR4RUSF	(return_code ,context_token ,environment_info ,side_information_options)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Retrieve_Side_Information_Fast service.

,context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the context token associated with the UR. Do not specify a context token of 0.

Your resource manager received the context token from Retrieve_Current_Context, Begin_Context for a privately managed context, or Express_UR_Interest or Express_Context_Interest for a native context.

,side_information_options
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Valid on ATRRUSF1 and ATR4RUSF only. Specifies one or more options to request the type of information to be returned by RRS. Any undefined bits are reserved and set to zero. You must ignore the reserved bits.

The parameter will have one or more bits turned on that can be referenced by the following mask:

Mask in: Hexadecimal Equate Symbol	Description
00000001 ATR_INTEREST_COUNT_MASK	A resource manager specifies this mask to request RRS to return the interest count information for the UR associated with the specified context. The interest count information ATRRUSF1/ATR4RUSF returns is valid only for a UR that is **in-flight**. You must not use the interest count information ATRRUSF1/ATR4RUSF returns if the UR is in any other state.
00000002 ATR_CASCADED_TRANSACTION_MASK	A resource manager specifies this mask to request RRS to return the cascaded transaction information for the UR that is associated with the specified context. This is for ATRRUSF1/ATR4RUSF callers on systems where the ATRPre_PrepareExitSupport flag is on in ATRRINST.

,environment_info

Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Receives UR environment information. Any undefined bits are reserved and set to zero. You must ignore the reserved bits.

The parameter will have one or more bits turned on that can be referenced by the following masks:

Mask in: Hexadecimal Equate Symbol	Description
00000001 ATR_NO_INTERESTS_MASK	There are no interests in the current UR. There are no protected resources to commit. Only one of the following indicators is set for a given UR at any given time: • ATR_NO_INTERESTS_MASK • ATR_RM_COORD_OK_MASK • ATR_RRS_MUST_COORD_MASK **Note:** This bit is never set if ATR_LOCAL_MODE_MASK is on.

Mask in: Hexadecimal Equate Symbol	Description
00000002 ATR_RM_COORD_OK_MASK	The UR has one or more expressions of interest by the same resource manager. The resource manager that owns the expression (or expressions) of interest can coordinate its own resources. **Note:** Do not confuse this condition with local transaction mode. Generally, when the resource manager decides to bypass RRS calls, the UR is in global transaction mode and will remain in **in-flight** state and global transaction mode. Only one of the following indicators is set for a given UR at any given time: • ATR_NO_INTERESTS_MASK • ATR_RM_COORD_OK_MASK • ATR_RRS_MUST_COORD_MASK
00000004 ATR_RRS_MUST_COORD_MASK	One of the following conditions have occurred that require RRS to coordinate the syncpoint: • Multiple resource managers have expressed interest in the UR • A resource manager has set an XID for the UR • A work manager has set a post syncpoint PET to be associated with the UR so it can know when the transaction completes • The UR of interest is a child UR If any of the above occurs, RRS must coordinate the syncpoint for this UR, because only RRS has all the information needed to properly coordinate the syncpoint. Only one of the following indicators is set for a given UR at any given time: • ATR_NO_INTERESTS_MASK • ATR_RM_COORD_OK_MASK • ATR_RRS_MUST_COORD_MASK

Mask in: Hexadecimal Equate Symbol	Description
00000010 ATR_ZERO_INTEREST_COUNT_MASK	There are no interests in the current UR. This information is returned if the caller specifies the ATR_ZERO_INTEREST_COUNT_MASK in the side_information_options. For information regarding who can and must coordinate the syncpoint for the UR, check the ATR_NO_INTEREST_MASK, ATR_RM_COORD_OK_MASK, and ATR_RRS_MUST_COORD_MASK. Only one of the following interest count indicators is set for a given UR at any given time: • ATR_ZERO_INTEREST_COUNT_MASK • ATR_ONE_INTEREST_COUNT_MASK • ATR_MULTIPLE_INTEREST_ COUNT_MASK
00000020 ATR_ONE_INTEREST_COUNT_MASK	Only one resource manager has expressed only one interest in the UR. This information is returned if the caller specifies the ATR_INTEREST_COUNT_MASK in the side_information_options. For information regarding who can and must coordinate the syncpoint for the UR, check the ATR_NO_INTEREST_MASK, ATR_RM_COORD_OK_MASK, and ATR_RRS_MUST_COORD_MASK. Only one of the following interest count indicators is set for a given UR at any given time: • ATR_ZERO_INTEREST_COUNT_MASK • ATR_ONE_INTEREST_COUNT_MASK • ATR_MULTIPLE_INTEREST_ COUNT_MASK

Mask in: Hexadecimal Equate Symbol	Description
00000040 ATR_MULTIPLE_INTEREST_ COUNT_MASK	There are two or more interests in the UR; either one resource manager has multiple interests, or multiple resource managers have one or more interests. This information is returned if the caller specifies the ATR_INTEREST_COUNT_MASK in the side_information_options. For information regarding who can and must coordinate the syncpoint for the UR, check the ATR_NO_INTEREST_MASK, ATR_RM_COORD_OK_MASK, and ATR_RRS_MUST_COORD_MASK. Only one of the following interest count indicators is set for a given UR at any given time: • ATR_ZERO_INTEREST_COUNT_MASK • ATR_ONE_INTEREST_COUNT_MASK • ATR_MULTIPLE_INTEREST_ COUNT_MASK
00000100 ATR_UR_STATE_IN_RESET_MASK	When set, the UR state is **in-reset**. No other indicators will be set when this indicator is returned.
00000200 ATR_UR_CASCADED_MASK	This is for ATRRUSF1/ATR4RUSF callers. When set, the UR is a cascaded UR, regardless if the UR is a parent or a child UR, or if the transaction is locally or sysplex cascaded. If this bit is set, the ATR_RRS_MUST_COORD_MASK indicator is also set to indicate that RRS must coordinate the syncpoint. This information is returned if the caller specifies the ATR_CASCADED_TRANSACTION_MASK in the side_information_options.
00010000 ATR_GLOBAL_MODE_MASK	The UR transaction mode is global, and the UR state is beyond **in-reset**. This setting is valid only when ATR_UR_STATE_IN_RESET_MASK is not set because the transaction mode for the UR has not yet been determined. Only one of the following indicators is set for a given UR: • ATR_GLOBAL_MODE_MASK • ATR_LOCAL_MODE_MASK • ATR_HYBRID_GLOBAL_MASK

Mask in: Hexadecimal Equate Symbol	Description
00020000 ATR_LOCAL_MODE_MASK	The UR transaction mode is local, and the UR state is beyond **in-reset**. No interest information will be returned when this bit is on, since RRS assumes that the resource manager is always the coordinator for a local transaction. When this indicator is set, no other indicators are set.
00040000 ATR_HYBRID_GLOBAL_MASK	The UR transaction mode is hybrid-global, and the UR state is beyond **in-reset**. RRS considers the UR to be a global transaction; however, resource managers may exhibit proprietary transactional behaviors. This setting is valid only when ATR_UR_STATE_IN_RESET_MASK is not set because the transaction mode for the UR has not yet been determined. Only one of the following indicators is set for a given UR: • ATR_GLOBAL_MODE_MASK • ATR_LOCAL_MODE_MASK • ATR_HYBRID_GLOBAL_MASK

ABEND codes

None.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports this level of RRS. Rerun the resource manager.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
3AF ATR_SIDE_INFORMATION_OPTIONS_INV	**Meaning**: Program error. The side_information_options value specified on the call is invalid. Either reserved bits were nonzero, or an unacceptable selection of options was specified. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later after RRS has been restarted.

Example

In the pseudocode example, the resource manager issues a call to request side information for the UR associated with a specific context. Storage for the call parameters has been allocated.

```
⋮
CTX_TOKEN = MY_CONTEXT_TOKEN
CALL ATRRUSF(RC,CTX_TOKEN,ENV_INFO)
IF RC = ATR_OK THEN
  CURRENT_ENV_INFO = ENV_INFO
  LOCAL_MODE = CURRENT_ENV_INFO && ATR_LOCAL_MODE_MASK
⋮
```

Retrieve_UR_Data (ATRRURD, ATRRURD1, ATRRURD2, ATR4RURD)

The resource manager calls the Retrieve_UR_Data service to retrieve data for a unit of recovery (UR). There are four versions of Retrieve_UR_Data, each with different parameters.

- ATRRURD is for AMODE(31) callers and is the basic version of the service. It must be called specifying a UR interest token.
- ATRRURD1 is for AMODE(31) callers and can be called specifying either a UR token or a UR interest token, and also supports the states option parameter.
- ATRRURD2 is for AMODE(31) callers and can be called specifying either a UR token or a UR interest token, supports the states option parameter, and returns a UR token for a new unit of recovery.
- ATR4RURD is for AMODE(64) callers, allows parameters in 64 bit addressable storage, and can be called specifying either a UR token or a UR interest token, supports the states option parameter, and returns a UR token for a new unit of recovery.

Code your resource manager to call the version that includes the support you need. In response to the call, RRS returns:

- A return code
- The UR identifier (URID)

- The UR state: **in-reset**, **in-flight**, **in-state-check**, **in-prepare**, **in-doubt**, **in-commit**, **in-backout**, **in-end**, **in-only-agent**, **in-completion**, or **in-forget**.
- For ATRRURD2 calls, the UR token

For ATRRURD1, ATRURD2, and ATR4URD callers, the list of returned states is dependent on the value specified in the *states_option*.

Environment

The requirements for the caller are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRRURD, ATRRURD1, ATRRURD2) 64 bit (ATR4RURD)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The state of the resource manager associated with the specified nonzero UR interest token must be:

- **Restart**, which means it has registered, set its exit routines with RRS, begun restart, and requested incomplete UR interests. Because Retrieve_UR_Data must supply a UR interest token as input, your resource manager must have called the Retrieve_UR_Interest service to retrieve this token for a restart UR interest.
- **Run**, which means it has registered, set its exit routines with RRS, and completed restart

When the resource manager issues the call in SRB mode, the call cannot specify a *ur_interest_token* or *ur_or_uri_token* of 0, indicating information for the current UR.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

IBM recommends using ATR_EXTENDED_STATES for states_option when a neither a UR identifier nor a URI token is needed. This will give you better performance and reduce use of system storage.

Syntax

Write the appropriate call as shown in the syntax diagrams. You must code the parameters in the CALL statement as shown.

CALL ATRRURD	(return_code ,ur_interest_token ,ur_identifier ,ur_state)

CALL ATRRURD1	(return_code ,ur_or_uri_token ,ur_identifier ,ur_state ,states_option)

CALL ATRRURD2	(return_code ,ur_or_uri_token ,ur_identifier ,ur_state ,states_option ,ur_token)

CALL ATR4RURD	(return_code ,ur_or_uri_token ,ur_identifier ,ur_state ,states_option ,ur_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Retrieve_UR_Data service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRRURD callers, specifies a token that uniquely identifies your resource manager's interest in the UR whose data you want to retrieve. Your resource manager received the token from one of the following services: Express_UR_Interest, Retrieve_Interest_Data, Retain_Interest.

When specifying the token, use one of the following:

- 0: Binary zeros specify the UR data associated with the current context of the current dispatchable unit.
- token: The UR interest token for an interest in a UR.

,ur_or_uri_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRRURD1 or ATRRURD2 callers, specifies a token that uniquely identifies either the UR, or your resource manager's interest in the UR, whose data you want to retrieve:

- 0: Binary zero specifies the current UR associated with the application task.
- UR token: The token for the UR.
- UR interest token: The UR interest token that identifies your resource manager's interest in the UR.

Your resource manager received the token from one of the following services: Express_UR_Interest, Retrieve_Interest_Data, Retain_Interest, Create_Cascaded_UR, or Retrieve_UR_Data.

Because you may pass two different types of tokens through this parameter, passing an invalid token can generate either a ATR_URI_TOKEN_INV or a ATR_UR_TOKEN_INV return code. For example, passing an invalid UR token might result in an ATR_URI_TOKEN_INV return code. Even though a UR token was passed, if it is invalid, then RRS may not understand what sort of token it was supposed to be. For this reason, IBM recommends callers check both return codes, even when they know what type of token they intend to pass.

,ur_identifier
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

If the returned *UR_state* is not ATR_IN_RESET, receives from the service the UR identifier (URID) that uniquely identifies the UR. If the returned *UR_state* is ATR_IN_RESET, contains binary zero.

,ur_state
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service the state of the UR. The UR state is indicated by one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	**UR state**
0 (0) ATR_IN_RESET	**In-reset** This value is only returned if *states_option* is ATR_EXTENDED_STATES.
1 (1) ATR_IN_FLIGHT	**In-flight**
2 (2) ATR_IN_STATE_CHECK	**In-state-check**
3 (3) ATR_IN_PREPARE	**In-prepare**

Constant in: Hexadecimal (Decimal) Equate Symbol	UR state
4 (4) ATR_IN_DOUBT	**In-doubt**
5 (5) ATR_IN_COMMIT	**In-commit**
6 (6) ATR_IN_BACKOUT	**In-backout**
7 (7) ATR_IN_END	**In-end**
8 (8) ATR_IN_ONLY_AGENT	**In-only-agent**
9 (9) ATR_IN_COMPLETION	**In-completion**
B (11) ATR_IN_FORGET	**In-forget**

,states_option
Supplied parameter

- Type: Integer
- Length: 4 bytes

For ATRRURD1 or ATRRURD2 callers, defines what states RRS may return for the specified UR. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	UR state
0 (0) ATR_STANDARD_STATES	One of the following states will be returned: **In-flight**, **In-state-check**, **In-prepare**, **In-doubt**, **In-commit**, **In-backout**, **In-end**, **In-only-agent**, **In-completion**, or **In-forget**. A URID will always be returned in *UR_state*.

Constant in: Hexadecimal (Decimal) Equate Symbol	UR state
1 (1) ATR_EXTENDED_STATES	One of the following states will be returned: **In-reset**, **In-flight**, **In-state-check**, **In-prepare**, **In-doubt**, **In-commit**, **In-backout**, **In-end**, **In-only-agent**, **In-completion**, or **In-forget**. If **In-reset** is returned for UR state, the value returned in *UR_identifier* will always be binary zero (which is not a valid URID).

If ATR_STANDARD_STATES is specified and the UR was **In-reset**, the UR is changed to the **In-flight** state. An **In-flight** UR cannot be made into a cascaded UR.

ur_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRRURD2 callers, receives the UR token that uniquely represents the new unit of recovery.

Note: UR tokens do not persist across restarts of the resource manager, RRS, or the system.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'000C0000' or X'000C0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
104 ATR_MODE_INV	**Meaning**: Program error. The calling program specified 0 in *context_token*, indicating the current context, but the calling program is not in task mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call does not identify one of the currently valid interests. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a UR token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
398 ATR_STATES_OPTION_INV	**Meaning**: Program error. The state option specified in the call is not valid. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
3A3 ATR_UR_TOKEN_INV	**Meaning**: Program error. The UR token specified in the call does not identify a valid UR. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a URI token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager state must be **restart** or **run**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning:** RRS is not available. **Action:** The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Retrieve_UR_Data was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to request data about the current UR. Storage for the call parameters has been allocated.

```
⋮
URI_TOKEN = 0
STATES_OPT = ATR_STANDARD_STATES
CALL ATRRURD2(RC,URI_TOKEN,URID,UR_STATE,STATES_OPT,
              UR_TOKEN)
IF RC = ATR_OK THEN
  CURRENT_URID = URID
  CURRENT_STATE = UR_STATE
  CURRENT_URTOKEN = UR_TOKEN
⋮
```

Retrieve_UR_Interest (ATRIRNI, ATR4IRNI)

- ATRIRNI is for AMODE(31) callers.
- ATR4IRNI is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

When restarting, a resource manager calls the Retrieve_UR_Interest service to retrieve information about its interest in an incomplete, protected unit of recovery (UR). In response to the call, RRS returns:

- A return code.
- The context token for the current context of the incomplete UR.
- The UR interest token to identify the resource manager's interest in the incomplete UR. You need this token for many calls to RRS services.
- The UR identifier (URID) for the incomplete UR.
- The role the resource manager has in the UR interest: participant, last agent participant, distributed syncpoint resource manager, or server distributed syncpoint manager. For a description of each role, see “Set_Syncpoint_Controls (ATRSSPC, ATR4SSPC)” on page 509.
- The state of the incomplete UR: **in-doubt**, **in-commit**, or **in-backout**.
- The length of the persistent interest data.
- The persistent interest data.

Note: RRS does not return information about URs in local transaction mode.

After the call, the resource manager should call the Respond_to_Retrieved_Interest service to process the incomplete UR interest. If the resource manager does not call the Respond_to_Retrieved_Interest service, RRS will return the incomplete UR

interest in a Retrieve_UR_Interest call each time the resource manager restarts until the resource manager completes processing of the UR interest or RRS undergoes a cold start.

URID: When your resource manager was running at an earlier time, it saved, with the UR data in its resource manager log, the URID returned by any of the following services: Express_UR_Interest, Retrieve_UR_Data, Change_Interest_Type, or Retain_Interest. The Retrieve_UR_Interest call provides your resource manager with the URID for an incomplete UR. Compare the URID from the Retrieve_UR_Interest call with URIDs in your resource manager log to find the data for the incomplete UR.

If your resource manager log includes a URID for an incomplete UR that is not returned by any Retrieve_UR_Interest call, do not make the UR's changes in the resource; treat the UR as though its state was **in-backout**. In contrast, if Retrieve_UR_Interest returns an incomplete UR that is not in your resource manager log, tell RRS that the UR interest is complete. After a successful Internal Cold Start, in response to a log stream error against the RRS RM.DATA log stream as identified by message:

```
ATR250E RRS LOGSTREAM ERROR FOUND. CORRECT THE ERROR OR OPTIONALLY REPLY
        COLDSTART TO BEGIN A RRS INTERNAL COLD START.
```

complete URs could be returned. If the UR is not in your resource manager log, tell RRS that the UR interest is complete.

Specifying retrieve interest calls: Your resource manager should call the Retrieve_UR_Interest service repeatedly to receive UR data for all of its incomplete interests. If the resource manager expressed protected interest multiple times for one UR, the Retrieve_UR_Interest service returns each interest separately. When all UR interests have been returned, the call returns the `ATR_NO_MORE_INCOMPLETE_INTERESTS` code.

Note: Retrieve_UR_Interest can be invoked in parallel. (It can be called from multiple threads simultaneously.) If you exploit the parallel retrieval of interests, then you should continue retrieving interests until *all* parallel threads receive the `ATR_NO_MORE_INCOMPLETE_INTERESTS` return code, not just the first thread.

UR states: The states given for the incomplete URs are:

- **In-doubt**: The state of the incomplete UR needs to be resolved.
- **In-commit**: The failure occurred after the UR was committed, but before the log record was physically deleted from the RRS log. The resource manager should change the resource.
- **In-backout**: The resource manager should not change the resource because the UR was to be backed out.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRIRNI)
	64 bit (ATR4IRNI)

ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage mode:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The state of the resource manager associated with the specified resource manager token must be **restart**, which means it has registered, set its exit routines with RRS, and begun restart.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller

depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRIRNI	(return_code ,resource_manager_token ,context_token ,ur_interest_token ,ur_identifier ,role ,ur_state ,persistent_interest_buffer_length ,persistent_interest_data_length ,persistent_interest_data)

CALL ATR4IRNI	(return_code ,resource_manager_token ,context_token ,ur_interest_token ,ur_identifier ,role ,ur_state ,persistent_interest_buffer_length ,persistent_interest_data_length ,persistent_interest_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Retrieve_UR_Interest service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

,context_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives from the service the context token that identifies the current context for the incomplete UR.

,ur_interest_token
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives from the service the UR interest token that identifies your resource manager's interest in the incomplete UR.

,ur_identifier
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Receives from the service a UR identifier that uniquely identifies the UR.

,role
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service the role of the resource manager in the UR interest identified by the returned UR interest token. If your resource manager is not a participant, your resource manager specified its role through a Set_Syncpoint_Controls call. The role is indicated by one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	**Role**
0 (0) ATR_PARTICIPANT	Participant
1 (1) ATR_LAST_AGENT	Last-agent participant
2 (2) ATR_DSRM	Distributed syncpoint resource manager

Constant in: Hexadecimal (Decimal) Equate Symbol	Role
3 (3) ATR_SDSRM	Server distributed syncpoint resource manager

For more information about each role, see "Set_Syncpoint_Controls (ATRSSPC, ATR4SSPC)" on page 509.

,ur_state
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service the state of the incomplete UR. The UR state is indicated by one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	UR state
4 (4) ATR_IN_DOUBT	**In-doubt**
5 (5) ATR_IN-COMMIT	**In-commit**
6 (6) ATR_IN-BACKOUT	**In-backout**

,persistent_interest_buffer_length
Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the length of the buffer that your resource manager provides for the persistent interest data. The value can be X'0' -X'1000' (0 - 4096).

,persistent_interest_data_length
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives from the service the actual length of the persistent interest data. The value can range from X'0' to X'1000' (4096), where 0 indicates that there is no data.

,persistent_interest_data
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified on *persistent_interest_buffer_length*

Provides a buffer to receive persistent interest data from the service. Your resource manager provided the data in a call to one of the following services: Express_UR_Interest, Change_Interest_Type, Set_Persistent_Interest_Data, or Retain_Interest.

Your resource manager can also retrieve persistent interest data from the Retrieve_Interest_Data (ATRRID) service (see the topic on "Retrieve_Interest_Data (ATRRID, ATR4RID)" on page 395).

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00060000' or X'00060001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and Action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
4 ATR_NO_MORE_INCOMPLETE_ INTERESTS	**Meaning**: Normal processing. RRS has no more incomplete UR interests for your resource manager. The parameters do not contain valid data. **Action**: Call the end restart service to complete restart processing.
5 ATR_PARTIAL_PERSISTENT_DATA	**Meaning**: Program error. The *persistent_interest_buffer_length* value is less than the actual length of the persistent interest data. The system accepts the service call. RRS places in the buffer as many characters of the data as will fit, starting at the left. **Action**: No action is required. If the result is not expected, check the resource manager for a probable coding error; correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and Action
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
301 ATR_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
37D ATR_PERSIS_DATA_BUF_LEN_INV	**Meaning**: Program error. The length specified for the persistent interest buffer is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the resource manager token specified in the call is not in a valid state to issue the service call. The resource manager state must be **restart**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and Action
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to request its interest in an incomplete UR. Storage for the call parameters has been allocated.

```
⋮
RM_TOKEN = MY_RM_TOKEN
PD_BUF_LEN = 256
DO UNTIL (RC=ATR_NO_MORE_INCOMPLETE_INTERESTS)
  CALL ATRIRNI(RC,RM_TOKEN,C_TOKEN,URI_TOKEN,URID,ROLE,
       UR_STATE,PD_BUF_LEN,PD_DATA_LEN,PD_DATA)
  IF RC ≠ 0 THEN
     /* Handle error */
  END DO
⋮
```

Retrieve_Work_Identifier (ATRRWID, ATRRWID2, ATR4RWID)

The resource manager calls the Retrieve_Work_Identifier service to retrieve a work identifier related to a unit of recovery (UR). In response to the call, RRS returns:

- A return code
- The actual length of the unit of work identifier (UWID)
- The UWID

There are three versions of Retrieve_Work_Identifier, each with different parameters.

- ATRRWID is for AMODE(31) callers and is the basic version of the service. It must be called specifying a UR interest token.
- ATRRWID2 is for AMODE(31) callers and can be called specifying either a UR token or a UR interest token.
- ATR4RWID is for AMODE(64) callers, allows parameters in 64 bit addressable storage, and can be called specifying either a UR token or a UR interest token.

Code your resource manager to call the version that includes the support you need.

Only global URs can have work identifiers. Global work identifiers are not assigned to URs that are in local transaction mode. If you call Retrieve_Work_Identifier and specify a UR that is in local transaction mode, no work identifier is returned.

Your resource manager can request the current or next UWID for this UR. Table 44 describes the UWIDs that can be requested.

Table 44. Unit of Work Identifiers

Unit of Work Identifier (UWID)	**Format**	**Generate option is allowed or required**	**Next UWID is available**
LU 6.2 logical unit of work identifier (LUWID)	`netid.luname.instnum.seqnum` **netid.luname** 1-17 character identifier of the network and LU, preceded by a 1-byte fixed length field **instnum** 6-byte fixed TP instance **seqnum** 2-byte fixed sequence number	Allowed only for authorized callers.	Yes
Enterprise Identifier (EID)	`tidgtid` **tid** 4-byte transaction identifier (TID) **gtid** 8-40 byte global transaction identifier (GTID)	Not allowed	No
X/Open Identifier (XID)	`FormatIDGtrid_lengthBqual_lengthID` **FormatID** 4-byte fixed format ID **Gtrid_length** 4-byte fixed GTRID length **Bqual_length** 4-byte fixed BQUAL length **ID** 128-byte character XID The GTRID length and BQUAL length define the length of the first and second subsection of the ID. The GTRID must have a length of at least 1 byte, however the BQUAL can have a length of 0. The length of the entire XID cannot exceed 140 bytes.	Required RRS automatically generates an XID whenever a request for an XID is made by an authorized caller against a UR which does not already have one.	No

If the requested LUWID or XID has not been set by a call to Set_Work_Identifier or generated by RRS, you can use Retrieve_Work_Identifier to generate the LUWID or XID. The service, however, cannot generate EIDs. This service will not generate LUWID or XID if invoked by an unauthorized caller.

An XID can be set for a UR by a call to Set_Work_Identifier or by a call to Express_UR_Interest. An XID is generated by RRS for a UR that does not already have an XID when it is first requested, or when the UR becomes part of a cascaded

UR family. All of the URs in a cascaded UR family will have the same FormatID and GTRID components in their XID. No generation of work identifiers will be performed for unauthorized callers.

All of the URs in a cascaded UR family will have the same FormatID and GTRID components in their XID.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state None if retrieval of current work identifier is requested; in this case no generation of work identifiers will be performed.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31 bit (ATRRWID, ATRRWID2) 64 bit (ATR4RWID)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Uses standard MVS linkage conventions

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The resource manager associated with the specified UR interest token must be in **restart** state or in **run** state.

The UR transaction mode cannot be local.

Note: The LUWID of a UR created by calling the Retain_Interest service is not available until the UR for which Retain_Interest was called has reached **in_completion** state. At that time, you can call Retrieve_Work_Identifier to retrieve the LUWID by specifying the current LUWID and the *new_ur_interest_token* returned on the call to Retain_Interest.

You cannot call this service to generate an LU 6.2 logical unit identifier (LUWID) when any of the following are true:

- The resource manager returned the ATRX_LATER_CONTINUE return code from an exit routine for this expression of interest.
- The UR state is **in_completion**.
- The caller is not authorized.

When the resource manager issues the call in SRB mode, the call must not specify binary zero for *ur_interest_token* or *ur_or_uri_token*.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the appropriate call as shown in the syntax diagrams. You must code the parameters in the CALL statement as shown.

CALL ATRRWID	(return_code ,ur_interest_token ,retrieve_option ,generate_option ,uwid_type ,uwid_buffer_len ,uwid_len ,uwid_data_buffer)

CALL ATRRWID2	(return_code ,ur_or_uri_token ,retrieve_option ,generate_option ,uwid_type ,uwid_buffer_len ,uwid_len ,uwid_data_buffer)

CALL ATR4RWID	(return_code ,ur_or_uri_token ,retrieve_option ,generate_option ,uwid_type ,uwid_buffer_len ,uwid_len ,uwid_data_buffer)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Retrieve_Work_Identifier service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRRWID callers, specifies a token that uniquely identifies your resource manager's interest in the UR whose data you want to retrieve. Your resource

manager received the token from one of the following services: Express_UR_Interest, Retrieve_UR_Interest, Retain_Interest.

,ur_or_uri_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRRWID2 callers, specifies a token that uniquely identifies either the UR, or your resource manager's interest in the UR, whose data you want to retrieve:

- 0: Binary zero specifies the current UR associated with the application task.
- UR token: The token for the UR.
- UR interest token: The UR interest token that identifies your resource manager's interest in the UR.

Your resource manager received the token from one of the following services: Express_UR_Interest, Retrieve_Interest_Data, Retain_Interest, Create_Cascaded_UR, or Retrieve_UR_Data.

You must specify a URITOKEN when you specify ATR_GENERATE on *generate_option* and ATR_LUWID on *uwid_type*.

Because you may pass two different types of tokens through this parameter, passing an invalid token can generate either a ATR_URI_TOKEN_INV or a ATR_UR_TOKEN_INV return code. For example, passing an invalid UR token might result in an ATR_URI_TOKEN_INV return code. Even though a UR token was passed, if it is invalid, then RRS may not understand what sort of token it was supposed to be. For this reason, IBM recommends callers check both return codes, even when they know what type of token they intend to pass.

,retrieve_option
Supplied parameter

- Type: Integer
- Length: 4 bytes

Indicates whether you want to retrieve the current UWID or the next UWID. Unauthorized callers must specify the ATR_CURRENT retrieve option.

Constant in: Hexadecimal (Decimal) Equate Symbol	**UWID to be retrieved**
0 (0) ATR_CURRENT	**Current UWID**: RRS is to retrieve the current UWID for this UR.
1 (1) ATR_NEXT	**Next UWID**: RRS is to retrieve the next UWID for this UR. **Note:** You can validly specify ATR_NEXT only when *uwid_type* is ATR_LUWID.

,generate_option
Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the action RRS is to take if a UWID has not been set by a call to the Set_Work_Identifier service or generated by RRS. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	**RRS's action**
0 (0) ATR_DO_NOT_GENERATE	**Do not generate**: RRS is not to generate a new UWID.
1 (1) ATR_GENERATE	**Generate**: RRS is to generate a new UWID if the identifier has not yet been set or generated.

You must specify ATR_GENERATE when uwid_type is ATR_XID. You must specify ATR_DO_NOT_GENERATE when uwid_type is ATR_EID. You may specify either ATR_GENERATE or ATR_DO_NOT_GENERATE when uwid_type is ATR_LUWID. Unauthorized callers must specify the ATR_DO_NOT_GENERATE option.

You can specify ATR_GENERATE to generate the current LUWID or the next LUWID.

RRS will generate the current LUWID only when all of the following are true:

- You specified ATR_CURRENT on *retrieve_option*,
- The call to the Set_Exit_Information service for RRS specified an LU name in *variable_data_1*.

RRS will generate the next LUWID only when the following are true:

- You specified ATR_CURRENT on *retrieve_option*,
- At least one of the following is true:
  - Your resource manager's call to the Set_Exit_Information service for RRS specified an LU name in *variable_data_1*.
  - The current LUWID has already been set or generated.

,uwid_type

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the type of the UWID you want to retrieve. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	UWID type
0 (0) ATR_LUWID	An LU 6.2 logical unit of work identifier (LUWID)
1 (1) ATR_EID	An Enterprise identifier (EID)
2 (2) ATR_XID	An X/Open transaction identifier (XID)

,uwid_buffer_len

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the length of the buffer that your resource manager supplies for the UWID. The recommended length of the buffer is the maximum length for the UWID you are retrieving. Specify one of the following:

Maximum Length in: Hexadecimal (Decimal) Equate Symbol	UWID type
A (10) ATR_MIN_LUWID_LENGTH	The minimum length of a LU 6.2 LUWID
1A (26) ATR_MAX_LUWID_LENGTH	The maximum length of a LU 6.2 LUWID
C (12) ATR_MIN_EID_LENGTH	The minimum length of an Enterprise identifier
2C (44) ATR_MAX_EID_LENGTH	The maximum length of an Enterprise identifier
D (13) ATR_MIN_XID_LENGTH	The minimum length of an X/Open identifier

Maximum Length in: Hexadecimal (Decimal) Equate Symbol	UWID type
8C (140) ATR_MAX_XID_LENGTH	The maximum length of an X/Open identifier

,uwid_len
Returned parameter

- Type: Integer
- Length: 4 bytes

Receives the actual hexadecimal length of the UWID from the service.

,uwid_data_buffer
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified in *uwid_buffer_len*

Specifies a buffer that receives the UWID from the service. The *uwid_buffer_len* parameter specifies the length of the buffer. The UWID length is from 10 to 140 bytes.

The format of the UWID returned depends on the UWID type. A LUWID has the following format:

```
netid.luname.instnum.seqnum
```

The fields are as follows:

netid.luname
1-17 character identifier of the network and LU, preceded by a 1-byte length field

instnum
6-byte TP instance

seqnum
2-byte sequence number

An EID has the following format:

```
tidgtid
```

The fields are as follows:

tid 4-byte transaction identifier (TID)

gtid 8-40 byte global transaction identifier (GTID)

For XID, the *uwid_data_buffer* contains the 4–byte address of the buffer to contain the retrieved XID. An XID has the following format:

```
FormatIDGtrid_lengthBqual_lengthID
```

The fields are as follows:

FormatID
4-byte fixed format ID

Gtrid_length
4-byte fixed Gtrid length

Bqual_length
4-byte fixed Bqual length

ID 128-byte character XID

The 1–128 byte ID field has the following format:

Gtrid 1–64 byte Gtrid

Bqual 0–64 byte Bqual

The length of the entire XID cannot exceed 140 bytes.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'000E0000' or X'000E0001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
7 ATR_REQUESTED_WID_UNAVAILABLE	**Meaning**: Program processing. The requested UWID has not been set by a call to the Set_Work_Identifier service or been generated by RRS. The *generate_option* parameter specified that a new UWID should not be generated. The system accepts the call. **Action**: If this result is expected, no action is needed. If this result is not expected, check the resource manager for a probable coding error; correct the resource manager and rerun it.
A ATR_PARTIAL_UWID_DATA	**Meaning**: Program error. The *uwid_buffer_len* value specified in the call is less than the actual length of the UWID. The system accepts the service call. RRS puts in the buffer as many characters of the data as will fit, starting at the left, and returns the actual length in uwid_len. **Action**: No action is required. If the result is not expected, check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call does not identify one of the currently valid interests. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a UR token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
37E ATR_RETRIEVE_OPTION_INV	**Meaning**: Program error. The retrieve option specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
380 ATR_UWID_TYPE_INV	**Meaning**: Program error. The specified uwid_type is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
382 ATR_UWID_BUF_LEN_INV	**Meaning**: Program error. The length specified for the UWID data buffer is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
388 ATR_GENERATE_OPTION_INV	**Meaning**: Program error. The *generate_option* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
3A3 ATR_UR_TOKEN_INV	**Meaning**: Program error. The UR token specified in the call does not identify a valid UR. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a URI token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
3B4 (948) ATR_AUTH_FAILURE_RETRIEVE_OPTION	**Meaning**: The caller, which is PKM 8-15 problem state, specified a retrieve_option not equal to ATR_CURRNET. Only ATR_CURRENT can be specified when the caller is PKM 8-15 problem state. To use other retrieve_option's the caller must be PKM allowing key 0-7, or supervisor state. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3B5 (949) ATR_AUTH_FAILURE_GENERATE_ OPTION	**Meaning**: The caller, which is PKM 8-15 problem state, specified a generate_option not equal to ATR_DO_NOT_GENERATE. Only ATR_DO_NOT_GENERATE can be specified when the caller is PKM 8-15 problem state. To use other retrieve_option's the caller must be PKM allowing key 0-7, or supervisor state. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager must be in **restart** state or in **run** state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
73C ATR_AFTER_NEW_UR	**Meaning**: The resource manager has previously returned the ATRX_LATER_CONTINUE return code from an exit routine for this expression of interest. You cannot generate an LUWID after that point for this expression of interest. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
73F ATR_LUWID_NOT_AVAILABLE	**Meaning**: Environmental error. The current LUWID for the UR specified in the call is not available. The LUWID created for a UR by a Retain_Interest call is not available until the previous UR has reached a **complete** state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
748 ATR_GEN_NOT_ALLOWED_NO_LUNAME	**Meaning**: The resource manager did not specify an LU prefix on a Set_Exit_Information call, so the resource manager cannot tell RRS to generate a LUWID. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
74D ATR_GEN_NOT_ALLOWED_EID	**Meaning**: The request to generate an Enterprise identifier is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
751 ATR_GEN_REQUIRED_XID	**Meaning**: Program error. The request to not generate an XID is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
753 ATR_GEN_NOT_ALLOWED_ NO_URI_TOKEN	**Meaning**: Program error. The request to generate a LUWID is not valid, because a URI token was not specified. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
754 ATR_RETRIEVE_NEXT_EID_INV	**Meaning**: Program error. The request to retrieve the next Enterprise identifier is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
755 ATR_RETRIEVE_NEXT_XID_INV	**Meaning**: Program error. Retrieval of XID for the next Unit of Recovery is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
765 ATR_GEN_LUWID_NOT _ALLOWED_LOCAL	**Meaning**: Program error. The request to generate a LUWID is not valid for a UR in local transaction mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. If the caller is a resource manager, it should not unset its exits with RRS.
767 ATR_GEN_XID_NOT _ALLOWED_LOCAL	**Meaning**: Program error. The request to generate an XID is not valid for a UR in local transaction mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. If the caller is a resource manager, it should not unset its exits with RRS.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Retrieve_Work_Identifier was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to retrieve the current LU 6.2 LUWID. If a UWID is not available, RRS will generate one.

```
⋮
URI_TOKEN = UR_INTEREST_TOKEN
RET_OPT = ATR_CURRENT
GEN_OPT = ATR_GENERATE
UWID_TYPE = ATR_LUWID
BUF_LEN = ATR_MAX_LUWID_LENGTH
CALL ATRRWID2(RC,URI_TOKEN,RET_OPT,GEN_OPT,UWID_TYPE,
     BUF_LEN,LUWID_LEN,LUWID)
IF RC ≠ ATR_OK THEN
```

```
  /* Handle error */
⋮
```

Set_Environment (ATRSENV, ATR4SENV)

- ATRSENV is for AMODE(31) callers.
- ATR4SENV is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A work manager calls the Set_Environment service to establish environmental settings for RRS, but RRS need not be available when the service is called. Typically, a work manager calls Set_Environment to establish transaction default modes for the primary address space scope and context-specific scopes. The work manager can also specify that the settings it chooses are to be protected against change by unauthorized programs. The work manager can direct RRS to establish the default transaction mode and the default two-phase commit action.

Calling any of the following services will cause a UR state to go from **in-reset** to **in-flight**, setting a transaction mode for that UR:

- Create_Cascaded_UR
- Express_UR_Interest
- Retrieve_Work_Identifier
- Set_Side_Information
- Set_Work_Identifier

Note that calls to Create_Cascaded_UR, Retrieve_Work_Identifier, or Set_Work_Identifier will never cause a UR to have a local transaction mode.

Establish the default transaction mode: The default transaction mode determines the transaction mode for an **in-reset** UR when the first expression of interest occurs. The first expression of interest establishes the transaction mode for the UR; the transaction mode cannot change for the life of the UR. The possible transaction modes are:

- **ATR_LOCAL_MODE**: Sets local transaction mode as the default for implicitly-started transactions
- **ATR_GLOBAL_MODE**: Sets global transaction mode as the default for implicitly-started transactions
- **ATR_HYBRID_GLOBAL_MODE**: Sets hybrid-global transaction mode as the default for implicitly-started transactions. Hybrid-global mode is the same as global except that it allows resource managers to exhibit proprietary transactional behavior. Hybrid-global is the default transaction mode in the absence of an applicable environment setting for a given UR. A resource manager that does not have any proprietary behaviors can treat hybrid-global transaction mode as global.
- **ATR_NOT_SET**: Removes the default transaction mode previously set for this scope (address space or context). The result of ATR_NOT_SET is the same as if Set_Environment for this scope had never been issued.

Note that applications cannot use Begin_Transaction to explicitly start a new transaction when the transaction mode environment is either not set (ATR_NOT_SET) or set to hybrid-global (ATR_HYBRID_GLOBAL_MODE).

Because the work manager can establish environmental settings for both address space scope and context scope, conflicting values might apply to a given UR. RRS uses the following precedence list to resolve conflicting settings:

1. Context scope transaction default (protected or not)
2. Home address space scope default (protected or not)
3. RRS system default

Note: It is important to distinguish that, for address space scope specifications when zero is specified for *stoken*, the Set_Environment service affects the primary address space, while the Retrieve_Environment service relates to the home address space.

Establish default two-phase commit actions: The work manager can provide a separate direction to RRS to commit or roll back URs when the UR state is still **in-flight** for the following event:

- **ATR_NORM_CTX_END_SETTING** RRS is to take a prescribed action when a UR's associated context goes through normal termination. The default action is to commit the UR.

Environment

The requirements for the caller are:

Minimum authorization:	Any
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRSENV) 64 bit (ATR4SENV)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The following restrictions are placed on unauthorized (PKM 8–15 problem state) callers. These callers:

- Must specify ATR_UNPROTECTED_SETTING for the *protection_level* parameter.
- Cannot change any environmental settings established (by an authorized caller) with ATR_PROTECTED_SETTING for each element of the *environment_protection* parameter.

- Can change an address space scope setting only when that space is the primary address space (the *stoken* parameter is 0) and the setting is unprotected.
- Can change a context scope setting only when:
  - The context is current, the setting is unprotected, and the home address space scope setting is unprotected.
  - The context is not current, the setting is unprotected, the context is owned by a key 8–15 problem state work manager registered in the home address space, and the home address space scope setting is unprotected.

SRB mode callers cannot specify a context token of 0 when trying to establish environment settings at the context scope (ATR_CONTEXT_SCOPE).

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRSENV	(return_code ,diag_area ,scope ,context_token ,stoken ,element_count ,environment_id ,environment_value ,environment_protection)

CALL ATR4SENV	(return_code ,diag_area ,scope ,context_token ,stoken ,element_count ,environment_id ,environment_value ,environment_protection)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Set_Environment service.

,diag_area
Returned parameter

- Type: Character string
- Character Set: No restriction
- Length: 32 bytes

Contains diagnostic data from Set_Environment to help IBM Service determine the cause of a Set_Environment failure. Be sure to log this data when recording any information about a Set_Environment failure.

,scope
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies the scope of the environmental setting value(s), either address space scope or context scope, as follows:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
1 (1) ATR_ADDRESS_SPACE_SCOPE	The environmental setting has address space scope for the address space represented by the token in the *stoken* parameter.
2 (2) ATR_CONTEXT_SCOPE	The environmental setting applies for the context represented by the token in the *context_token* parameter.

,context_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token of the context for which the resource manager is establishing context scope environment settings:

- 0: Binary zeros specify either:
  - The current context (when *scope* is ATR_CONTEXT_SCOPE)
  - No context (when *scope* is ATR_ADDRESS_SPACE_SCOPE)

  If *scope* is ATR_ADDRESS_SPACE_SCOPE, then *context_token* must be 0.
- token: Specifies a valid context token.

,stoken
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 8 bytes

Specifies the space token (stoken) of the address space for which the resource manager is establishing address space scope environment settings:

- 0: Binary zeros indicate the primary address space (required for unauthorized callers). If *scope* is ATR_CONTEXT_SCOPE, then *stoken* must be 0.
- token: Specifies a valid address space token.

Note: If an unauthorized caller in cross-memory mode specifies 0, the system rejects the call.

,element_count
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies the number of elements in the environment-setting array, which consists of the *environment_id*, *environment_value*, and *environment_protection* parameters.

The maximum number of elements is the number of possible environment settings (transaction mode and two-phase commit action) times the number of environment-setting parameters. The maximum number is 2.

,environment_id

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Specifies one or more identifiers; each identifier supplies an environment attribute that is to be set. When you specify more than one identifier, you must define an array; *element_count* indicates the number of elements in the array. The positions of the identifiers in this array define the positions of the environment attributes in the environment-setting array. The *scope* parameter specifies the scope at which these settings are to apply. Specify each identifier as one of the following:

Identifier in: Hexadecimal (Decimal) Equate Symbol	**Description**
1 (1) ATR_TRAN_MODE_SETTING	The service is to set the transaction mode.
2 (2) ATR_NORM_CTX_END_SETTING	The service is to set the two-phase commit action RRS should take for **in-flight** URs when the associated context goes through normal end processing. **Note:** This setting does not apply to a cascaded (child) UR. The outcome of the entire syncpoint tree is determined by the environmental setting of only the top-level UR, not any of the child URs.

,environment_value

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Supplies a value for each identifier on the *environment_id* parameter. When you specify more than one identifier, you must define an array, where *element_count* indicates the number of elements in the array. The positions of the identifiers in the *environment_id* array define the positions of the environment attributes in the *environment_value* array.

Specify the value for ATR_TRAN_MODE_SETTING as one of the following:

Value in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) ATR_NOT_SET	The transaction mode environment setting for this *environment_id* (address space or context) is to be removed. Setting this value makes it as if Set_Environment (for the *environment_id* specified) was never issued.
1 (1) ATR_GLOBAL_MODE	The transaction mode is set to global for the requested scope.
2 (2) ATR_LOCAL_MODE	The transaction mode is set to local for the requested scope.
3 (3) ATR_HYBRID_GLOBAL_MODE	The transaction mode is set to hybrid-global for the requested scope. This is the same as global mode, except it allows the resource manager to exhibit proprietary connection behavior.

Specify the value for ATR_NORM_CTX_END_SETTING as one of the following:

Value in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) ATR_NOT_SET	The environment setting at this *environment_id* (address space or context) is removed. Setting this value makes it as if Set_Environment (for the *environment_id* specified) was never issued.
1 (1) ATR_COMMIT_ACTION	RRS is to commit **in-flight** URs.
2 (2) ATR_ROLLBACK_ACTION	RRS is to roll back **in-flight** URs.

,environment_protection

Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Supplies a protection value for each identifier in the *environment_id* parameter. When you specify more than one identifier, you must define an array, where *element_count* indicates the number of elements in the array. The positions of

the identifiers in the *environment_id* array define the positions of the environment attributes in the *environment_protection* array.

Only an authorized caller can set the protection value. By setting ATR_PROTECTED_SETTING, authorized callers can prevent unauthorized callers from changing the environmental settings. Specify each element as one of the following:

Value in: Hexadecimal (Decimal) Equate Symbol	**Description**
1 (1) ATR_UNPROTECTED_SETTING	An unauthorized caller can change the settings specified in the corresponding element in the *environment_value* array.
2 (2) ATR_PROTECTED_SETTING	Only an authorized caller can change the settings specified in the corresponding element in the *environment_value* array.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00260000' or X'00260001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. The environment has been set as desired. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
104 ATR_MODE_INV	**Meaning**: Program error. The calling program is not in task mode, specified a zero context token, and attempted to set environment settings at a scope of ATR_CONTEXT_SCOPE. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The application is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the application for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that is running a version of RRS that supports this service call. Then rerun the resource manager.
361 ATR_CONTEXT_TOKEN_INV	**Meaning**: Program error. The context token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
362 ATR_STOKEN_INV	**Meaning**: Program error. The address space token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
364 ATR_ENV_SETTING_ID_INV	**Meaning**: Program error. A value in the *environment_id* parameter specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
365 ATR_ENV_SETTING_INV	**Meaning**: Program error. A value in the *environment_value* parameter specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
366 ATR_SCOPE_INV	**Meaning**: Program error. The scope specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
36B ATR_ACTION_INV	**Meaning**: Program error. The value specified for ATR_NORM_CTX_END_SETTING in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
36C ATR_PROTLEVEL_INV	**Meaning**: Program error. The environment setting protection value in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
392 ATR_ELEMENT_COUNT_INV	**Meaning**: Program error. The element count value in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
3AB ATR_AUTH_FAILURE	**Meaning**: Program error. An unauthorized caller tried to change a setting for an authorized context other than the current context. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
801 ATR_SETTING_PROTECTED	**Meaning**: Program error. An unauthorized caller tried to change a setting that was protected by an authorized caller. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
802 ATR_STOKEN_NOT_ZERO	**Meaning**: Program error. The stoken parameter was incorrectly specified. One of the following is true: • The stoken is not zero, but the caller specified ATR_CONTEXT_SCOPE on the scope parameter; or, • The stoken is not zero and the caller is unauthorized. In either case, the system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
803 ATR_CTOKEN_NOT_ZERO	**Meaning**: Program error. The context token parameter was incorrectly specified. The caller specified ATR_ADDRESS_SPACE_SCOPE on the scope parameter and a non-zero value on the context token parameter. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the work manager issues a call to establish environmental settings for RRS.

```
⋮
SCOPE = ATR_CONTEXT_SCOPE
C_TOKEN = MY_CONTEXT_TOKEN
A_TOKEN = 0
ELE_CNT = 1
ENV_SET_ID = ATR_NORM_CTX_END_SETTING
ENV_SET = ATR_COMMIT_ACTION
ENV_SET_PROT = ATR_PROTECTED_SETTING
CALL ATRSENV(RC,DIAG_DATA,SCOPE,C_TOKEN,A_TOKEN,ELE_CNT,
            ENV_SET_ID,ENV_SET,ENV_SET_PROT)
⋮
```

Set_Log_Name (ATRISLN, ATR4ISLN)

- ATRISLN is for AMODE(31) callers.

- ATR4ISLN is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Set_Log_Name service to give its log name to RRS. Note that this name is not necessarily the name of an actual log. It is a value that your resource manager uses at restart to synchronize processing with RRS.

RRS hardens the resource manager log name in the RRS log. The next time your resource manager restarts, it can call the Retrieve_Log_Name service to retrieve the name.

In response to the call, RRS returns a return code.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRISLN) 64 bit (ATR4ISLN)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The resource manager associated with the specified resource manager token must be in one of the following states:

- **Set**, which means it has registered and set its exit routines with RRS
- **Restart**, which means it has registered, set its exit routines with RRS, and begun restart
- **Run**, which means it has registered, set its exit routines with RRS, and completed restart

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRISLN	(return_code ,resource_manager_token ,rm_logname_len ,rm_logname)

CALL ATR4ISLN	(return_code ,resource_manager_token ,rm_logname_len ,rm_logname)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Set_Log_Name service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

,rm_logname_len
Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies, in hexadecimal, the length of the resource manager's log name. The length can be from X'1' to X'40' (1 - 64) bytes.

,rm_logname
Supplied parameter

- Type: Character string
- Character Set: See description
- Length: Specified in *rm_logname_len*

Specifies the resource manager's log name. The log name can consist of:

- Alphanumeric characters: A-Z and 0-9.
- National characters: $ (X'5B'), # (X'7B'), @ (X'7C').
- The period (.).
- The underscore(_).
- Trailing blank characters. The name may not start with a blank or contain embedded blanks.

The field is not case-sensitive. If you specify lower-case letters, RRS converts them to upper case.

Use the following conventions to avoid name conflicts:

- IBM-provided resource managers use A-C or G-I as the first character and `.IBM` as the ending qualifier.
- Other resource managers should begin the name with D-F or J-Z and end the name with a period and the company name or acronym.

For example:

```
RMLOG.VENDORCORP
RESMANAGERLOG.GROWTHCOMPANY
```

Note: The resource manager log name is preserved across restarts of the system, restarts of RRS, and restarts of the resource manager.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00080000' or X'00080001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
301 ATR_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
37A ATR_RM_LOGNAME_INV	**Meaning**: Program error. The name for the resource manager log specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
37B ATR_RM_LOGNAME_LEN_INV	**Meaning**: Program error. The length of the resource manager log name specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the resource manager token specified in the call is not in a valid state to issue the service call. The resource manager state must be **set**, **restart**, or **run**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to give its log name to RRS.

```
⋮
RM_TOKEN = MY_RM_TOKEN
LOGN_LEN = MY_LOGNAME_LEN
LOGNAME = MY_LOGNAME
CALL ATRISLN(RC,RM_TOKEN,LOGN_LEN,LOGNAME)
IF RC=0 THEN
⋮
```

Set_Persistent_Interest_Data (ATRSPID, ATR4SPID)

- ATRSPID is for AMODE(31) callers.
- ATR4SPID is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Set_Persistent_Interest_Data service to provide persistent interest data for a protected interest in a unit of recovery (UR). In response to the call, RRS returns a return code.

Note: Set_Persistent_Interest_Data can be used for an interest in a UR in local transaction mode, but the data will not be logged.

The call can also be used to delete existing persistent interest data.

Persistent Interest Data: When it hardens information for the interest in an RRS log, RRS records the persistent interest data. Because the data is hardened, it will be available if your resource manager restarts or if RRS restarts, forcing your resource manager to restart.

Your resource manager can also provide persistent interest data in a call to the following services: Express_UR_Interest, Change_Interest_Type, or Retain_Interest. Your resource manager can retrieve persistent interest data in a call to the Retrieve_UR_Interest service or the Retrieve_UR_Data service.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRSPID) 64 bit (ATR4SPID)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	MVS standard linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The state of the resource manager associated with the specified UR interest token must be **run**, which means it has registered, set its exit routines with RRS, and completed restart.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRSPID	(return_code ,ur_interest_token ,persistent_interest_data_length ,persistent_interest_data)

CALL ATR4SPID	(return_code ,ur_interest_token ,persistent_interest_data_length ,persistent_interest_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Set_Persistent_Interest_Data service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that identifies your resource manager's interest in the UR. Your resource manager received the token from the Express_UR_Interest service or the Retain_Interest service.

,persistent_interest_data_length
Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies, in hexadecimal, the length of the persistent interest data. The length can be from X'0' to X'1000' (0 - 4096) bytes. The maximum amount of data that can be logged for a particular UR is 60K (61440) bytes, which includes the persistent data and all other data that RRS must log for the UR.

If the call specifies a length of zero, RRS deletes the persistent interest data for the interest.

,persistent_interest_data
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified in *persistent_interest_data_length*

Specifies the persistent interest data you want to set for your resource manager's interest.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00100000' or X'00100001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call is not one of the currently valid interests. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
376 ATR_PERSISTENT_DATA_LEN_INV	**Meaning**: Program error. The length specified in the *persistent_interest_data_len* parameter in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager state must be **run**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
730 ATR_NOT_PROTECTED_INTEREST	**Meaning**: Program error. The UR interest token does not represent a protected interest. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
749 ATR_MAX_UR_LOG_DATA_ EXCEEDED	**Meaning**: Environmental error. This request will exceed the maximum amount of data that RRS can log for a UR. The system rejects the service call. **Action**: Fail the client program request or back out the UR. Verify that the space set up for logging is adequate.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Set_Persistent_Interest_Data was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to provide persistent interest data.

```
⋮
URI_TOKEN = MY_URI_TOKEN
P_DATA_LEN = LENGTH(MY_P_DATA)
P_DATA = MY_P_DATA
CALL ATRSPID(RC,URI_TOKEN,P_DATA_LEN,P_DATA)
IF RC ≠ ATR_OK THEN
   /* Handle error */
⋮
```

Set_Post_Sync_PET (ATRSPSP2, ATR4SPSP)

- ATRSPSP2 is for AMODE(31) callers.

- ATR4SPSP is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A work manager calls the Set_Post_Sync_PET service to enable a program to know when a syncpoint completes and receive information about that syncpoint without actually expressing interest in a UR or work context. The caller provides a pause element token (PET) to be associated with a target UR. RRS will release the pause element designated by the specified PET when the context the UR is associated with can either be reused for a new UR or ended without RRS holding up the process. This occurs when the UR reaches the **forgotten** state, except under the following circumstances:

- When the UR syncpoint is under the control of an SDSRM and circumstances require the SDSRM to issue a Forget_Agent_UR_Interest call before the UR can reach the **forgotten** state. In this case, RRS will release the PET when the UR reaches the **in-forget** state.
- When a COMPLETION exit returned the ATRX_LATER_CONTINUE return code and RRS is going to give control back to the application prior to transitioning the UR to the **forgotten** state. In this case, RRS will release the PET when the UR is in the **in-completion** state when all exits finished or returned ATRX_LATER_CONTINUE.
- When RRS fails. RRS will ensure that the PET is released with an appropriate release code. The state of the UR cannot be determined.

When RRS issues a release, it will specify a release code that contains information about the results of the UR.

If you create cascaded transactions, you can also use the Set_Post_Sync_PET service to determine when a cascaded transaction has completed.

Note: RRS will consider each PET associated with a UR as the equivalent of an expression of interest when queried by the Retrieve_Interest_Count service.

Because setting a post-syncpoint PET does not cause the UR state to change from **in-reset** to **in-flight**, it is possible for the transaction mode to change after the PET is associated with a UR. The PET is released when the UR completes, regardless of the transaction mode. In addition to normal syncpoint completion, a UR in local transaction mode is considered complete when all resource manager interests in it are deleted. The release code provides an indication of the transaction mode at the time RRS releases the PET.

Environment

The requirements for the caller are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRSPSP2) 64 bit (ATR4SPSP)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	MVS standard linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

For the call, the UR state must be **in-reset** or **in-flight**.

When the resource manager issues the call in SRB mode, the call cannot specify a *ur_token* of 0, indicating information for the current UR.

A PKM 8–15 problem state caller must specify a PET for a Pause Element allocated with *auth_level*=IEA_UNAUTHORIZED.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRSPSP2	(return_code ,UR_token ,pause_element_token)

CALL ATR4SPSP	(return_code ,UR_token ,pause_element_token)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Set_Post_Sync_PET service.

,UR_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the token of the UR to which the PET specified by *pause_element_token* is to be associated:

- 0: Binary zero specifies the current UR associated with the application's task.
- *token*: The UR token of a particular UR.

,pause_element_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the pause element token to be associated with the UR specified by the *UR_token*. When the UR reaches the **forgotten** state (or one of the other conditions specified in the description of this service call), RRS will release the pause element specified by the *pause_element_token*.

Pause element tokens are not preserved across system or RRS failures. Any PETs that are associated with URs via the Set_Post_Sync_PET service will be released if RRS terminates.

When the pause element is released, RRS will place a set of flags in the release code. The flags are:

Bit Flag name	Meaning, if Bit is On
0 ATRRCODENORRS	RRS will always set this bit to 0. An application can release the pause element with this bit set to indicate to the resource manager that RRS did not release it.
1 ATRRCODERRSFAILED	RRS terminated. Bits 2–23 have undefined contents.
2–8	Reserved.
9 ATRRCODETERMINATINGSYNCPOINT	The context is ending. RRS issued an implicit commit or backout for the UR. There cannot be any more new URs for this context.
10 ATRRCODERESOLVEDBYINSTALLATION	The installation used an RRS panel to commit or back out the UR, which had been in an **in-doubt** state.
11 ATRRCODEHEURISTICMIXED	RRS detected a heuristic-mixed condition for this UR.
12 ATRRCODERESYNCINPROGRESS	RRS detected a resync in progress for this UR.
13 ATRRCODEPREPARERESULTFORGET	The collected prepare vote was forget.
14 ATRRCODEIMMEDIATEBACKOUT	Backout was requested by the application.
15	Reserved.
16 ATRRCODECOMMIT	If set, the overall vote for the UR is commit. If not set, and ATRRCODEPREPARERESULTFORGET is not set, the overall vote for the UR is backout.
17–18	Reserved.
19 ATRRCODECASCADEDUR	The UR is a cascaded UR.
20 ATRRCODELOCALMODE	If set, the UR was in local transaction mode. If this bit is set, then ATRRCODEGLOBALMODE cannot also be set. If neither is set, then the UR was in hybrid-global transaction mode (ATRRCODEHYBRIDGLOBALMODE).

Bit Flag name	Meaning, if Bit is On
21 ATRRCODEGLOBALMODE	If set, the UR was in global transaction mode. If this bit is set, then ATRRCODELOCALMODE cannot also be set. If neither is set, then the UR was in hybrid-global transaction mode (ATRRCODEHYBRIDGLOBALMODE).
22–23	Reserved.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00200000' or X'00200001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The calling program is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
104 ATR_MODE_INV	**Meaning**: Program error. The calling program specified 0 in *UR_token*, indicating the context associated with the current UR, but the calling program is not in task mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The calling program is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the calling program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that is running a version of RRS that supports this service call. Then rerun the resource manager.
3A3 ATR_UR_TOKEN_INV	**Meaning**: Program error. The UR token specified in the *UR_token* parameter is not valid. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A6 ATR_PET_INV	**Meaning**: Program error. The pause element token specified in the *pause_element_token* parameter is not valid. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A7 ATR_PET_OUTDATED	**Meaning**: Program error. The pause element token specified in the *pause_element_token* parameter is outdated. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A8 ATR_PET_AUTH_FAILURE	**Meaning**: Program error. The pause element token specified in the *pause_element_token* parameter was allocated with *auth_level*=IEA_AUTHORIZED, and the caller is unauthorized. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
3A9 ATR_PET_SPACE_FAILURE	**Meaning**: Program error. The pause element token specified in the *pause_element_token* parameter represents a pause element belonging to another address space, and the caller is unauthorized. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The UR specified by the *UR_token* is not in a valid state for the service call. The UR must be in an **in-reset** or **in-flight** state. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Set_Post_Sync_PET was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the calling program attempts to associate a UR a pause element token. Storage for the call parameters has already been allocated.

```
⋮
CALL ATRSPSP2(RC, URTOKEN, PETOKEN)
⋮
```

Set_RM_Metadata (ATRSDTA, ATR4SDTA)

- ATRSDTA is for AMODE(31) callers.
- ATR4SDTA is for AMODE(64) callers, and allows parameters in 64 bit addressable storage.

A resource manager calls the Set_RM Metadata service to give up to 8K (8192) bytes of data to RRS. RRS will harden the data.

Environment

The requirements for the resource manager are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRSDTA) 64 bit (ATR4SDTA)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from *SYS1.CSSLIB*, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The resource manager associated with the specified resource manager token must be in Run state, which means it has been registered, set its exit routines with RRS, and completed restart.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRSDTA	(return_code ,resource_manager_token ,rm_metadata_len ,rm_metadata)

CALL ATR4SDTA	(return_code ,resource_manager_token ,rm_metadata_len ,rm_metadata)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Set_RM_Metadata service.

,resource_manager_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the resource manager token that identifies the resource manager. Your resource manager received the token from the Register_Resource_Manager service.

,rm_metadata_len
Supplied parameter

- Type: Integer
- Character Set: N/A
- Length: 4 bytes

Length of the data specified by the *rm_metadata* keyword. The maximum amount of metadata that can be given to RRS to store is 8192 bytes. If a length of zero is specified, RRS deletes the resource manager's metadata from the log stream.

,rm_metadata
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified in *rm_metadata_len*

Specifies the resource manager's metadata you want RRS to store.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00280000' or X'00280001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: **Hexadecimal** **Equate Symbol**	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports registration services. Then rerun the resource manager.
301 ATR_RM_TOKEN_INV	**Meaning**: Program error. The resource manager token specified in the call is incorrect. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
38A ATR_RM_METADATA_LEN_INV	**Meaning**: Program error. The length of the resource manager metadata specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
38C ATR_RM_METADATA_LOG_ UNAVAILABLE	**Meaning**: The MetaData callable service failed since the resource manager MetaData log stream is not available. **Action**: Check SYSLOG for messages ATR132I or ATR172E that will further explain why the log is unavailable.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
38D ATR_RM_8K_METADATA_NOT_ ALLOWED	**Meaning**: The resource manager did not set the *ATR_8K_RM_METADATA_REQUESTED* flag on *Set Exit Information (CRGSEIF, CRGSEIF1,and CRG4SEIF)* so the resource manager cannot set or retrieve 8K Meta Data. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
38E ATR_RM_METADATA_MISSING_DATA	**Meaning**: When reading from the RM Meta Data log stream, records were encountered that indicate there was a loss of data or a gap in the log stream. If Meta Data was stored for the RM, it cannot be found. **Action**: Check SYSLOG for messages ATR202D and ATR212I that will further explain the error and how to correct it.
701 ATR_RM_STATE_ERROR	**Meaning**: The resource manager associated with the resource manager token specified in the call is not in a valid state to issue the service call. The resource manager state must be run. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to give its RM Metadata to RRS.

```
.
.
.
RM_TOKEN = MY_RM_TOKEN
META_LEN = MY_META_LEN
META = MY_META
CALL ATRSDTA(RC,RM_TOKEN,META_LEN,META)
IF RC=0 THEN
.
.
.
```

Set_Side_Information (ATRSUSI, ATRSUSI2, ATR4SUSI)

The resource manager calls the Set_Side_Information service to set side information for an interest in a unit of recovery (UR). In response to the call, RRS returns a return code. There are three versions of Set_Side_Information, each with different parameters.

- ATRSUSI is for AMODE(31) callers and is the basic version of the service. It must be called specifying a UR interest token.
- ATRSUSI2 is for AMODE(31) callers and can be called specifying either a UR token or a UR interest token.
- ATR4SUSI is for AMODE(64) callers, allows parameters in 64 bit addressable storage, and can be called specifying either a UR token or a UR interest token.

Code your resource manager to call the version that includes the support you need.

Side Information: The side information is set by RRS or, in a call to Set_Side_Information, by a resource manager that is interested in the UR. Much of the side information is set only by a resource manager that uses Systems Network Architecture (SNA) Logical Unit (LU) 6.2 sync point architecture. The side information, defined in the *side_info_id* parameter, indicates the following:

- Heuristic-mixed condition
- Backout required
- Break tree
- Backout of next UR
- Resync in progress
- New LUWID presentation header (PSH) unacceptable
- Invoke completion exits
- Application complete
- Reset application complete

Your resource manager can call the Retrieve_Side_Information service to retrieve side information.

Parameter Array: The *side_info_id* parameter is an input array; each position provides side information to RRS. The *element_count* parameter indicates the number of positions in the array.

Environment

The requirements for the caller are:

Minimum authorization:	None
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	
	31 bit (ATRSUSI, ATRSUSI2) 64 bit (ATR4SUSI)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The state of the resource manager associated with the specified UR interest token must be **run**, which means it has registered, set its exit routines with RRS, and completed restart.

For a UR in local transaction mode, the only *side_info_id* you can specify is ATR_DRIVE_COMPLETION.

If the resource manager returns ATRX_LATER_CONTINUE from its END_UR exit, then this service cannot be called to set ATR_BREAK_TREE at any time after that point for this expression of interest.

A caller that is PKM 8–15 problem state must specify a *UR_token* for a UR that is either:

- Associated with a DU native context associated with a task in the current home address space, or
- A private context owned by a PKM 8–15 problem state resource manager registered in the home address space.

A caller that is PKM 8–15 problem state can only specify ATR_APPL_COMPLETE or ATR_RESET_APPL_COMPLETE for *side_info_id*.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the appropriate call as shown in the syntax diagrams. You must code the parameters in the CALL statement as shown.

CALL ATRSUSI	(return_code ,ur_interest_token ,element_count ,side_info_id)

CALL ATRSUSI2	(return_code ,ur_or_uri_token ,element_count ,side_info_id)

CALL ATR4SUSI	(return_code ,ur_or_uri_token ,element_count ,side_info_id)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Character Set: N/A
- Length: 4 bytes

Contains the return code from the Set_Side_Information service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRSUSI callers, specifies a token that uniquely identifies your resource manager's interest in the UR whose side information you want to set. Your resource manager received the token from one of the following services: Express_UR_Interest, Retain_Interest.

,ur_or_uri_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRSUSI2 callers, specifies a token that uniquely identifies either the UR, or your resource manager's interest in the UR, whose side information you want to set:

- UR token: The token for the UR.
- UR interest token: The UR interest token that identifies your resource manager's interest in the UR.

Your resource manager received the token from one of the following services: Express_UR_Interest, Retrieve_Interest_Data, Retain_Interest, Create_Cascaded_UR, or Retrieve_UR_Data.

Because you may pass two different types of tokens through this parameter, passing an invalid token can generate either a ATR_URI_TOKEN_INV or a ATR_UR_TOKEN_INV return code. For example, passing an invalid UR token might result in an ATR_URI_TOKEN_INV return code. Even though a UR token was passed, if it is invalid, then RRS may not understand what sort of token it was supposed to be. For this reason, IBM recommends callers check both return codes, even when they know what type of token they intend to pass.

,element_count

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the number of elements in the array for the *side_info_id* parameter. Specify a hexadecimal value from X'1' to X'8'

,side_info_id

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies one or more identifiers that set the side information. When you specify two or more identifiers, place the identifiers in an array. For a UR in local transaction mode, the only identifier you can specify is ATR_DRIVE_COMPLETION. The possible identifiers are:

Constant in: Hexadecimal (Decimal) Equate Symbol	**Identifier**
0 (0) ATR_HEURISTIC_MIX	Heuristic-mixed condition A heuristic commit occurred while the UR state was **in_backout**, a heuristic reset occurred while the UR state was **in_commit**, or a resource manager of distributed resources informed RRS of a heuristic-mixed condition. When this information is initially set through a call to Set_Side_Information, RRS will harden (or reharden) the UR state immediately (before control returns to the caller), which will make the information available on restart.
1 (1) ATR_BACKOUT_REQUIRED	Backout required When this identifier is initially set, RRS will force the current UR to back out when a syncpoint occurs. If a UR state has passed beyond **in_prepare**, RRS ignores this identifier.
10 (16) ATR_BREAK_TREE	Break tree When this identifier is initially set, RRS will reset the logical unit of work identifier (LUWID) for the next UR.

Constant in: Hexadecimal (Decimal) Equate Symbol	Identifier
11 (17) ATR_DRIVE_BACKOUT	Backout of next UR When this identifier is initially set, RRS will, if any resource manger has called Retain_Interest for the next UR, complete backout for the next UR before returning control to the application program. This processing ensures that the next UR will be backed out.
12 (18) ATR_RESYNC_IN_PROGRESS	Resync in progress A resync in progress condition has occurred. If the UR state is before **in_end** when this identifier is initially set, RRS will harden (or reharden) the UR state at the next state change, which will make the information available on restart.
13 (19) ATR_NEW_LUWID_PSH_UNACCEPTABLE	New LUWID PSH unacceptable A communication resource manager cannot accept a new LUWID presentation header (PSH). RRS takes no action when this identifier is set.
14 (20) ATR_DRIVE_COMPLETION	Invoke completion exit(s) When this identifier is set, RRS will invoke any defined COMPLETION exit routines when the UR is complete. If the UR state is before **in_end** when this identifier is initially set, RRS will harden (or reharden) the UR state, including this identifier, at the next logging point.
21 (33) ATR_APPL_COMPLETE	This identifier indicates completion of an individual UR in a cascaded UR family. Setting this identifier informs RRS that the application executing for this UR is complete. RRS will not commit a cascaded UR family until RRS is informed that all of the individual cascaded URs in the family are complete. **Note:** You cannot specify ATR_APPL_COMPLETE and ATR_RESET_APPL_COMPLETE on the same call to Set_Side_Information.

Constant in: Hexadecimal (Decimal) Equate Symbol	Identifier
22 (34) ATR_RESET_APPL_COMPLETE	Application processing is not complete When this identifier is used, RRS resets the ATR_APPL_COMPLETE identifier. RRS will not commit a cascaded UR family until RRS is informed that all of the individual cascaded URs in the family are complete. **Note:** You cannot specify ATR_APPL_COMPLETE and ATR_RESET_APPL_COMPLETE on the same call to Set_Side_Information.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00130000' or X'00130001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call does not identify one of the currently valid interests. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a UR token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
383 ATR_SIDE_INFO_ID_INV	**Meaning**: Program error. The identifier for a side information value in the *side_info_id* parameter specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
392 ATR_ELEMENT_COUNT_INV	**Meaning**: The specified element count is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
3A3 ATR_UR_TOKEN_INV	**Meaning**: Program error. The UR token specified in the call does not identify a valid UR. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a URI token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
3A8 ATR_AUTH_FAILURE	**Meaning**: Program error. The caller is PKM 8–15 problem state and one of the following occurred: • The caller specified the UR token of a UR associated with a context that neither belongs to a PKM 8–15 problem state resource manager registered in the home address space, nor is it a native context in the home address space. • The caller specified a value for side_info_id other than ATR_APPL_COMPLETE or ATR_RESET_APPL_COMPLETE. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager state must be **run**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
73C ATR_AFTER_NEW_UR	**Meaning**: Program error. The resource manager called the Set_Side_Information service to set `ATR_BREAK_TREE`, but the application is already running under a new UR. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
745 ATR_AFTER_IN_PREPARE	**Meaning**: Program error. The resource manager cannot call the Set_Side_Information service to set `ATR_BACKOUT_REQUIRED` because the UR state is beyond **in-prepare**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
757 ATR_ID_CONFLICT	**Meaning**: Program error. Information identifiers specified on the call conflict with each other. For example, ATR_APPL_COMPLETE and ATR_RESET_APPL_COMPLETE. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
758 ATR_APPL_COMPLETE_INV	**Meaning**: Program error. The specified UR cannot be set as ATR_APPL_COMPLETE. The Set_Side_Information service cannot set a top-level UR ATR_APPL_COMPLETE or ATR_RESET_APPL_COMPLETE. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
761 ATR_APPL_COMPLETE_INV _STATE	**Meaning**: Program error. The specified UR cannot be set as ATR_APPL_COMPLETE or not ATR_APPL_COMPLETE, because the UR is not in a valid state. The system rejects the service call. **Action**: Check the calling program for a probable coding error. Correct the program and rerun it.
766 ATR_SIDE_INFO_ID _LOCAL_INV	**Meaning**: Program error. The identifier (or one of the identifiers) specified in the *side_info_id* array is not allowed when the UR is in local transaction mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Set_Side_Information was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to set the side information for ATR_BACKOUT_REQUIRED.

```
⋮
URI_TOKEN = UR_INTEREST_TOKEN
COUNT = 1
ID = ATR_BACKOUT_REQUIRED
CALL ATRSUSI(RC,URI_TOKEN,COUNT,ID)
IF RC = ATR_OK THEN
⋮
```

Set_Syncpoint_Controls (ATRSSPC, ATR4SSPC)

- ATRSSPC is for AMODE(31) callers.
- ATR4SSPC is for AMODE(64) callers and allows parameters in 64 bit addressable storage.

A resource manager calls the Set_Syncpoint_Controls service in a distributed environment. Set_Syncpoint_Controls defines the role the resource manager will play in processing a UR. Your resource manager can also use the call to set return codes for exit routines that you want RRS to bypass. In response to the call, RRS returns a return code.

A distributed syncpoint resource manager (DSRM) usually issues this call in its STATE_CHECK or PREPARE exit routine. A server distributed syncpoint resource manager (SDSRM) usually issues this call before it initiates a syncpoint operation.

Resource Manager Role in UR Processing: The possible roles for the resource manager for the specified interest in a unit of recovery (UR) are:

- **Distributed syncpoint resource manager (DSRM)**: The resource manager becomes the DSRM and coordinator for the UR. RRS changes from its usual role of coordinator for a UR into an agent of the coordinator. RRS expects the DSRM to be using a peer-to-peer protocol.

  When it specifies this role, the Set_Syncpoint_Controls service enables the resource manager's DISTRIBUTED_SYNCPOINT exit routine. When the local interests in a UR vote to commit the UR, RRS gives control to this routine. For the **in-doubt** UR, the DISTRIBUTED_SYNCPOINT exit routine:

  1. Communicates with another system to determine its prepare vote for the UR
  2. Returns the overall commit or backout vote to RRS

- **Server distributed syncpoint resource manager (SDSRM)**: The resource manager becomes the SDSRM and coordinator for the UR. RRS changes from its usual role of coordinator for a UR into an agent of the coordinator. RRS expects the SDSRM to be using a client/server protocol.

  When it specifies this role, the Set_Syncpoint_Controls service disables the resource manager's ONLY_AGENT exit routine and enables it to use the following services: Backout_Agent_UR, Commit_Agent_UR, Forget_Agent_UR_Interest, and Prepare_Agent_UR. These services allow the resource manager to:

  1. Initiate the prepare phase of a syncpoint operation.
  2. Obtain the result from the prepare vote collection.
  3. Delay and control the initiation of commit or backout processing.
  4. Control the deletion of its entries from log records.

  See “Protecting distributed resources” on page 67 for more information.

- **Last agent participant**: The resource manager becomes the last agent for a distributed syncpoint operation. RRS continues its usual role as coordinator.

  For additional information about the roles involved in a distributed syncpoint operation, see *SNA Sync Point Services Architecture*

- **Participant**: The resource manager takes part in the UR interest; participant is the resource manager's usual role.

  As necessary, a resource manager that has assumed another role can reset itself to participant. Also, any resource manager that wants to prevote exit routines can call Set_Syncpoint_Controls to take the role of participant. If a UR is in local transaction mode, participant is the only role a resource manager can have.

When it hardens information for the interest in an RRS log, RRS records the resource manager's role. However, if the UR state is **in-prepare**, RRS might not record the role and options specified in the Set_Syncpoint_Controls call.

Notes on changing roles: For any interest in a particular UR, RRS prevents any resource manager from successfully calling the Set_Syncpoint_Controls service to request the DSRM, SDSRM, or last agent participant role if any of the following are true:

- A resource manager already has the distributed syncpoint resource manager (DSRM) role for the UR.
- A resource manager already has the server distributed syncpoint resource manager (SDSRM) role for the UR.
- A resource manager already has the last agent participant role for the UR.
- The UR is a cascaded UR.

A communication resource manager with multiple interests in a UR can move a role from one of its interests to another. The movable roles are:

- Distributed syncpoint resource manager
- Server distributed syncpoint resource manager
- Last agent participant

To move a role, the resource manager must call the Set_Syncpoint_Controls service twice, in the following order;

1. For one interest, reset its role to participant from its original role.
2. For the other interest, reset its role from participant to the desired role.

The resource manager is responsible for serializing its processing to make the resets in the correct order.

Once a syncpoint operation has started, the SDSRM role cannot be changed.

Exit routine return codes: In the call, you can specify whether or not RRS is to invoke each of the following exit routines:

- PREPARE exit routine
- COMMIT exit routine
- BACKOUT exit routine

For an interest in a local transaction, a resource manager can prevote only its COMMIT and BACKOUT exits.

If you want RRS to bypass an exit routine, you provide the exit routine's return code on the call.

Note: Your resource manager might need to know whether an RRS panel was used to resolve an **in_doubt** UR or when a resync for the UR is in progress. If so, do not bypass the COMMIT and/or BACKOUT exit routines unless you provide an END_UR exit routine.

Environment

The requirements for the caller are:

Minimum authorization:	PKM allowing key 0-7, or supervisor state
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN

AMODE:	
	31 bit (ATRSSPC)
	64 bit (ATR4SSPC)
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	Control parameters must be in the primary address space and addressable by the caller.
Linkage:	Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The restrictions for the call are:

- The state of the resource manager associated with the specified UR interest token must be **run**.
- The UR state must be **in-flight**, **in-state-check**, or **in-prepare**. To change to or from the SDSRM role, the UR state must be **in-flight**.
- The resource manager's interest must be a protected interest if the requested role is distributed syncpoint resource manager, server distributed syncpoint resource manager, or last agent participant.
- If the UR transaction mode is local, a resource manager cannot change its role from ATR_PARTICIPANT.
- To set the role of DSRM, your resource manager needs a DISTRIBUTED_SYNCPOINT exit routine.

If your resource manager calls the Set_Syncpoint_Controls service asynchronously for a UR in an **in-flight** state, there is no way to ensure that the service will be invoked before the application issues a commit UR request or a backout UR request. In this case, your resource manager must have a PREPARE or STATE_CHECK exit routine, even if you expect to use the call to Set_Syncpoint_Controls to set the return code for the PREPARE exit routine.

A better alternative is either to call Set_Syncpoint_Controls from within a STATE_CHECK exit routine or to call Set_Syncpoint-Controls before the application code can get control.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the call as shown in the syntax diagram. You must code the parameters in the CALL statement as shown.

CALL ATRSSPC	(return_code ,ur_interest_token ,prepare_exit_code ,commit_exit_code ,backout_exit_code ,role)

CALL ATR4SSPC	(return_code ,ur_interest_token ,prepare_exit_code ,commit_exit_code ,backout_exit_code ,role)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Set_Syncpoint_Controls service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

Specifies the UR interest token that identifies an instance of your resource manager's interest in a UR. Your resource manager received the token from the Express_UR_Interest service or the Retain_Interest service.

,prepare_exit_code
Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies whether RRS is to invoke the PREPARE exit routine:

- To invoke your resource manager's PREPARE exit routine, specify `ATR_DRIVE_PREPARE_EXIT`.
- To bypass your resource manager's PREPARE exit, specify ATR_PREPARE_OK or ATR_PREPARE_ABSTAIN to set the return code that the PREPARE exit routine would have returned.

Note: If the UR state is **in-prepare** or if the UR is in local transaction mode, RRS ignores the *prepare_exit_code* parameter.

Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	**Invoke the PREPARE exit routine?**	**PREPARE exit routine return code**
0 (0) ATR_PREPARE_OK	No	ATRX_OK
14 (20) ATR_PREPARE_ABSTAIN	No	ATRX_ABSTAIN To set this value, the resource manager should: • Be the DSRM or SDSRM • Have an END_UR exit routine
FFF (4095) ATR_DRIVE_PREPARE_EXIT	YES	

,commit_exit_code

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies whether RRS is to invoke the COMMIT exit routine:

- To invoke your resource manager's COMMIT exit routine, specify `ATR_DRIVE_COMMIT_EXIT`.
- To bypass your resource manager's exit routine, set the return code for the COMMIT exit routine.

Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Invoke the COMMIT exit routine?	COMMIT Exit routine return code
0 (0) ATR_COMMIT_OK	No	ATRX_OK
FFF (4095) ATR_DRIVE_COMMIT_EXIT	Yes	

,backout_exit_code

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies whether the resource manager is to invoke the BACKOUT exit routine:

- To invoke your resource manager's BACKOUT exit routine, specify `ATR_DRIVE_BACKOUT_EXIT`.
- To bypass your resource manager's exit routine, set the return code for the BACKOUT exit routine.

Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Invoke the BACKOUT exit routine?	BACKOUT exit routine return code
0 (0) ATR_BACKOUT_OK	No	ATRX_OK
FFF (4095) ATR_DRIVE_BACKOUT_EXIT	Yes	

,role

Supplied parameter

- Type: Integer
- Length: 4 bytes

Defines the role your resource manager wants to take for the specified UR interest. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	Description
0 (0) ATR_PARTICIPANT	**Participant**: For this interest, the resource manager is to be a participant.
1 (1) ATR_LAST_AGENT	**Last-agent participant**: For this interest, the resource manger is to be the last-agent participant. Specify this role only if the UR interest token represents a protected interest.
2 (2) ATR_DSRM	**Distributed syncpoint resource manager**: For this interest, the resource manager is to be the distributed syncpoint resource manager. Specify this role only if the UR interest token represents a protected interest.
3 (3) ATR_SDSRM	**Server distributed syncpoint resource manager**: For this interest, the resource manager is to be the server distributed syncpoint resource manager. Specify this role only when the UR interest token represents: • A protected interest • A UR that is **in-flight**

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00120000' or X'00120001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call is not one of the currently valid interests. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
373 ATR_PREPARE_CODE_INV	**Meaning**: Program error. The *prepare_exit_code* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
374 ATR_COMMIT_CODE_INV	**Meaning**: Program error. The *commit_exit_code* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
387 ATR_PREPARE_CODE_INCORRECT	**Meaning**: Program error. The *prepare_exit_code* value specified in the call is incorrect. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
390 ATR_ROLE_INV	**Meaning**: Program error. The role specified in the call in not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
394 ATR_BACKOUT_CODE_INV	**Meaning**: Program error. The *backout_exit_code* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager state must be **run**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
731 ATR_UR_STATE_ERROR	**Meaning**: Program error. The UR is not in a valid state for the service call. The UR state must be **in-flight**, **in-state-check**, or **in-prepare**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
732 ATR_NO_DIST_SYNC_EXIT	**Meaning**: Program error. A resource manager that has taken the DSRM role did not set a DISTRIBUTED_SYNCPOINT exit routine with RRS. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
733 ATR_SSPC_ROLE_ERROR_DSRM	**Meaning**: Program error. A resource manager has already obtained the distributed syncpoint resource manager role in a call to Set_Syncpoint_Controls. This action prevents subsequent calls for this UR to Set_Syncpoint_Controls for either the distributed syncpoint resource manager role, the server distributed syncpoint manager role, or the last agent role. The system rejects the service call. **Action**: Check the resource manager for a coding error. Correct the resource manager and rerun it. If your resource manager does not have a coding error, this problem may be beyond its control. Your resource manager might need to take other actions, such as interacting with the system operator, to indicate that this UR cannot be processed successfully. Another possible approach is to call the Set_Side_Information service to set backout required for the UR.
734 ATR_SSPC_ROLE_ERROR_LAST_AGENT	**Meaning**: Program error. A resource manager has already obtained the last agent role in a call to the Set_Syncpoint_Controls service. This action prevents subsequent calls for this UR to the Set_Syncpoint_Controls service for either the distributed syncpoint resource manager role or the last agent role. The system rejects the service call. **Action**: Check the resource manager for a coding error. Correct the resource manager and rerun it. If your resource manager does not have a coding error, this problem may be beyond control by the resource manager. Your resource manager might need to take other actions, such as interacting with the system operator, to indicate that this UR cannot be processed successfully. Another possible approach is to call the Set_Side_Information service to set backout required for the UR.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
746 ATR_ROLE_INCORRECT	**Meaning**: Program error. The interest for the *ur_interest_token* specified in the call is not protected, or the UR is in local transaction mode. The specified role (`ATR_DSRM`, `ATR_LAST_AGENT`, or `ATR_SDSRM`) is valid only for a protected interest in a global or hybrid-global mode transaction. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
74B ATR_SSPC_ROLE_ERROR_SERVER_ DSRM	**Meaning**: Program error. A resource manager has already obtained the server distributed syncpoint resource manager role in a call to Set_Syncpoint_Controls. This action prevents subsequent calls for this UR to Set_Syncpoint_Controls for either the distributed syncpoint resource manager role, the server distributed syncpoint manager role, or the last agent role. The system rejects the service call. **Action**: Check the resource manager for a coding error. Correct the resource manager and rerun it. If your resource manager does not have a coding error, this problem may be beyond its control. Your resource manager might need to take other actions, such as interacting with the system operator, to indicate that this UR cannot be processed successfully. Another possible approach is to call the Set_Side_Information service to set backout required for the UR.
74F ATR_ROLE_CHANGE_AFTER_SYNC	**Meaning**: Program error. The resource manager tried to set a role change after a syncpoint operation had begun. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
759 ATR_ROLE_ERROR_CASCADED_UR	**Meaning**: Program error. The specified UR is a cascaded UR. Only the participant role is valid for cascaded URs. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Set_Syncpoint_Controls was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to request the distributed syncpoint resource manager role and bypass the exit routines.

```
⋮
URI_TOKEN = MY_URI_TOKEN
PREP_CODE = ATR_PREPARE_ABSTAIN
CMIT_CODE = ATR_COMMIT_OK
BACK_CODE = ATR_BACKOUT_OK
ROLE = ATR_DSRM
```

```
CALL ATRSSPC(RC,URI_TOKEN,PREP_CODE,CMIT_CODE,BACK_CODE,ROLE)
IF RC ≠ ATR_OK THEN
   /* Handle error */
⋮
```

Set_Work_Identifier (ATRSWID, ATRSWID2, ATR4SWID)

The resource manager calls the Set_Work_Identifier service to set the current or next unit of work identifier (UWID) for a unit of recovery (UR). In response to the call, RRS returns a return code. There are three versions of Set_Work_Identifier, each with different parameters.

- ATRSWID is for AMODE(31) callers and is the basic version of the service. It must be called specifying a UR interest token.
- ATRSWID2 is for AMODE(31) callers and can be called specifying either a UR token or a UR interest token.
- ATR4SWID is for AMODE(64) callers, allows parameters in 64 bit addressable storage and can be called specifying either a UR token or a UR interest token.

Code your resource manager to call the version that includes the support you need.

Table 45 describes the UWIDs that this service can set. Your resource manager can use the call to set a UWID only if one does not already exist for the UR; the UWID must be null before the call.

Table 45. Setting Unit of Work Identifiers

Unit of Work Identifier (UWID)	**Format**	**Current UWID can be set**	**Next UWID can be set**
LU 6.2 logical unit of work identifier (LUWID)	`netid.luname.instnum.seqnum` **netid.luname** 1-17 character identifier of the network and LU, preceded by a 1-byte fixed length field **instnum** 6-byte fixed TP instance **seqnum** 2-byte fixed sequence number	Yes	Yes
Enterprise Identifier (EID)	`tidgtid` **tid** 4-byte transaction identifier (TID) **gtid** 8-40 byte global transaction identifier (GTID)	Yes	No

Table 45. Setting Unit of Work Identifiers (continued)

Unit of Work Identifier (UWID)	Format	Current UWID can be set	Next UWID can be set
X/Open Identifier (XID)	`FormatIDGtrid_lengthBqual_lengthID` **FormatID** 4-byte fixed format ID **Gtrid_length** 4-byte fixed GTRID length **Bqual_length** 4-byte fixed BQUAL length **ID** 128-byte character XID The GTRID length and BQUAL length define the length of the first and second subsection of the ID. The GTRID must have a length of at least 1 byte, however the BQUAL can have a length of 0. The length of the entire XID cannot exceed 140 bytes.	Yes	No

The XID for a unit of recovery can also be set with the Express_UR_Interest service.

The next LUWID is used for the next UR, unless ATR_BREAK_TREE was set in the UR's side information by a Set_Side_Information call.

All of the URs in a cascaded UR family will have the same GTRID component in their XID.

Environment

The requirements for the caller are:

Minimum authorization: PKM allowing key 0-7, or supervisor state
Dispatchable unit mode: Task or SRB
Cross memory mode: Any PASN, any HASN, any SASN
AMODE:
31 bit (ATRSWID, ATRSWID2)
64 bit (ATR4SWID)
ASC mode: Primary or access register (AR)
Interrupt status: Enabled for I/O and external interrupts
Locks: No locks held
Control parameters: Control parameters must be in the primary address space and addressable by the caller.
Linkage: Standard MVS linkage conventions are used.

Programming requirements

Either link edit your object code with the linkable stub routine ATRRCSS (31 bit) or ATRR4CSS (64 bit) from SYS1.CSSLIB, or LOAD and CALL the callable service. The high level language (HLL) definitions for the callable service are:

HLL definition	Description
ATRRASM	390 Assembler declarations
ATRRC	C/390 declarations
FOMURRC	z/OS HFS header files

Restrictions

The state of the resource manager associated with the specified UR interest token must be **run**, which means it has registered, set its exit routines with RRS, and completed restart.

You cannot set a global work identifier for a UR in local transaction mode.

For the specified interest in the UR, your resource manager cannot call the Set_Work_Identifier service if any of the following are true:

- A resource manager's exit routine returns `ATRX_LATER_CONTINUE`
- The UR state is **in_completion**.
- For a server distributed syncpoint manager (SDSRM), the UR state is **in_forget**.

When the resource manager issues the call in SRB mode, the call must not specify binary zero for *ur_interest_token* or *ur_or_uri_token*.

Input register information

Before issuing the call, the caller does not have to place any information into any register unless using it in register notation for the parameters, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:.

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a call. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

Write the appropriate call as shown in the syntax diagrams. You must code the parameters in the CALL statement as shown.

CALL ATRSWID	(return_code ,ur_interest_token ,set_option ,uwid_type ,uwid_len ,uwid_data)

CALL ATRSWID2	(return_code ,ur_or_uri_token ,set_option ,uwid_type ,uwid_len ,uwid_data)

CALL ATR4SWID	(return_code ,ur_or_uri_token ,set_option ,uwid_type ,uwid_len ,uwid_data)

Parameters

The parameters are explained as follows:

return_code
Returned parameter

- Type: Integer
- Length: 4 bytes

Contains the return code from the Set_Work_Identifier service.

,ur_interest_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRSWID callers, specifies a token that uniquely identifies your resource manager's interest in the UR whose data you want to set. Your resource manager received the token from one of the following services: Express_UR_Interest, Retain_Interest.

,ur_or_uri_token
Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: 16 bytes

For ATRSWID2 callers, specifies a token that uniquely identifies either the UR, or your resource manager's interest in the UR, whose data you want to set:

- UR token: The token for the UR.
- UR interest token: The UR interest token that identifies your resource manager's interest in the UR.

Your resource manager received the token from one of the following services: Express_UR_Interest, Retrieve_Interest_Data, Retain_Interest, Create_Cascaded_UR, or Retrieve_UR_Data.

Because you may pass two different types of tokens through this parameter, passing an invalid token can generate either a ATR_URI_TOKEN_INV or a ATR_UR_TOKEN_INV return code. For example, passing an invalid UR token might result in an ATR_URI_TOKEN_INV return code. Even though a UR token was passed, if it is invalid, then RRS may not understand what sort of token it was supposed to be. For this reason, IBM recommends callers check both return codes, even when they know what type of token they intend to pass.

,set_option
Supplied parameter

- Type: Integer
- Length: 4 bytes

Identifies the UWID to be set. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	**UWID to be set**
0 (0) ATR_CURRENT	**Current UWID**: RRS is to set the current UWID for this UR.
1 (1) ATR_NEXT	**Next UWID**: RRS is to set the next UWID for this UR. The next UWID cannot be an EID or XID.

,uwid_type
Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the type of the UWID you want to set. Specify one of the following:

Constant in: Hexadecimal (Decimal) Equate Symbol	UWID Type
0 (0) ATR_LUWID	An LU 6.2 logical unit of work identifier (LUWID)
1 (1) ATR_EID	An Enterprise identifier (EID)
2 (2) ATR_XID	An X/Open transaction identifier (XID)

,uwid_len

Supplied parameter

- Type: Integer
- Length: 4 bytes

Specifies the length of the UWID your resource manager wants to set. The length of the UWID depends on its type. Specify the actual length of the UWID between the following upper and lower limits:

Maximum Length in: Hexadecimal (Decimal) Equate Symbol	UWID type
A (10) ATR_MIN_LUWID_LENGTH	The minimum length of a LU 6.2 LUWID
1A (26) ATR_MAX_LUWID_LENGTH	The maximum length of a LU 6.2 LUWID
C (12) ATR_MIN_EID_LENGTH	The minimum length of an Enterprise identifier
2C (44) ATR_MAX_EID_LENGTH	The maximum length of an Enterprise identifier
D (13) ATR_MIN_XID_LENGTH	The minimum length of an X/Open identifier

Maximum Length in: Hexadecimal (Decimal) Equate Symbol	UWID type
8C (140) ATR_MAX_XID_LENGTH	The maximum length of an X/Open identifier

,uwid_data

Supplied parameter

- Type: Character string
- Character Set: No restriction
- Length: Specified in *uwid_len* parameter

Specifies the contents of the UWID your resource manager wants to set.

The format of the UWID depends on the UWID type. A LUWID has the following format:

```
netid.luname.instnum.seqnum
```

The fields are as follows:

netid.luname
1-17 character identifier of the network and LU, preceded by a 1-byte length field

instnum
6-byte TP instance

seqnum
2-byte sequence number

An EID has the following format:

```
tidgtid
```

The fields are as follows:

tid 4-byte transaction identifier (TID)

gtid 8-40 byte global transaction identifier (GTID)

For XID, the *uwid_data_buffer* contains the 4–byte address of the buffer to contain the retrieved XID. An XID has the following format:

```
FormatIDGtrid_lengthBqual_lengthID
```

The fields are as follows:

FormatID
4-byte fixed format ID

Gtrid_length
4-byte fixed Gtrid length

Bqual_length
4-byte fixed Bqual length

ID 128-byte character XID

The 1–128 byte ID field has the following format:

Gtrid 1–64 byte Gtrid

Bqual 0–64 byte Bqual

The length of the entire XID cannot exceed 140 bytes.

ABEND codes

The call might result in an abend X'5C4' with a reason code of either X'00140000' or X'00140001'. See *z/OS MVS System Codes* for the explanations and actions.

Return codes

When the service returns control to the resource manager, GPR 15 and *return_code* contain a hexadecimal return code.

Return Code in: Hexadecimal Equate Symbol	**Meaning and action**
0 ATR_OK	**Meaning**: Successful completion. **Action**: None.
103 ATR_INTERRUPT_STATUS_INV	**Meaning**: Program error. The resource manager is disabled; the interrupt status must be enabled for I/O and external interrupts. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
105 ATR_LOCKS_HELD	**Meaning**: Program error. The resource manager is holding one or more locks; no locks must be held. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
107 ATR_UNSUPPORTED_RELEASE	**Meaning**: Environmental error. The system release does not support this service. The system rejects the service call. **Action**: Remove the resource manager from the system, and install it on a system that supports RRS. Then rerun the resource manager.
370 ATR_URI_TOKEN_INV	**Meaning**: Program error. The UR interest token specified in the call does not identify one of the currently valid interests. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a UR token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
377 ATR_UWID_LEN_INV	**Meaning**: Program error. The UWID length specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
37F ATR_SET_OPTION_INV	**Meaning**: Program error. The *set_option* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
380 ATR_UWID_TYPE_INV	**Meaning**: Program error. The *uwid_type* value specified in the call is not valid. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
393 ATR_LUWID_DATA_INV	**Meaning**: Program error. The LUWID specified in *uwid_data* is not valid. The first byte of this data must contain a valid length of an LU name (a value from 1 to 17). The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
397 ATR_XID_DATA_INV	**Meaning**: Program error. The computed length of the XID does not match the specified length passed via the *uwid_len* parameter. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
3A3 ATR_UR_TOKEN_INV	**Meaning**: Program error. The UR token specified in the call does not identify a valid UR. If the specified token is not a valid UR or URI token, RRS may return this return code even if the resource manager was attempting to specify a URI token. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
701 ATR_RM_STATE_ERROR	**Meaning**: Program error. The resource manager associated with the UR interest token specified in the call is not in a valid state to issue the service call. The resource manager state must be **run**. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
702 ATR_RM_EXITS_UNSET	**Meaning**: Program error. RRS has unset the RRS exit routines for the resource manager. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
735 ATR_UWID_ALREADY_SET	**Meaning**: The UR already has a UWID. The system rejects the service call. It is possible that another program in the system previously set the UWID. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
73C ATR_AFTER_NEW_UR	**Meaning**: Program error. The application is already running under a new UR. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
74E ATR_SET_NEXT_EID_INV	**Meaning**: Program error. The resource manager tried to set the next Enterprise identifier, but the requested function is not available. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.
752 ATR_SET_NEXT_XID_INV	**Meaning**: Program error. The next XID cannot be set. The system rejects the service call. **Action**: Check the resource manager for a probable coding error. Correct the resource manager and rerun it.

Return Code in: Hexadecimal Equate Symbol	Meaning and action
764 ATR_LOCAL_TRAN_MODE_INV	**Meaning**: Program error. The current UR is in local transaction mode. This service is valid only for a UR in global transaction mode. The system rejects the service call. **Action**: Check the calling program for a probable coding error. If the caller is a resource manager, it should not unset its exits with RRS.
F00 ATR_NOT_AVAILABLE	**Meaning**: RRS is not available. **Action**: The system rejects the service request. Retry the request later. Before retrying the request, the resource manager must reset its RRS exit routine information and begin restart processing with RRS.
F06 ATR_WAS_NOT_AVAILABLE	**Meaning**: RRS was available to the resource manager, but went down and came back up again. A commit or backout operation may or may not have been in progress for the context under which the Set_Work_Identifier was done at the time of the RRS failure. A new unit of recovery can not be created until the current unit of recovery is completed. **Action**: The system rejects the service request. Restart your resource manager, making sure to reset the resource manager's exit routines with RRS. The resource manager must inform the application that one of the following actions must be taken to complete the current unit of recovery: • If a commit or backout request was not active at the time of the RRS failure, a commit or backout must be requested before a new unit of recovery can begin. • If a commit or backout request was active at the time of the RRS failure, the context must be ended, via the CTXENDC service, before a new unit of recovery can begin.
FFF ATR_UNEXPECTED_ERROR	**Meaning**: System error. The service that was called encountered an unexpected error. The system rejects the service call. **Action**: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Example

In the pseudocode example, the resource manager issues a call to set a UWID.

```
⋮
URI_TOKEN = UR_INTEREST_TOKEN
SET_OPT = ATR_CURRENT
UWID_TYPE = ATR_LUWID
LUWID_LEN = 26
LUWID = LUWID_1
CALL ATRSWID2(RC,URI_TOKEN,SET_OPT,UWID_TYPE,LUWID_LEN,LUWID)
IF RC ≠ ATR_OK THEN
  /* Handle error */
⋮
```

Chapter 8. RRS setup and control

If your installation runs a resource manager that provides resource recovery through RRS, you might need some or all of the following information to help you manage resource recovery. Your resource manager might provide related information. This chapter includes the following topics:

Defining RRS as a subsystem

To define RRS as a subsystem, place the following statement in the IEFSSNxx parmlib member:

```
SUBSYS SUBNAME(RRS)
```

Place this statement after the statement that defines the primary subsystem. You can replace RRS with a subsystem name of your choice, but do not supply any other parameters. In particular, do not supply an initialization routine. For more information about IEFSSNxx, see *z/OS MVS Initialization and Tuning Reference*.

Or, issue the `SETSSI ADD,SUBNAME=RRS` console command after IPL to dynamically define the RRS subsystem to the SSI. You can replace RRS with a subsystem name of your choice but do not supply any other parameters. In particular, do not supply an initialization routine. For more information about the `SETSSI` command, see *z/OS MVS System Commands*.

Note: RRS does not support any of the other functions provided by the `SETSSI` command; especially the `ACTIVATE` function. RRS can only be started by using the procedure documented in the next section.

Establishing dispatching priority of the RRS address space

You must establish the dispatching priority of the RRS address space. The best way to control RRS's dispatching priority is through the workload manager (WLM). IBM recommends that you put RRS in the SYSSTC service class. The service class you choose must give RRS a dispatching priority greater than or equal to the

dispatching priority of applications and resource managers that use RRS. SYSSTC will usually accomplish this. For information about system-provided service classes, see *z/OS MVS Planning: Workload Management*.

Creating default RRS CTRACE parmlib member

If no trace options are supplied in a named parmlib member or in a REPLY to a TRACE operator command, RRS component trace will trace only unexpected events. You can find information about the CTncccxx (component trace) parmlib member in *z/OS MVS Initialization and Tuning Reference*, and information about component trace for RRS in *z/OS MVS Diagnosis: Tools and Service Aids*.

RRS performance can be severely impacted by component tracing. For this reason, a CTRACE parmlib member for RRS is not provided since you might start CTRACing causing performance degradation without needing or knowing about it. However, for problem determination, CTRACing can provide valuable information. Member ATRCTRRS supplied in SYS1.SAMPLIB is available with an optimal set of CTRACE parameters used for debugging RRS problems. The BUFSIZE in the sample has been established to hold 10-30 minutes of tracing on most systems. When problem determination is needed, see the SET-UP and ACTIVATION instructions in the ATRCTRRS sample.

Creating a cataloged procedure for starting RRS

IBM supplies, in SYS1.SAMPLIB, a cataloged procedure named ATRRRS that you can use to start RRS after system initialization. Your installation should copy SYS1.SAMPLIB(ATRRRS) to SYS1.PROCLIB(RRS). The membername RRS specified here can be replaced with any other membername, as long as it matches the subsystem name specified in the SYS1.PARMLIB(IEFSSNxx) used by the installation. If the names do not match, you may receive error messages when you start the subsystem.

The contents of ATRRRS are:

```
//RRS  PROC  CTMEM='',GNAME=''
//RRS  EXEC  PGM=ATRIMIKE,REGION=0K,TIME=NOLIMIT,
             PARM='GNAME=&GNAME,CTMEM=&CTMEM'
```

The parameters in the procedure are as follows:

CTMEM
Specifies the CTnRRSxx parmlib member that RRS component trace is to use. CTMEM='' indicates that the START command can supply the member name.

GNAME
Specifies the log group name. A log group is a group of systems that share an RRS workload. Specify a value if your installation needs an RRS log group that is a subset of the systems in a sysplex. Otherwise, the name defaults to the sysplex name.

If you specify a name, it must be 1-8 characters in length. The first character must be alphabetic or @, #, or $. The remaining characters must be alphabetic, numeric, or @, #, or $.

PGM=ATRIMIKE
Specifies the initialization routine for the RRS jobstep task. PGM=ATRIMIKE is required.

REGION=0K
Specifies that the address space is to have the largest allowable size on the system.

TIME=NOLIMIT
Specifies that there is no time limit for RRS.

If you used the GNAME parameter to define a log group that is a subset of the sysplex, take great care to ensure that a resource manager always restarts on a system within the same log group. If a resource manager restarts on a system within a different log group, RRS does not detect the discrepancy; it assumes that the resource manager is doing a cold start.

Defining RRS to automatic restart management (ARM)

If RRS fails, it can use automatic restart management (ARM) to restart itself in a different address space on the same system. RRS, however, will not restart itself following a SETRRS CANCEL command. To stop RRS and cause it to restart automatically, use the FORCE command with ARM and ARMRESTART.

To make automatic restart possible, your installation must:

- Provide an ARM couple data set that contains, either explicitly or through defaults, an automatic restart management policy for RRS. When setting up your ARM policy, use the element name SYS_RRS_*sysname* for RRS.
- Activate the ARM couple data set through a COUPLExx parmlib member or a SETXCF operator command. The data set must be available when RRS starts and when it restarts.
- Make sure that no element-restart denies the restart of an RRS element or changes its restart. An exception is an exit routine that vetoes RRS restart but then itself starts the RRS address space. This technique, however, might delay other elements in the restart group that have to wait for RRS services to become available.

As with other automatic restart management elements, an ENF signal for event 38 occurs when RRS registers with automatic restart management or is automatically restarted.

For information about automatic restart management parameters, see *z/OS MVS Setting Up a Sysplex*.

Configuring and defining RRS logging requirements

RRS uses six log streams that can be shared by multiple systems in a sysplex. If more than one system wants to use the same set of RRS log streams at the same time, then the log streams must reside in structures. Each system that wants to connect to the set of log streams will then need to have access to the coupling facility (CF) and the DASD on which the system logger offload data set will reside. If only one system with one RRS image is going to use a particular set of log streams, or the log streams are used in a sysplex in which information should not be shared among RRS images, then the log streams can be defined as DASDONLY log streams. If DASDONLY log streams are used, logger allocates a staging dataset, not a structure, to which to write data in the interim, so no CF is required. *z/OS MVS Setting Up a Sysplex* contains information about the tasks you need to perform related to the system logger set up. See “Defining the log streams” on page 539 for specific details related to RRS.

The RRS images on different systems in a sysplex run independently. However, RRS images that are in the same log group share log streams to keep track of the work. If a system in the same log group fails, RRS on a different system in the same log group in the sysplex can use the shared logs to take over the failed system's work. If there is only one system connected to the structure-based log streams, or the log streams are DASDONLY, then no other system will take over the failed system's work. Any outstanding syncpoints will be resolved when RRS restarts using the logging group, and the resource managers become active within that logging group.

Table 46 summarizes the RRS logs. In the figure, *gname* is the log group name. A log group is a group of systems that share an RRS workload. The default log group name is the sysplex name.

Table 46. RRS Logs

Data set name and log name	Contents	Storage requirements
ATR.gname.ARCHIVE RRS archive log	Information about completed URs. This log is recommended but optional.	High
ATR.gname.RM.DATA RRS resource manager data log	Information about the resource managers using RRS services.	Low, if few resource managers; Medium, if many resource managers
ATR.gname.RM.METADATA RRS resource manager Meta Data log	Any collection of data that a resource manger wants to save. This log is optional.	Low
ATR.gname.MAIN.UR RRS main UR state log	The state of active URs. RRS periodically moves this information into the RRS delayed UR state log when UR completion is delayed.	High
ATR.gname.DELAYED.UR RRS delayed UR state log	The state of active URs, when UR completion is delayed.	High
ATR.gname.RESTART RRS restart log	Information about incomplete URs needed during restart. This information enables a functioning RRS instance to take over incomplete work left over from an RRS instance that failed.	Medium

Your installation might require an RRS log group that is a subset of the systems in a sysplex. Using different logging groups allows:

- The separation of production and test environments that exist in the same sysplex.
- "Rolling" cold starts of RRS.

Use caution when setting up logging groups because RRS is unaware of separate groups, and resource managers can restart in an unintended group. IBM recommends using ARM to control restart locations.

To use a log group name different from the sysplex name, define the name on the procedure used to start RRS. Otherwise, the name defaults to the sysplex name. See “Creating a cataloged procedure for starting RRS” on page 536 for more information.

To minimize any risk of losing data for structure-based log streams, you can specify that a staging dataset should be used as the duplexing medium at all times. Specifying DUPLEXMODE(UNCOND) when defining a structure-based log stream tells system logger to allocate and use staging datasets as the duplexing medium on all systems connected to the log stream. If an error occurs in the coupling facility's data, the DASD backup is a reliable copy of valid data that is available for restart.

Note that duplexing the logs can significantly slow performance, and RRS will run effectively without duplexing. But, if RRS logs are damaged, RRS might be unable to maintain integrity for the work it coordinates, resulting in inconsistent resources or RRS failure.

While your installation must decide on the risk it can afford to take, IBM strongly recommends that you use unconditional duplexing for both the resource manager data log (ATR.gname.RM.DATA) and the restart log (ATR.gname.RESTART), because any loss of data, unresolved gap, or permanent error against either of these logstreams will force an RRS cold start. The RM.DATA and RESTART logs are small and infrequently updated, so the impact on performance is minimal.

Defining the log streams

z/OS MVS Setting Up a Sysplex contains information about planning for system logger applications. To define the RRS log streams, use the information about system logger in that book along with the following details.

Use the following information to plan the system logger configuration:

define the log stream as a coupling facility log stream or a DASD-only log stream

For a coupling facility log stream, you must perform all the setup steps in *z/OS MVS Setting Up a Sysplex*.

For a DASD-only log stream, you must perform all the set up steps in *z/OS MVS Setting Up a Sysplex* and see "Using System Logger Services" in *z/OS MVS Programming: Assembler Services Guide* for special concerns about DASD-only log streams.

define the number of log streams

RRS uses 6 log streams, although only 4 are required. The archive and meta data logs are optional.

plan DASD log data sets and staging data sets

System logger allocates VSAM linear data sets for the DASD log data sets and DASD staging data sets. Even if you do not use DUPLEXMODE(UNCOND), it is a good idea to provide staging data sets, and system logger requires the log data sets. Be sure to specify VSAM shareoptions of 3,3 for the DASD log data sets and the DASD staging data sets. See "Set Up the SMS Environment for DASD Data Sets" in *z/OS MVS Setting Up a Sysplex* for more information.

size DASD log data sets and staging data sets

If your RRS log streams are using a coupling facility (CF), then size each staging data set to be large enough to hold all the log data that can reside in the CF structure associated with each RRS log stream. The size of the staging

data set should be the same as the value specified on the SIZE parameter in the CFRM policy couple data set. If your RRS log streams are DASD-only log streams, then determine the size of the CF structure that would be needed by each log stream, and make the staging dataset as big as the CF structure would be. See "Plan Space for Staging Data Sets" in *z/OS MVS Setting Up a Sysplex* for more information.

Note: If the log stream is defined to use a structure, and no STG_SIZE parameter is coded, then the default size for a staging dataset is the size of the structure.

Mapping log streams to structures

Each non-DASD-only log stream must be mapped to a coupling facility structure. Coupling facility structures are defined in the CFRM policy. Log streams are then mapped to those structures via the LOGR policy. A basic structure definition and mapping for the RRS log streams with an ARCHIVE log would define six structures and map the log streams as shown in Table 47.

Table 47. Basic Coupling Facility Structures

Structure name	Log streams mapped to the structure
Structure One	MAIN.UR
Structure Two	DELAYED.UR
Structure Three	ARCHIVE
Structure Four	RM.DATA
Structure Five	RESTART
Structure Six	METADATA

If you choose not to use the optional ARCHIVE and METADATA logs, then only four structures need to be defined.

The amount of storage that will be required by each of the log streams will depend on the number of applications using RRS and the amount of work RRS has to do. The CFSIZER tool can be used to obtain initial sizes of the structures to be used by RRS. For more information about the CFSIZER tool, see *z/OS MVS Setting Up a Sysplex*. Table 48 provides a starting point for new users of RRS. This table specifies the maximum and initial sizes of each structure:

Table 48. RRS Structure Sizes

Structure name	Log stream	Writes per sec
Structure One	MAIN.UR	100
Structure Two	DELAYED.UR	20
Structure Three	ARCHIVE	20
Structure Four	RM.DATA	10
Structure Five	RESTART	20
Structure Six	RM.METADATA	10

Note: IBM created these general recommendations for the amount of structure storage required for the various RRS log streams through experience testing various workloads. These recommendations should result in reasonably efficient usage of coupling facility storage while minimizing the likelihood that you will

have to redefine the structures due to variations in your workload. However, the exact amount of storage you need for log streams will depend on your actual usage.

The information below is also useful when defining the log streams:

AVGBUFSIZE (average buffer size)
Specify the average size of the block RRS writes to a log:

RM.DATA — 252 bytes

MAIN.UR — 158 bytes (average UR size)

DELAYED.UR — 158 bytes (average UR size)

RESTART — 158 bytes (average UR size)

ARCHIVE — 262 bytes (average UR size + control information)

RM.METADATA — 8460 bytes (8192 bytes of data + control information)

The AVGBUFSIZE is specified when defining the structure to the LOGR couple data set. If your couple data set has the format provided with OS/390® Release 3, RRS will regularly reset the average buffer size to the optimum value.

MAXBUFSIZE (maximum buffer size)
Specify the maximum size of the block RRS writes to a log:

RM.DATA — 1024 bytes

MAIN.UR — 65276 bytes (64K–260 bytes)

DELAYED.UR — 65276 bytes (64K–260 bytes)

RESTART — 65276 bytes (64K–260 bytes)

ARCHIVE — 65276 bytes (64K–260 bytes)

RM.METADATA — 8460 bytes

If a log stream resides in a structure, then the MAXBUFSIZE is specified when defining the structure to the LOGR couple data set. If the log stream is DASDONLY, then the MAXBUFSIZE is a log stream-defined keyword.

You also need the following **information about each log stream** to determine the amount of coupling facility space you will need.

LOWOFFLOAD
The point, in a percentage, when system logger will stop offloading coupling facility data to the DASD data sets for this log stream. For all RRS log streams, IBM recommends that you use different LOWOFFLOAD defaults for each log stream as follows:

- RM.DATA — 20
- MAIN.UR — 40
- DELAYED.UR — 40
- RESTART — 20
- ARCHIVE — 0
- RM.METADATA — 20

HIGHOFFLOAD
The point, in a percentage, when system logger is to begin offloading coupling facility data to the DASD data sets for this log stream. For all RRS log streams, IBM recommends that you use the HIGHOFFLOAD default of 80.

RRS archive log

RRS writes to the archive log for each completed UR. RRS never uses information written on the RRS archive log; the information is intended for the installation to use if a catastrophic problem occurs. To manage the offload datasets allocated for this log stream, use retention period and autodelete support provided by the system logger.

RRS writes in the RRS archive log for each completed UR.

RRS RM meta data log

The following describes the RRS RM Meta Data Log:

1. RRS never uses information written on the RRS RM meta data log. It is resource manager specific intended for their own use.
2. Since the meta data log is optional, RRS can be started without it. Message ATR132I will be written to SYSLOG and RMs will not be allowed to set/retrieve RM meta data.
3. Should the RM meta data log be defined while RRS is operational, RRS will connect to the log automatically. Once connected, RRS will make sure the MAXBUFSIZE is met or exceeded. If the MAXBUFSIZE is below the required size, RRS will disconnect from the log and issue message ATR172E. RRS will continue to connect and check until the properly sized meta data log is defined. Only then can a resource manager request meta data usage (via set exit information CRGSEIF/CRGSEIF1/CRG4SEIF) and then set and retrieve meta data.
4. When RRS connects to a properly sized RM meta data log, the connection will remain until RRS is terminated.
5. If a resource manager becomes unset, either due to RRS termination or because the resource manager has itself requested the state change, the resource manager must restart and again request meta data usage via set exits before using meta data.

Defining the logs

To define the RRS log streams, use IXCMIAPU, a utility program provided in the SYS1.MIGLIB system library. Each structure-based log stream needs to be mapped to a coupling facility structure. An example of JCL to map two of the RRS log streams to two structures is:

```
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
    DATA TYPE(LOGR)
    DEFINE STRUCTURE
      NAME(LIST15)
  LOGSNUM(1)
  MAXBUFSIZE(1024)
  AVGBUFSIZE(252)
  DEFINE STRUCTURE
      NAME(LIST14)
      LOGSNUM(1)
      MAXBUFSIZE(65276)
      AVGBUFSIZE(158)
    DEFINE LOGSTREAM NAME(ATR.PLEX.RM.DATA)
      STRUCTURE(LIST15)
      LS_DATACLAS(VSAMLS)
      LS_SIZE(size of offload datasets)
      HLQ(RRS)
      LOWOFFLOAD(20)
```

```
      HIGHOFFLOAD(80)
      STG_DUPLEX(YES)
      DUPLEXMODE(UNCOND)
      STG_SIZE(number of 4K blocks for staging dataset)
    DEFINE LOGSTREAM NAME(ATR.PLEX.MAIN.UR)
      STRUCTURE(LIST14)
      LS_DATACLASS(VSAMLS)
      LS_SIZE(size of offload datasets)
      HLQ(RRS)
      LOWOFFLOAD(40)
      HIGHOFFLOAD(80)
/*
```

Notice in this example that the RM.DATA log stream DUPLEXMODE(UNCOND) is specified so that the data written to that log will also be written to a staging dataset. The MAIN.UR log stream will write the log blocks to the structure and to the system logger's buffers until the data is offloaded to DASD. Note that the structures to which the log streams are mapped must also be defined in the active CFRM policy. Also, if the STG_SIZE is not specified, the default value will be the size of the structure.

An example of JCL to define a DASDONLY log stream:

```
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
   DATA TYPE(LOGR)
   DEFINE LOGSTREAM NAME(ATR.PLEX.RM.DATA)
      DASDONLY(YES)
      LS_DATACLAS(VSAMLS)
      LS_SIZE(size of offload datasets)
      HLQ(RRS)
      LOWOFFLOAD(20)
      HIGHOFFLOAD(80)
      STG_SIZE(number 4K blocks for staging dataset)
      MAXBUFSIZE(1024)
/*
```

An example of JCL to delete a single RRS log stream and a structure is:

```
//STEP2    EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
    DATA TYPE(LOGR)
    DELETE LOGSTREAM NAME(log.stream.name)
    DELETE STRUCTURE NAME(structure_name)
/*
```

For more information, see:

- *z/OS MVS Setting Up a Sysplex* for information about using system logger
- *z/OS MVS JCL Reference* for information about JCL

Actions to avoid

You should avoid the following actions because they will cause data to be lost from RRS's log streams. When data is lost from the RM Data log stream, you must cold start RRS. When data is lost from one or more of the other RRS log streams, you might need to cold start RRS. An RRS cold start is usually very disruptive.

Do not Power-on-reset and IPL without cancelling RRS

Cancel RRS before performing a power-on-reset and IPL. Details follow:

Make sure that RRS disconnects from its log streams before shutting down the z/OS images and performing a power-on-reset of the coupling facility and z/OS CECs. This action will prevent loss of data, and will prevent message ATR212I RRS DETECTED LOG DATA LOSS at IPL time. When you are shutting down your system, first bring down all the resource managers, then issue the SETRRS CANCEL command to bring RRS down. When RRS is cancelled, RRS will disconnect from its log streams. When RRS disconnects, logger will copy the contents of the log streams to offload data sets. This preserves the data in the RRS log streams.

Do not delete offload datasets

Do not delete any offload datasets that contain data from RRS's logs. RRS's offload datasets have names that begin like this:

<high level qualifier>.ATR.<logging group name>.RM.DATA...

<high level qualifier>.ATR.<logging group name>.ARCHIVE...

<high level qualifier>.ATR.<logging group name>.DELAYED.UR...

<high level qualifier>.ATR.<logging group name>.MAIN.UR...

<high level qualifier>.ATR.<logging group name>.RESTART...

To determine what <high level qualifier> is, create JCL to run IXCMIAPU. For example:

```
//REPTLOG   JOB   MSGLEVEL=(1,1)
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSIN   DD
    DATA TYPE (LOGR)
    REPORT (YES)
/*
```

To determine what <logging group name> is, use the RRS panels, option 6, " Display RRS-related system information".

Use only VSAM shareoptions(3,3) for log stream datasets and staging datasets

If you have multiple systems in the sysplex, it is typical for system logger to require access to log stream data sets and staging data sets from multiple systems. For this reason, you must specify VSAM SHAREOPTIONS(3,3) for log stream data sets and staging data sets. See the topic on Planning for System Logger Applications in *z/OS MVS Setting Up a Sysplex*.

Use only Retpd(0) and Autodelete (No)

When you define the RRS log streams, use only AUTODELETE(NO) and RETPD(0) for all RRS log streams except the archive log. If you fail to do this, some offload data sets might be automatically deleted even though they still contain data RRS needs. When data is lost from the RM Data log stream, you must cold start RRS. See the topic on Add Information about Log Streams and Coupling Facility Structures to the LOGR Policy in *z/OS MVS Setting Up a Sysplex*.

RRS use of XCF

RRS uses a single XCF group, called ATRRRS, to communicate between images in a sysplex. No special processing is required by an installation to enable RRS usage of XCF. You do not need to modify XCF transport classes. See *z/OS MVS Setting Up a Sysplex* for more information.

Should the need arise where a z/OS V1R10 system needs to fall back to a lower release, a fallback toleration APAR, number OA23153, should be installed to allow the lower level of RRS to start and preserve the Archive Logging preference from the SETRRS command. Without this APAR on the lower level system, RRS will not be able to start on that system. This will be identified with the messages:

ATR235I RRS FAILED TO JOIN THE RRS XCF GROUP. RC = 00000008, RSN = 00000010

ASA2013I RRS INITIALIZATION FAILED. COMPONENT ID=SCRRS

In this case, RRS must be removed from the XCF group on the lower level system to allow the RRS restart. This can be done by the following steps:

Steps to Remove RRS from XCF:

Note: When completed, the Archive Logging Enable/Disable setting on the V1R10 system will be deleted.

1. Set up the proper access authorization to the Facility Class Resource MVSADMIN.XCF.IXCM2DEL. If your installation uses the RACF® component of SecureWay for z/OS, this can be done from an authorized userid using the following commands:

   RDEFINE FACILITY (MVSADMIN.XCF.IXCM2DEL) UACC(ALTER)

   SETROPTS RACLIST (FACILITY) REFRESH

2. Code up, submit, and review the output from the following JCL sample. The source for the JCL can be found in SYS1.SAMPLIB member IXCDELUT and more details can be found in the Deletion Utility for XCF Group Members section in *z/OS MVS Setting Up a Sysplex*.

   ```
   //IXCDELUT JOB
   //S1  EXEC  PGM=IXCM2DEL,PARM='ATRRRS,mem01'
   //SYSPRINT  DD   SYSOUT=A
   ```

   Where:

   ```
   mem01 is the member name of the member to be deleted.
   ```

   mem01 can be determined by issuing system command:

   ```
   D  XCF,GROUP,ATRRRS
   ```

   which results in message:

   ```
    IXC332I  09.49.44  DISPLAY XCF
    GROUP ATRRRS:      SY1               SY2
   ```

   Using the above display as an example, mem01 should be replaced with either SY1 or SY2 depending on which system RRS is being removed from.

3. Upon successful completion of the JCL job, RRS will now be able to start on the lower level system.

Starting RRS

Once you have set address space priority, provided the statement in IEFSSNxx, and know that system logger is active, you can start RRS with the following operator command:

START RRS

The start can be a warm start or a cold start.

Note: Do not try to start RRS from the IEACMD00 parmlib member; programs RRS depends on have not been started.

Warm start

In a warm start, RRS can complete work that was in progress when a previous RRS instance failed or was intentionally stopped. A warm start occurs when all of the RRS logs are intact and available to the restarting RRS instance.

To enable a warm start, enter the SETRRS CANCEL command, or the FORCE command with ARMRESTART, to stop RRS. Then enter a START command, specifying the name of the RRS cataloged procedure in the SYS1.PROCLIB system library.

A warm start also occurs when RRS is started on any system in a sysplex using the same logging group after the first. In effect, any attempt to start RRS when its logs are not empty is a warm start.

Note: While you can warm start RRS as long as the resource manager data log is intact, a warm start after damage to other logs generally causes loss of data about incomplete transactions.

Cold start

In contrast, a cold start occurs when the RRS RM data log is empty. The main and delayed logs are flushed to the archive log if they are not empty during an RRS cold start. If the RM.METADATA log is defined it will also be cleared. An RRS cold start is a sysplex-wide operation, affecting all RRS subsystems using the logging group. In a cold start, RRS cannot complete any work that was in progress; the RRS logs are not available.

To enable a cold start, do the following:

1. Route a SETRRS CANCEL command to all systems in the sysplex where RRS is active and using the affected logging group.
2. Decide whether to save the contents of the logs. To save the contents, which is the safer choice, choose a log group name that is different from the one previously in use. If there is no need to save any data in the logs, use the same log group name.
3. Specifying the log group name you need, use the IXCMIAPU utility to delete and redefine the RRS resource manager data log. The ATRCOLD member of SYS1.SAMPLIB contains sample JCL to invoke IXCMIAPU to delete and redefine the RRS resource manager data log.

   Figure 18 on page 547 shows the sample JCL. Replace the highlighted field with the sysplex name or the name of an RRS log group within the sysplex.

   Note: Do not run IXCMIAPU when RRS is active; stop RRS before you run the utility.
4. Start RRS on one system.

In response, the first RRS that starts initiates a sysplex-wide cold start applied to all RRS subsystems using the affected logging group. This cold start does the following:

- Moves logged information about incomplete URs to the RRS archive log. Later, you can use ISPF panels to browse the RRS archive log to see the incomplete URs.

- Deletes the contents of the RRS main UR state log, RRS delayed UR state log, and RRS restart log (and if being used, the RRS RM MetaDataLog).

Once RRS has been started for the first time, do not use the IXCMIAPU utility to change any logs other than the RRS resource manager data log. Such action might cause serious database inconsistencies that require manual verification and updating.

If, however, you want to force a cold start and keep the data in the existing logs, specify a different log group name in the JCL that invokes IXCMIAPU, as described for step 3 in the preceding list.

```
//ATRCOLD  JOB  MSGLEVEL=(1,1)
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
    DATA TYPE(LOGR)
    DELETE LOGSTREAM NAME(ATR.RRSGROUP.RM.DATA)
/*
//       IF (STEP1.RC = 0) THEN
//STEP2    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
    DATA TYPE(LOGR)
    DEFINE LOGSTREAM NAME(ATR.RRSGROUP.RM.DATA) STRUCTNAME(LIST01)
           LS_DATACLAS(VSAMLS)
/*
//       ENDIF
```

Figure 18. Sample JCL for IXCMIAPU

Internal Cold Start

Internal cold start processing is designed to eliminate the sysplex wide outage when certain problems are detected with the RM Data log. Normally the problem is corrected by cancelling RRS in the entire sysplex, request a cold start of RRS using the ATRCOLD procedure, and then restart RRS on each system in the sysplex. Internal cold start will try to resolve the problem without the outage.

Internal cold start processing, assuming there are no errors, is done by:

1. RM DATA log problem identified by message:

   ```
   ATR212I RRS DETECTED LOG DATA LOSS ON LOGSTREAM logstreamname DUE TO
           INACCESSIBLE LOG DATA. LOG DATA FROM lowGMT TO highGMT ARE AFFECTED.
   ```

2. The operator can decide to do an Internal Cold Start by replying COLDSTART to message:

   ```
   ATR250E RRS LOGSTREAM ERROR FOUND. CORRECT THE ERROR OR OPTIONALLY REPLY
           COLDSTART TO BEGIN A RRS INTERNAL COLD START.
   ```

   This message is only issued if all the systems in the sysplex support Internal Cold Start at the z/OS V2R1 or higher level.

When a resource manager (RM) registers with RRS on a particular system in the sysplex, an in-storage copy of the RM is created as well as an entry in the RM Data log. The RM Data log contains active/inactive RMs such that if something happened to RRS, the RMs involvement with RRS could be recreated from the log. Internal cold start is just the opposite, if something happened to the RM Data log, the log could be recreated from the in-storage copy of each RM on every system in the sysplex.

The log can only be recreated if the in-storage RMs remain viable and RRS on all the systems in the sysplex remain active. If anything fails during an Internal Cold Start, the ability to recreate the log from the in-storage RMs is not possible. For that reason, RRS will be terminated for any error forcing the operator to do a ATRCOLD procedure, and then restart RRS on each system in the sysplex. Once an internal cold start is started, the termination of one RRS image in the sysplex will cause all other images to terminate. The termination is identified by an MVS/RRS TERMINATION DUMP, 5C4, with REASON xxxx0029.

After an Internal Cold Start, the log data is marked for deletion. System logger does not physically delete the log data or log data set until an offload requires a new data set to be allocated. This means that there will often be a delay before eligible log data sets are physically deleted, since offloading will not occur until the high threshold associated with the log stream is reached. If off-loads are relatively infrequent, then there may be a considerable delay before log data sets that are eligible for deletion are actually deleted.

Stopping RRS

Recommendation: Bring down all applications and RMs that utilize RRS services prior to cancelling RRS. This will minimize the amount of manual intervention required when you restart the applications and RMs.

You can stop RRS with the following operator commands:

SETRRS CANCEL

SETRRS SHUTDOWN

Issuing SETRRS CANCEL with non-resource manager programs in syncpoint may result in a X'058' abend. If the abend occurs, transactions that were in progress will be resolved when RRS restarts.

Issuing SETRRS SHUTDOWN provides a normal shutdown command to bring down RRS without resulting in a *X'058*' abend or *X'0D6*' abend. In order to notify RRS resource managers that RRS is terminating, all the currently active resource managers will be unset. After the unset processing is completed, the RRS jobstep task and all of its subtasks will normally be terminated to clean up the address space. In addition to the RRS infrastructure tasks, there are also timed process tasks and server tasks running in the RRS address space. These tasks will also be shut down normally as well.

If SETRRS CANCEL or SETRRS SHUTDOWN does not stop RRS, you can use the FORCE RRS,ARM command. In this command, RRS is the subsystem name your installation assigned to RRS in parmlib member IEFSSNxx.

For information about the operator commands, see *z/OS MVS System Commands*. For information about the IEFSSNxx parmlib member, see *z/OS MVS Initialization and Tuning Reference*.

RRS should be active on an MVS system that has programs involved in resource recovery. In a sysplex, RRS should be active on every MVS system image that might take part in distributed resource recovery. Only one instance of RRS can be active on each system image.

Usually RRS should be active all the time. However, you should issue a SETRRS CANCEL or SETRRS SHUTDOWN command prior to a system IPL. Cancelling

RRS before an IPL will result in a cleaner system recovery. Use a START ATRRRS command in the COMMNDxx parmlib member to start RRS during system initialization. If RRS fails, it can restart; see "Defining RRS to automatic restart management (ARM)" on page 537.

Using the SETRRS ARCHIVELOGGING [DISABLE | ENABLE] command

Use the SETRRS ARCHIVELOGGING command to disable or enable RRS archive logging on a given system.

For a complete description of the SETRRS ARCHIVELOGGING command, see *z/OS MVS System Commands*.

Using the SETRRS CANCEL command

Use the SETRRS CANCEL command to end RRS abnormally. Use this command only at the direction of the system programmer. Normally, you will not use this command, because RRS should be running at all times; stopping RRS can cause application programs to abend or wait until RRS is restarted.

For a complete description of the SETRRS CANCEL command, see *z/OS MVS System Commands*.

Using the SETRRS SHUTDOWN command

Use the SETRRS SHUTDOWN command to end RRS normally. Use this command only at the direction of the system programmer. Normally, you will not use this command, because RRS should be running at all times; stopping RRS can cause application programs to abend or wait until RRS is restarted.

For a complete description of the SETRRS SHUTDOWN command, see *z/OS MVS System Commands*.

Using the DISPLAY RRS command

Use the DISPLAY RRS command to display status information about RRS coordinated transactions to the system console and SYSLOG.

For a complete description of the DISPLAY RRS command, see *z/OS MVS System Commands*.

Collecting problem data

RRS provides Interactive System Productivity Facility (ISPF) panels to allow installation people, such as database administrators or system programmers, to work with RRS. These are described in Chapter 10, "Using RRS panels," on page 565. In addition, you can write a batch job to take a snapshot of the contents of RRS's logs. The ATRBATCH member of SYS1.SAMPLIB contains sample JCL to do this. Refer to the sample JCL for more details. You will need to modify this sample before you run it.

RRS provides a DISPLAY RRS command, which is described in *z/OS MVS System Commands*.. In addition, you can write a batch job to issue RRS display commands and place the output in a dataset. The ATRBDISP member of SYS1.SAMPLIB contains sample JCL to do this. Refer to the sample JCL for more details. You will need to modify this sample before you run it.

Recovering from a hung UR after an SDSRM failure

When an SDSRM terminates while it is interested in a UR whose state is *indoubt*, certain control structures must be destroyed. These control structures are rebuilt before the RMs needs them. However, the control structures are rebuilt when each RM interested in the *indoubt* UR restarts. Thus, the *indoubt* UR is not allowed to progress until all the RMs interested in the *indoubt* UR have restarted.

If an SDSRM fails and restarts, and you then notice that a UR is not progressing and the SDSRM is interested in the UR, then use the RRS panels to investigate the UR. If the RRS panels indicate that the UR's state is *indoubt*, the type is *PROT*, and the comments are *DX*, then restarting all the RMs interested in the *indoubt* UR may allow the UR to progress.

Latch identification

RRS uses numerous latches to establish an orderly flow during transaction processing. If a unit of work hangs up while holding a latch, other work units may also wait for the latch causing a hung transaction. The Display GRS Contention (D GRS,C) command, will establish there is contention but does not provide sufficient information to quickly narrow the problem down to a particular transaction. If contention persists, the command: D GRS,ANALYZE,LATCH,DEPENDENCY,DETAIL will display more information along with the latch identifiers that have been established for some of the RRS latches. For the RRS address space, the following Latch Identifiers have been created.

Table 49. Latch Identifiers

Latch set	**Latch identifier**	**Identifier description**	**Identify the transaction by RRS panel option:**
CD Context Data	CD URID: *urid*	urid - Unit of Recovery Identifier	3 - Display/Update RRS Unit of Recovery information and search on the urid.
RM Resource Manager	RM: *rmname*	rmname – Resource Manager Name	2 - Display/Update RRS related Resource Manager information and search on the rmname.
SHT System Hash Table	SHT System: *sysname*	sysname – name of the system that owns the SHT. See note below.	If SHT contention persists, contact your IBM Support Center.
SHT System Bucket Hash Table	SHT Bucket: *number* System: *sysname*	number – bucket number. sysname – name of the system that owns the SHT. See note below.	If SHT Bucket contention persists, contact your IBM Support Center.
SHE Sysplex Hash Element	SHE SURID: *surid*	surid - Sysplex Unit of Recovery Identifier	3 - Display/Update RRS Unit of Recovery information and search on the surid.

Table 49. Latch Identifiers (continued)

Latch set	Latch identifier	Identifier description	Identify the transaction by RRS panel option:
UR Units of Recovery	UR URID: *urid*	urid - Unit of Recovery Identifier	3 - Display/Update RRS Unit of Recovery information and search on the urid.
UR Main State Log	MainQ Log: *log*	log – UR Log Stream Name	If State Log contention persists, contact your IBM Support Center.
UR Compression State Log	CompQ Log: *log*	log – UR Log Stream Name	If State Log contention persists, contact your IBM Support Center.

Note: For the SHT and SHT Bucket Latch Sets, "Unknown" might be displayed for the system name. The Unknown SHT table keeps track of transactions that are moving between systems sometimes because of system restart problems.

As an alternative to identify the transaction by the RRS panels, the DISPLAY RRS command (D RRS,RM or D RRS,UR) can be used for CD, RM, and UR latch sets.

The following output is an example from command: D GRS,ANALYZE,LATCH,DEPENDENCY,DETAIL

```
SY1  ISG374I 16.25.10 GRS ANALYSIS 220
   DEPENDENCY ANALYSIS: ENTIRE SYSTEM
   ----- LONG WAITER #1
            JOBNAME: RRS       (ASID=002A, TCB=004E03E8)
            REQUEST: EXCLUSIVE                      LT:7EA8907800000007
   WAITING  00:59:17 FOR RESOURCE (CREATOR ASID=002A)
   SYS.ATRURCPO.00000001                          LST:7EA8C10000000124
   5: UR URID:C30C9DB97E0AC0000000000201020000
            JOBNAME: MAINASID  (ASID=0030, TCB=004E6D90)
            REQUEST: EXCLUSIVE                      LT:7EA8B010000000A2
   ANALYSIS ENDED: THIS UNIT OF WORK IS NOT WAITING
```

In the example, the Latch Identifier is:
UR URID:C30C9DB97E0AC0000000000201020000.

For more information about the DISPLAY RRS and DISPLAY GRS commands, see *z/OS MVS System Commands*.

RRS SDUMP exit

RRS uses the IEASDUMP.SERVER exit to add RRS information to a dump when a resource manager is registered with RRS or an application is running an RRS syncpoint service is dumped as a result of an operator issued console dump command or SLIP SVC dump command. RRS must be active at the time of the dump.

The exit is added during RRS initialization and removed during RRS termination.

Chapter 9. RRS application programming

Working with application programs

RRS provides an application programming interface (API) consisting of two callable services:

- Application_Commit_UR
- Application_Backout_UR

These callable services are described in *z/OS MVS Programming: Callable Services for High-Level Languages*, including a description of the return codes intended for the application programmer. To provide additional information a system programmer or data base administrator might need to help the application programmer, the following table explains conditions that cause RRS to issue each return code.

Table 50. Application program conditions that cause RRS to issue return codes

Return Code in: **Hexadecimal** **Decimal** **Equate Symbol**	**Conditions**
0 0 RR_OK	One of the following: • No resource manager expressed interest in the UR. • The collective vote from the PREPARE exit routines is FORGET. • The collective vote from the PREPARE exit routines is COMMIT. No COMMIT exit routine returned ATRX_HM, ATRX_HR, or ATRX_OK_OUTCOME_PENDING. No resource manager became unregistered or unset with an incomplete protected interest in the UR. • The return code from the ONLY_AGENT exit routine, if invoked, is ATRX_OK.
65 101 RR_COMMITTED_OUTCOME_PENDING	One of the following: • The collective vote from the PREPARE exit routines is COMMIT. No COMMIT exit routine returned ATRX_HM or ATRX_HR. However, at least one COMMIT exit routine returned ATRX_OK_OUTCOME_PENDING or a resource manager became unregistered or unset with an incomplete protected interest in the UR. • The return code from the ONLY_AGENT exit routine, if invoked, is ATRX_OK_OUTCOME_PENDING.

Table 50. Application program conditions that cause RRS to issue return codes (continued)

Return Code in: **Hexadecimal** **Decimal** **Equate Symbol**	**Conditions**
66 102 RR_COMMITTED_OUTCOME_ MIXED	The collective vote from the PREPARE exit routines is COMMIT. However, one of the following is true: • At least one COMMIT exit routine set a heuristic mix (HM) or heuristic reset (HR) return code. • The resource manager called Set_Side_Information and set heuristic mix (HM). • A DISTRIBUTED_SYNCPOINT exit routine set a heuristic mix (HM) return code.
C8 200 RR_PROGRAM_STATE_CHECK	A STATE_CHECK exit routine returned an ATRX_STATE_INCORRECT code. None of the STATE_CHECK exit routines returned an ATRX_REDRIVE code.
12C 300 RR_BACKED_OUT	One of the following: • The collective vote from the PREPARE exit routines is BACKOUT. No BACKOUT exit routine returned ATRX_HM, ATRX_HC, or ATRX_OK_OUTCOME_PENDING. No resource manager became unregistered or unset with an incomplete protected interest in the UR. • The return code from the ONLY_AGENT exit routine, if invoked, is ATRX_BACKOUT.
12D 301 RR_BACKED_OUT_OUTCOME_ PENDING	One of the following: • The collective vote from the PREPARE exit routines is BACKOUT. No BACKOUT exit routine returned ATRX_HM or ATRX_HC. However, at least one BACKOUT exit routine returned ATRX_OK_OUTCOME_PENDING or a resource manager became unregistered or unset with an incomplete protected interest in the UR. • The return code from the ONLY_AGENT exit routine, if invoked, is ATRX_BACKOUT_OUTCOME_PENDING.

Table 50. Application program conditions that cause RRS to issue return codes (continued)

Return Code in: **Hexadecimal** **Decimal** **Equate Symbol**	**Conditions**
12E 302 RR_BACKED_OUT_OUTCOME_ MIXED	One of the following: • The collective vote from the PREPARE exit routines is HM. • The collective vote from the PREPARE exit routines is BACKOUT. At least one BACKOUT exit routine returned ATRX_HM or ATRX_HC. • The return code from the ONLY_AGENT exit routine, if invoked, is ATRX_HM. • The resource manager called Set_Side_Information and set heuristic mix (HM). • A DISTRIBUTED_SYNCPOINT exit routine set a heuristic mix (HM) return code.
Note: In COBOL, the equate symbols are truncated at 30 characters. In PL/I, the equate symbols are truncated at 31 characters.	

During syncpoint operations, RRS default actions are to commit on normal context termination and backout on abnormal context termination.

If RRS fails during a syncpoint operation, the application terminates abnormally. If RRS fails before the application issues a commit or backout, RRS ensures that the application will receive an OUTCOME_PENDING return code for each incomplete UR. However, for an **in-doubt** UR, RRS does not issue a return code. Later, if RRS restarts without a system reIPL and the application is still active when the **In-doubt** UR is resolved, RRS issues a return code to the application at that time.

When an application ends abnormally during syncpoint processing, and the ABEND is caused by an outside source, such as the CANCEL command, the condition is called an asynchronous ABEND. The application needs to consider the following points related to an asynchronous ABEND:

- If the application encounters an asynchronous ABEND during processing of a backout request, the application can, on restart, retry the backout request.
- If the application encounters an asynchronous ABEND during processing of a commit request, the application receives, on restart, no indication of the outcome for the UR. To continue processing, the application should retry the commit request that failed rather than trying to back out the UR. Retrying the commit request can cause any of the following:
  - RRS might commit the original UR.
  - RRS might commit an empty UR, which is a new UR with no changed resources.
  - An X'5C4' abend might occur.

  If the attempt to retry the commit request succeeds (the service return code is ATR_OK), the application cannot assume that the original commit request succeeded. The outcome of the original request is unknown. All the application can assume from the ATR_OK return code is that the context and the current UR are consistent and that normal syncpoint operations can continue.

- If a resource manager restarts, it can obtain its failed protected interests unless the syncpoints completed successfully under the RRS server task.

Working with cascaded transactions

Cascaded transactions affect the way applications and work managers must operate. See "Cascaded transactions" on page 69 for more information about cascaded transactions. Cascaded transactions should not have any effect on resource managers that are not work managers. The following topics describe issues an application programmer or system programmer should be aware of when working with cascaded transactions in:

- Application programs
- Work managers

Application rules

Applications that work with cascaded transactions have special requirements.

Initiating syncpoints

Like the protocols of Transactional Remote Procedure Calls (TRPC), when a transaction consists of multiple URs linked together to form a cascaded UR family, only the application running under the top-level UR can validly request commit to be initiated. The application can use either the Application_Commit_UR service or the Commit_UR service to do so.

If a resource manager has taken the SDSRM role on the top-level UR, only that SDSRM may initiate commit processing for the cascaded UR family. It does so with the Prepare_Agent_UR service.

All of the application pieces running under cascaded URs would normally complete their processing, return the results to the application that initiated them, and then return to their work manager. The work manager would then issue a Set_Side_Information call for the cascaded UR, indicating that it is complete. When the top-level UR initiates commit processing, all of the URs will be committed or backed out as a single atomic transaction.

Any piece of the transaction can validly initiate a backout operation at any time. Initiating a backout immediately backs out that piece of the overall transaction, and causes the overall transaction to be backed out eventually.

Application controlled parallelism

An application program can use cascaded transactions to enable transaction parallelism. While RRMS allows you to do this, use the capability with care. Many work managers expect to have control over all of the tasks and contexts in their address space. Work managers may not work correctly if an application is attempting to create transaction parallelism through RRMS services without their knowledge.

For an unauthorized application to parallel itself, it needs to do the following:

1. Register as an unauthorized resource manager with Registration Services by calling Register_Resource_Manager.
2. Identify itself to Context Services by calling Set_Exit_Information.
3. Obtain a private context by calling Begin_Context.
4. Create a cascaded UR associated with the private context by calling Create_Cascaded_UR.

This call will make the initiating UR (probably the one associated with the current task) the top-level UR of the cascaded UR family.

5. Create a task (TCB) by calling the MVS macro, ATTACH.

   The parallel thread will execute under this task.

6. While running under the new task, switch the private context to the current TCB by calling Switch_Context.
7. Repeat steps 3–6 to create each parallel thread.

When each task completes its processing, the work manager should mark the UR on the context switched to that task as application complete. You may also have the work manager switch the work context off of that task. When all of the parallel threads have completed their processing, the application running under the top-level UR can initiate the commit of the transaction.

Cascaded URs and database locking

Some databases will not recognize that the separate URs that make up a cascaded UR family are part of the same transaction. Others may require an application to explicitly indicate that it wants to use some form of global transaction locking. When they do not, the database cannot allow the separate pieces of the transaction to share database locks. The separate threads of a transaction executing under the URs of a cascaded UR family must therefore ensure that they do not attempt to access the same data from different URs. Doing so will result in deadlocks which could keep the application from ever completing, or cause the overall transaction to be backed out by the affected resource manager. Remember, transactions are supposed to be isolated from one another. One transaction may not see the changes made by another transaction until the changes have been committed.

Just like a normal cascaded UR, a multisystem cascaded UR may be viewed by a resource manager as being in a different locking scope from other URs in the cascaded transaction. Database managers may not support sharing database locks across URs that are executing on different systems. Therefore, you must ensure that multiple parts of a multisystem cascaded transaction executing on different systems do not attempt to access the same locked resources. If you do not, deadlocks could keep your application from ever completing or cause the overall transaction to be backed out by the affected resource manager.

Work manager guidelines

A work manager is likely to need to use cascaded URs when it is doing one of two things:

- Moving work between different work managers
- Executing parts of an application in parallel

Work managers using cascaded URs must also be aware of when an application should be marked **application-complete**, and when it should be marked **not application-complete**.

Moving work between work managers

When initiating transactions that are combinations of existing transactions, the separate applications that make up the overall transaction may need to execute in different work manager environments. For example, an application executing in Environment A might need to initiate a transactional application that executes in Environment B; but both pieces must be part of the same transactional scope. Both pieces of the application need to run under a single RRS unit of recovery. When

the piece of the application running in Environment A requests that Environment B run the second piece of the transaction, Environment B has two choices:

- Move the current context and UR from Environment A to Environment B.
- Create a cascaded UR in Environment B, which will have a parent UR in Environment A.

Moving the current context can be more efficient, because it does not require that a new context and UR be created. Moving the current context also has the advantage of being more likely to allow the separate pieces of the transaction to share database locks. However, work manager B cannot be sure that it can actually move the context. If the context is a DU native context, or if a work manager with an expression of interest in the context disallows work manager B's switch request, Context Services will not move the context.

Creating a cascaded UR will always work. Neither the type of context being used in Environment A nor any RM associated with it can stop work manager B from creating the cascaded UR. Unfortunately, the separate pieces of the application may not be able to share database locks.

A work manager may attempt to move the current context, but create a cascaded UR if the attempt fails. This can be a reasonably efficient choice, as long as the switch attempt works most of the time. Unfortunately, the application must assume that it cannot share locks, since it has no way of knowing if the work manager will switch the context or create a cascaded UR.

Some work manager interfaces, like IMS via OTMA, are *context input work manager interfaces*. These interfaces require that they be passed a context as input to start any transaction coordinated by RRS. The originating work manager must be aware when an application needs to invoke a context input work manager interface, so that they can first create a cascaded UR to be passed across the interface. IBM recommends work managers avoid creating new context input work manager interfaces.

Parallel processing

A work manager can split a transaction into pieces that execute in parallel, usually at the request of an application program or an installation. If the multiple separate pieces of the divided transaction need to touch protected resources, they may need to have the same transactional scope. To ensure that they do, a work manager could create cascaded URs for the separate pieces of the transaction. “Application controlled parallelism” on page 556 describes how an unauthorized application can use cascaded URs to enable parallelism.

For an authorized work manager to parallel itself, it needs to do the following:

1. Register as an authorized resource manager with Registration Services by calling Register_Resource_Manager.
2. Identify itself to Context Services by calling Set_Exit_Information.
3. Obtain a private context by calling Begin_Context.
4. Create a cascaded UR associated with the private context by calling Create_Cascaded_UR.

   This call will make the initiating UR (probably the one associated with the current task) the top-level UR of the cascaded UR family.
5. Create a task (TCB) by calling the MVS macro, ATTACH.

   The parallel thread will execute under this task.

6. While running under the new task, switch the private context to the current TCB by calling Switch_Context.
7. Repeat steps 3–6 to create each parallel thread.

When each task completes its processing, the work manager should mark the UR on the context switched to that task as application complete. You may also have the work manager switch the work context off of that task. When all of the parallel threads have completed their processing, the work manager can initiate the commit of the transaction.

Application-complete

RRS will not initiate commit processing on a cascaded UR family until all of the cascaded URs in the family have been marked as application-complete. This ensures that RRS will not initiate commit or backout processing while a resource manager is processing an application request.

When RRS initially creates a cascaded UR, it is not application-complete. It is the responsibility of the work manager that created the cascaded UR to tell RRS when it is application-complete by using the Set_Side_Information service. Similarly, if a work manager decides to reuse a cascaded UR after marking it application-complete, it must tell RRS that it is no longer complete by invoking the Set_Side_Information service before allowing the application program to run under the UR. Once it has done this, it must also tell RRS that it is application-complete when the application program is finished.

Managing contexts of cascaded URs

Once a cascaded UR is created, the context of the cascaded UR cannot end until the cascaded UR family has reached the second phase of the two-phase commit (**in-commit** or **in-backout** state), without making all of the transactions in the cascaded UR family back out. Because of this complication, a work manager that creates a cascaded UR associated with a privately managed context must know when the cascaded UR is **forgotten**. The following methods allow a work manager to either find out when the UR is forgotten, or avoid having to end the work context:

- The work manager can use a DU native context and allow the context to end normally when the DU ends.
- The work manager can express interest in the UR. With its END_UR exit, the work manager can schedule an asynchronous request, through its own mechanisms, and end the context under the asynchronous request.

  Note:

  1. This option is not available to unauthorized work managers.
  2. A request to end the context must not be issued from the exit or from any routines it calls or any routines it waits on. Doing so will result in a deadlock as the exit waits for the context to complete, and the context completion waits for the exit to complete.
  3. The work manager does not know when the UR completes using this method. If the work manager wants to allow another thread to reuse the context, one of the other methods must be used to determine when the UR completes.
- The work manager can use the Set_Post_Sync_PET service to have RRS release the PET when the cascaded UR has completed. The work manager can pause on the PET or periodically query the PET to determine when it has been released. Once the cascaded UR has completed, the context can safely be ended.

Note: A UR exit routine should never wait directly or indirectly on the PET. Doing so will result in a deadlock as the exit waits for the UR to complete, and the UR completion waits for the exit to complete.

- If the work manager is responsible for managing the top-level UR, the work manager can safely end the contexts of URs cascaded from the top-level after it commits or backs out the top-level UR or ends the top-level UR's work context.
- The work manager can periodically query the state of the cascaded UR, and end the context when the UR has reached the **forgotten** state.

  If the work manager uses Retrieve_UR_Data to obtain the UR state, it may receive a return code indicating that the specified UR token is not valid. That would indicate that the UR has been forgotten.

Additional multisystem cascaded transaction guidelines

A multisystem cascaded transaction can only span across multiple systems that use the same logging group. A work manager can use the ATRQUERY REQUEST=SYSINFO macro interface to determine which systems are in a particular logging group.

When moving work between systems, a work manager is responsible for transferring all of the data needed for a particular piece of work, including a UR token to provide transactional context. Once the data is transferred, the work manager can create a new work context or use an existing work context to represent the work request locally. When the work manager creates a cascaded transaction via the Create_Cascaded_UR or Express_UR_Interest service specifying the transferred UR token, RRS will recognize that the UR token represents a UR on another system and create a multisystem cascaded transaction.

Just like normal cascaded transactions, the work manager is responsible for informing RRS when the part of the application executing under a multisystem cascaded UR is complete by using the Set_Side_Information service to mark the UR as application-complete.

Since the UR token used to specify the parent of a multisystem cascaded transaction represents a UR on another system, it is possible that the specified UR token is no longer valid. This could occur if the parent transaction, the system it is resident on, the system it is executing on, or the RRS running on that system has failed. If any of those conditions occur, the token will be recognized as invalid and RRS will issue a failure return code to the requester.

Note: The token will **never** again be valid. This could affect a work manager removing a work request from a queue within a syncpoint and then trying to create a cascaded transaction with the information taken from the queue. If the multisystem cascaded transaction cannot be created due to an invalid UR token, the removal of the work request from the queue cannot be rolled back because it would result in an endless loop.

Logging data

RRS hardens information about URs and resource managers in RRS logs. Hardening means storing the information in RRS logs residing on non-volatile external storage that can be accessed during restart after a failure.

RRS hardens information like the following, which a resource manager needs during restart for an incomplete UR:

- The UR state

- The name and role of a resource manager with a protected interest in the UR
- The name of the resource manager's log
- The resource manager's persistent interest data

Table 51 summarizes the events that cause RRS to harden log information. It highlights the additional logging performed for UR interests that have a presumed nothing protocol. Note that all RRS logging is forced unless Table 51 indicates otherwise.

Table 51. Event Logging Summary

UR State or condition	Event	Logging
In-state-check to **in-prepare**	The STATE_CHECK exit routines completed successfully.	RRS logs information only for presumed nothing interests.
In-prepare to **in-commit**	The overall prepare vote is okay, and the syncpoint is local.	RRS logs information for all protected interests.
In-prepare to **in-doubt**	The overall local prepare vote is YES, and the syncpoint is distributed; a resource manager has taken the DSRM or SDSRM role.	RRS logs information for all protected interests.
In-doubt to **in-backout**	The Backout_Agent_UR service tells RRS to back out the UR with a *log_option* of ATR_DEFER_IMPLICIT.	RRS logs information only for presumed nothing interests.
In-doubt to **in-backout**	The Backout_Agent_UR service tells RRS to back out the UR with a *log_option* of ATR_DEFER_EXPLICIT.	RRS logs information for all protected interests. If, on restart, the SDSRM always wants **in-backout** information rather than **in-doubt** for URs resolved to **in-backout**, it should always issue Backout_Agent_UR with a *log_option* of ATR_DEFER_EXPLICIT.
In-doubt to **in-backout**	The Backout_Agent_UR service tells RRS to back out the UR with a *log_option* of ATR_IMMEDIATE.	RRS logs information for all protected interests except the interest of the SDSRM.
In-doubt to **in-backout**	The DISTRIBUTED_SYNCPOINT exit routine tells RRS to back out the UR.	RRS logs information only for presumed nothing interests.
In-doubt to **in-backout**	The installation resolves the **in-doubt** state through a backout command on a panel	RRS logs information for all protected interests.

Table 51. Event Logging Summary (continued)

UR State or condition	Event	Logging
In-doubt to **in-commit**	One of the following occurs: • The SDSRM calls the Commit_Agent_UR service to tell RRS to commit the UR. • The DISTRIBUTED_SYNCPOINT exit routine tells RRS to commit the UR. • The installation resolves the **in-doubt** state through a commit command on a panel.	RRS logs information for all protected interests.
In-forget to **forgotten**	The Forget_Agent_UR_Interest service with a *log_option* of ATR_IMMEDIATE tells RRS to forget the UR.	The SDSRM's expression of interest is deleted from the log.
In-forget to **forgotten**	The UR completes normally.	RRS deletes the log entry for all protected interests. This deletion is an unforced deletion.
Any state	RRS detected a heuristic-mixed outcome.	RRS logs information for all protected interests at the next state change.
All states, when information has already been logged	An RM calls the Set_Persistent_Interest_Data service.	RRS immediately logs updated information for all protected interests.
All states except **in-flight**, **in-state-check**, **in-prepare**	The Set_Side_Information service indicates heuristic mix.	RRS immediately logs information for all protected interests.
All states except **in-only-agent**	The Set_Side_Information service indicates resync-in-progress.	RRS logs information for all protected interests at the next state change.
All states except **in-flight**, **in-state-check**, **in-prepare**, **in-doubt**	The SDSRM calls the Forget_Agent_UR_Interest service with a *log_option* of ATR_IMMEDIATE to tell RRS to forget the UR.	The SDSRM's expression of interest is physically deleted from the log.
Any state	When UR information has been hardened, and the installation uses a panel to remove a resource manager's interest in a UR.	RRS rehardens all protected interests in the UR except the interest being removed.

As Table 51 on page 561 indicates, RRS hardens data only for protected interests in a UR. Data is not hardened for unprotected interests. Other considerations related to hardening are:

- RRS retains hardened information until the resource managers interested in the UR indicate, through return codes from exit routines or through service calls, that they have completed their interests in the UR. At that point, RRS deletes the UR information from its RRS logs.

  Note: Because the hardened information for a UR may not be deleted immediately, a resource manager could see on restart an interest in a UR for

which the resource manager had completed processing. There is no way for a resource manager to force immediate deletion of hardened information, with one exception: an SDSRM that uses a *log-option* of ATR_IMMEDIATE forces immediate deletion of hardened information.

- RRS does not log any information for a UR with only one expression of interest when the resource manager has an ONLY_AGENT exit routine.
- If a resource manager's exit routine passes a FORGET return code, RRS does not log any information for that interest at any subsequent log points.
- For a heuristic-mixed outcome or a resync in progress, RRS hardens, or rehardens, the UR state of all protected interests in the UR, including the heuristic-mixed or resync in progress information, at the next state change.

  If a resync in progress occurs before the UR reaches an **in-prepare** state, RRS defers hardening until the UR reaches the **in-prepare** state.
- If DRIVE_COMPLETION or DRIVE_BACKOUT is set, RRS hardens, or rehardens, at the next logging point, the UR state of all protected interests in the UR.
- When the installation uses a panel to resolve an **in-doubt** state to **in-backout**, RRS hardens all protected interests in the UR. In this way, RRS makes sure that, on restart, a distributed syncpoint resource manager sees an **in-backout** state for a presumed abort interest in a UR. During restart, if a resource manager specifies ATR_RESPOND_CONTINUE in its process interest call for this UR, RRS notifies the resource manager that the installation resolved the **in-doubt** state to **in-backout** by invoking the resource manager's BACKOUT exit routine.
- When the installation uses a panel to remove a resource manager's interest in a UR and RRS had hardened information about the UR, then RRS rehardens the most recent information in the RRS log, minus the interest the installation removed. If the installation removed the last interest in the UR, RRS still rehardens the UR state with no expressions of interest. In this way, RRS makes sure that, on restart, the resource manager does not see the interest that the installation removed.

Logging cascaded transactions

When RRS URs are coordinated under a single transactional scope by distributed protocols that use the distributed syncpoint resource manager (DSRM) or server distributed syncpoint manager (SDSRM) roles, RRS is required to log separate in-doubt records for each of the URs except the initiating one. However, RRS manages all of the URs in a cascaded UR family as if the family was a single UR. There is no need for cascaded URs to go **in-doubt**, or be logged separately. If the top-level UR does not have a DSRM or SDSRM role, RRS can immediately make the commit or backout decision for the UR family and log only those records needed to record the final outcome of the cascaded transaction. If the top-level UR has a RM which has taken the DSRM or SDSRM role, RRS will have to log an **in-doubt** record, but it will still only have to log one **in-doubt** record for the entire cascaded UR family.

Prior to the existence of cascaded transactions, RRS restricted the amount of data that could be logged for a single transaction to a single System Logger log block, 64K of data. While all of the data for a single UR must still fit within a single log block, RRS allows data for cascaded UR families to span multiple log blocks.

Chapter 10. Using RRS panels

RRS provides Interactive System Productivity Facility (ISPF) panels to allow installation people, such as database administrators or system programmers, to work with RRS. In addition, you can:

- Write a batch job to take a snapshot of the contents of RRS's logs. The ATRBATCH member of SYS1.SAMPLIB contains sample JCL to do this. Refer to the sample JCL for more details.
- Use the DISPLAY RRS command to display status information about RRS coordinated transactions to the system console and SYSLOG. For a complete description of the DISPLAY RRS command, see *z/OS MVS System Commands*.
- Use the ATRQSRV Utility to Query and Update RRS Information from JCL jobs. For a complete description of the ATRQSRV utility, see Chapter 13, "ATRQSRV utility - query and update RRS information," on page 643.

When you use the panels, you can view the following information:

- RRS logs
- UR information
- Resource manager information

Through the panels, you can also take the following actions:

- Determine where a resource manager can restart after a system failure
- Resolve an **in-doubt** state for a UR to **in-commit** or **in-backout**
- Remove a resource manager's interest in a UR
- Delete a resource manager from RRS
- Unregister a resource manager to clean up the resource manager's involvement with RRS.

Thus, the panels provide a way for you to troubleshoot resource recovery. You might use them, for example, if an application, is hung up and you suspect that resource recovery might be the cause of the problem.

Before you can use the panels, however, you must set up access authorization, allocate the libraries containing the panels, and add the RRS application to the ISPF primary option menu.

Setting up access authorization

If your installation uses the RACF component of SecureWay for z/OS, you can control access to the information and actions the panels provide. In a Parallel Sysplex®, you can configure RRS to allow a user to manage all the RRS images in the sysplex from a single image. Access to RRS system management functions is controlled by two RACF resources.

To control RRS access across a sysplex, RRS uses the MVSADMIN.RRS.COMMANDS.*gname.sysname* resource in the FACILITY class, where *gname* is the logging group name, and *sysname* is the system name. You may create a RACF profile to permit access to multiple logging groups and systems by including RACF valid generic characters (**, *, and %) in *gname* and *sysname*. See the *z/OS Security Server RACF Security Administrator's Guide* and *z/OS Security*

Server RACF Command Language Reference for more information about using these RACF generic characters and defining RACF profiles. By permitting appropriate access, you can allow users to view or alter RRS information on any number of systems in the sysplex.

If you are running RRS on a single system, RRS can use either the MVSADMIN.RRS.COMMANDS.*gname.sysname* resource or the MVSADMIN.RRS.COMMANDS resource in the FACILITY class to control access to RRS system management functions. The MVSADMIN.RRS.COMMANDS resource only allows access to RRS system management functions on the current system. You cannot use MVSADMIN.RRS.COMMANDS to allow or disallow use of RRS on another system.

Note: This restriction does not apply to shared restart or RRS log stream data being used by the local system. Access to the log stream datasets requires the appropriate authorization for the system logger address space to the *hlq.data_set_name* resource in the DATASET class for each DASD log stream and staging data set. Use the MVSADMIN.RRS.COMMANDS.*gname.sysname* resource to control access to use RRS services to view or modify information in the logs, including the restart log, of logging groups that are not being used by the local system.

For example:

- To allow a user to view RRS information only on the current system, you could provide READ access to the MVSADMIN.RRS.COMMANDS resource in the FACILITY class.
- Provide ALTER access to the MVSADMIN.RRS.COMMANDS.gname.sysname resource in the FACILITY class for a particular system in the sysplex, to allow a user to:
  - Resolve an **in-doubt** UR
  - Remove a resource manager's interest in a UR
  - Delete a resource manager from RRS
  - Unregister a resource manager to clean up the resource manager's involvement with RRS.

Allocating the RRS panel libraries

Before you use RRS panels, you need to allocate the libraries where the panels are stored as part of the DD specified in Table 52.

Note: These libraries are not specific to RRS, but are used for all of MVS dialogues (for example see IPCS dialogues).

Table 52. RRS Panel Libraries

Library	DD
SYS1.SBLSPNL0	ISPPLIB
SYS1.SBLSTBL0	ISPTLIB
SYS1.SBLSMSG0	ISPMLIB
SYS1.SBLSCLI0	SYSEXEC

Adding RRS as an ISPF menu option

To add the RRS panels as an option on your ISPF primary panel, you should make a copy of the ISPF primary option menu — ISR@PRIM. Then, add the following information to the processing section of the panel:

```
RRS,'PANEL(ATRFPCMN) NEWAPPL(RRSP)'
```

Make sure you concatenate the library containing your customized primary panel before any others in your logon procedure or CLIST.

The following example shows the two lines added to a copy of the ISPF primary panel. The two lines are highlighted in the example.

```
------------------------  ISPF/PDF PRIMARY OPTION MENU  ----------------
%OPTION  ===>_ZCMD
%                                                          +USERID    -
%  0 +ISPF PARMS  - Specify terminal and user parameters   +TIME      -
%  1 +BROWSE      - Display source data or output listings +TERMINAL  -
%  2 +EDIT        - Create or change source data           +PF KEYS   -
%  3 +UTILITIES   - Perform utility functions
%  4 +FOREGROUND  - Invoke language processors in foreground
%  5 +BATCH       - Submit job for language processing
%  6 +COMMAND     - Enter TSO/E command or CLIST
%  7 +DIALOG TEST - Perform dialog testing
%  8 +LM UTILITIES- Perform library management utility functions
%  C +CHANGES     - Display summary of changes for this release
%  R +RRS         - RRS resource recovery information
%  T +TUTORIAL    - Display information about ISPF/PDF
%  X +EXIT        - Terminate ISPF using log and list defaults
%
+Enter%END+command to terminate ISPF.
%
)INIT
  .HELP = ISR00003
  &ZPRIM = YES        /* ALWAYS A PRIMARY OPTION MENU      */
  &ZHTOP = ISR00003   /* TUTORIAL TABLE OF CONTENTS        */
  &ZHINDEX = ISR91000 /* TUTORIAL INDEX - 1ST PAGE         */
  VPUT (ZHTOP,ZHINDEX) PROFILE
)PROC
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
                0,'PANEL(ISPOPTA)'
                1,'PGM(ISRBRO) PARM(ISRBRO01)'
                2,'PGM(ISREDIT) PARM(P,ISREDM01)'
                3,'PANEL(ISRUTIL)'
                4,'PANEL(ISRFPA)'
                5,'PGM(ISRJB1) PARM(ISRJPA) NOCHECK'
                6,'PGM(ISRPTC)'
                7,'PGM(ISRYXDR) NOCHECK'
                8,'PANEL(ISRLPRIM)'
                C,'PGM(ISPTUTOR) PARM(ISR00005)'
                R,'PANEL(ATRFPCMN) NEWAPPL(RRSP)'
                T,'PGM(ISPTUTOR) PARM(ISR00000)'
              ' ',' '
                X,'EXIT'
                *,'?' )
  &ZTRAIL. = .TRAIL
)END
```

Figure 19. Example: Adding RRS as an Option on your ISPF Menu

There is an alternative way to access the RRS panels from the ISPF Main Menu. You can achieve this method by doing the following steps:

1. Go to your ISPF Main Menu and choose option 7.1 to perform dialog testing:
   ```
   Option ===> 7.1
   ```
2. After the ISPF Dialogue Test Menu is displayed, you can invoke the selection panel by doing the following steps:
   a. Put atrfpcmn in the panel option.
   b. Use atrk as the ID.
   c. Press ENTER to display the RRS main panel.
   ```
   PANEL  . . atrfpcmn
   ```

   Note: This panel is longer than what you see on the screen, make sure to scroll down to enter the ID.
   ```
   ID . . . atrk
   ```

Using the main selection panel

When you select RRS from the ISPF menu, the system displays the main selection panel, shown in Figure 20.

```
                                  RRS
Option ===> ____________________________________________________________

Select an option and press ENTER:

0 Set RRS global panel options
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
```

Figure 20. Main Selection Panel (ATRFPCMN)

To continue, you must select an option from this menu. Use the following table as a guide for selecting the appropriate option:

Table 53. Summary of main selection panel options

Option number	Function	Additional information
0	Set or change global options that control the processing throughout various RRS panels.	"Specifying global options" on page 570
1	• Check logs for resource recovery actions related to a resource that may have been damaged • Determine from which system a particular resource manager can restart	"Checking the log streams" on page 570

Table 53. Summary of main selection panel options (continued)

Option number	Function	Additional information
2	• Identify resource managers that are known to RRS • Investigate a problem related to a specific resource manager	"Working with resource manager information" on page 577
3	Work directly with one or more known units of recovery (URs).	"Working with UR information" on page 579
4	• Identify work managers that are known to RRS • Investigate a problem related to a specific work manager	"Working with work manager information" on page 590
5	View or change criteria used to select URs	"Working with UR selection profiles" on page 583
6	View information about system names and RRS logging groups across a Parallel Sysplex.	"Working with RRS system information" on page 593

Help: For additional information about using the panels, press PF1 from any panel.

Messages: While you are working with the panels, RRS can issue messages; these messages appear in *z/OS MVS System Messages, Vol 3 (ASB-BPX)*.

Using wildcards in RRS panels

RRS allows you to use wildcard characters in strings to specify:

- Resource manager names (RMNAME)
- Work manager names (WMNAME)
- RRS logging group names (GNAME)
- System names (SYSNAME)
- Logical unit of work identifiers (LUWIDSTR)
- Global transaction identifiers (GTIDSTR)
- Unit of recovery identifiers (URIDSTR)
- Sysplex unit of recovery identifiers (SURID) (On UR panels)

The character string you specify can contain two types of wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character. Using wildcards allows you to quickly and conveniently obtain sets of similar or related pieces of RRS information. See Chapter 11, "ATRQUERY — Obtain RRS Information," on page 595 for more information about the ATRQUERY parameters that allow wildcards.

Examples using wildcards:

- Work manager or resource manager name: `ATR.*.IBM`
- Logical unit of work identifier: `A?CD.DEF*,*,*`
- Global transaction identifier: `003?234*`

Specifying global options

Many RRS panels use globally set options to control processing. Use the "RRS Global Panel Options" panel shown in Figure 21 to set or change options. The following option is available:

Time Display Format: Some RRS panels display a timestamp. You can use this option to indicate how the timestamp is displayed by selecting either GMT time or LOCAL time. The default setting is GMT. The change of the time format only affects the "Display/Update RRS Unit of Recovery information" option. See Figure 29 on page 581 for reference.

Note: To keep consistency with the existing log stream browse option, the time displayed in the "RRS Log Stream Browse Selection" panel as shown in Figure 22 on page 571 is still in the local format. It is not affected by the selection on the "RRS Global Panel Options" panel. This is also the case for the log browse function from the ATRQSRV utility and from the ATRBATCH sample JCL.

```
                        RRS Global Panel Options

Time Display Format ................: z    (GMT|Local )
```

Figure 21. RRS Global Panel Options (ATRFPVAR)

Checking the log streams

To check, or browse, the contents of a log stream, you use the panel shown in Figure 22 on page 571. In the panel, you identify the log you want to view, whether you want a summary report or a detailed report, and supply any optional filters that you need to limit the information returned.

If you want to determine on which system a resource manager can restart, select log option 4, the RRS Resource Manager Data Log, and specify the resource manager name. For a description of the output, see “Resource manager entry” on page 576.

On the panel, you can specify only log streams that are currently defined to system logger. You cannot browse a log stream that has been copied and is no longer defined to system logger.

The system displays the sysplex name in the log group name field. The Default Group Name field will be set to the RRS logging group name being used on the system; or, the sysplex name, if RRS is not active on the system. If you want to browse the contents of logs with a different name, type the log group name in the field.

You can also identify the output data set (the data set the system is to use to hold the information you request). If you do not provide a name, the system uses *userid*.ATR.REPORT.

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

Provide selection criteria and press Enter:


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

RRS Group Name . . . ________  Default Group Name: : ________
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  . . . . . .                                (Options 4,5)
  SURID                                               (Options 1,2,3)
```

Figure 22. Log Stream Selection (ATRFPLBS)

Use filters as much as possible; a request without filters might return a huge number of entries. Using filters is especially important if you are running RRS in a Parallel Sysplex. When you select any of the first three logs, you can specify a UR identifier (URID), which limits the search to entries for that UR. When you select option 4 or 5, you can identify a resource manager, which limits the search to entries associated with that resource manager.

With any log, you can use the date and time filters to limit the search to log entries after a certain date or time, log entries before a certain date or time, or log entries that fall within a range of dates and/or times. The following example shows how to define a range of hours within the same day:

```
Entries from.  . . . 1997/03/15  (yyyy/mm/dd)
                     02:00:00 (hh:mm:ss)
through. . . . . . . 1997/03/15  (yyyy/mm/dd)
                     10:00:00 (hh:mm:ss)
```

IBM Recommends that you specify date and time filters to limit the amount of data searched when browsing the RRS Archive log using a URID filter. The URID filter causes the log browser to search the entire section of the RRS Archive log that might contain the URID. Because of the potentially large amount of data in the RRS Archive log, the absence of the date and time filters may result in noticeable delays.

After you have defined your selection and pressed Enter, the system browses the logs for the log group you name and returns the log entries that match the criteria you specified. Each entry block has a header line that includes the name of the system from which the block was written, the date and time when the block was written, and the system logger identifier of the block. The header line has the following format:

```
system  mm/dd/yyyy hh:mm:ss.ssssss  LOCAL         BLOCKID=xxxxxxxxxxxxxxxx
BLOCK=xxxx OF yyyy NEXT BLOCKID=xxxxxxxxxxxxxxxx
```

You can obtain the following types of report entries:

- UR entry, either summary or detail

Available from the RRS archive log stream, the RRS unit of recovery (UR) state log stream, and the RRS restart log stream. See "UR entry."

- Archive entry, either summary or detail

  Available from the RRS archive log stream. See "Archive entry" on page 574.
- Resource manager entry

  Available from the RRS resource manager log stream. See "Resource manager entry" on page 576
- Resource manager meta data entry.

  Available from the Resource manager meta data log stream. See "Resource manager meta data entry" on page 577.

Each UR has a unique UR identifier. When you browse the UR state log stream, you will find multiple entries for a particular UR. Multiple resource managers might have expressed an interest in the UR, and the entries are logged over time. Only the entry with the latest timestamp contains current information.

Each resource manager also has a unique name. When you browse the resource manager data log stream, you might find multiple entries for a particular resource manager. Only the entry with the latest timestamp contains current information.

You can also filter by sysplex unit of recovery identifier (SURID). No wildcards are supported for SURID on this panel. SURID filtering is only valid for UR related log streams (archive, unit of recovery, and restart log streams).

UR entry

A **summary UR entry** includes the following information:

URID UR identifier

LOG STREAM
RRS log stream name

PARENT URID
If the UR is a cascaded UR, PARENT URID contains the identifier of the cascaded UR's parent UR.

SURID
Sysplex UR identifier for the UR. If a UR does not have a SURID, "N/A" will be displayed. The output also indicates when a UR is part of a cascaded (or multisystem cascaded) transaction.

WORK MANAGER NAME
Work manager name

Note: If the log entry being browsed was written by a version of RRS that does not save work manager names, the Work Manager name field will display `<NOT LOGGED>`.

STATE
UR state when the block was written

EXITFLAGS
Flag byte for the exit routine most recently driven for the UR. For the possible indicators, see the description of *exit_flags* in "Parameters" on page 101.

FLAGS
Indicators for the UR. You might need to supply the contents of this field to IBM service personnel.

LUWID
LU 6.2 logical unit of work identifier for this UR

TID Transaction identifier for this UR

GTID Global transaction identifier for this UR

FORMATID
Format identifier portion (in decimal) of the X/Open ID for this UR

GTRID
Global Transaction Identifier portion of the X/Open ID for this UR

BQUAL
Branch Qualifier portion of the X/Open ID for this UR

If an XID is not present, the FORMATID=, GTRID=, and BQUAL= fields are displayed as blanks.

A **detail UR entry** includes both the information in the summary UR entry and the following additional information for each protected expression of interest:

RMNAME
The name of the resource manager associated with this expression of interest in the UR. If there are multiple expressions of interest, there will be multiple detail UR entries.

ROLE The role the resource manager has in relation to this UR:

Participant: the resource manager is not the coordinator for this UR.

DSRM: the resource manager is the distributed syncpoint resource manager for this UR.

SDSRM: the resource manager is the server distributed syncpoint resource manager for this UR.

Last Agent: the resource manager is the last agent for this UR.

CMITCODE
A control code that indicates how the resource manager wants RRS to treat its COMMIT exit routine. A value of X'00000FFF' indicates that RRS is to drive the COMMIT exit routine. Any other value indicates that RRS is not to drive the exit routine; instead, RRS is to use the control code as the return code from the COMMIT exit routine.

BACKCODE
A control code that indicates how the resource manager wants RRS to treat its BACKOUT exit routine. A value of X'00000FFF' indicates that RRS is to drive the BACKOUT exit routine. Any other value indicates that RRS is not to drive the exit routine; instead, RRS is to use the control code as the return code from the BACKOUT exit routine.

PROTOCOL
The syncpoint logging protocol the resource manager has requested for this expression of interest:

1. PresumeNothing
2. PresumeAbort

Figure 23 shows the format of a detail UR entry.

```
SY2      2001/04/18 11:55:35.315619  LOCAL         BLOCKID=000000000006EFDE
  URID=B5B4AD1F7E87E4B80000000001020000 LOGSTREAM=ATR.PLEX1.DELAYED.UR
  PARENT URID=00000000000000000000000000000000 CASCADED
  SURID=020000100000000001000000000000B5B4AD1F7E87E4B80000000001020000
  WORK MANAGER NAME=SY2_RRJRLHY6_RM4
  STATE=InCommit     EXITFLAGS=00800000 FLAGS=A0800000
  LUWID=                                  TID=             GTID=

  FORMATID=003654931682 (decimal) D9D9D4E2 (hexadecimal)
  GTRID=
000000000000000000000000000000000000000000000000000B5B4AD1FA3CF4285
  BQUAL=
D9D9D4E24BB5B4AD1FA3CF4825000000000000000000000000000000000000000000000000000
  RMNAME=SY2_RRJRLHY6_RM4                 ROLE=Participant
    CMITCODE=00000FFF BACKCODE=00000FFF PROTOCOL=PresumeNothing
```

Figure 23. Detail Unit of Recovery Report Entry

Archive entry

The **summary archive entry** includes the following information:

URID Identifier of the UR

JOBNAME
Name of the job that requested commit or backout.

USERID
Identifier of the user that requested commit or backout.

PARENT URID
If the UR is a cascaded UR, PARENT URID contains the identifier of the cascaded UR's parent UR.

SURID
Sysplex UR identifier for the UR. If a UR does not have a SURID, "N/A" will be displayed. The output also indicates when a UR is part of a cascaded (or multisystem cascaded) transaction.

WORK MANAGER NAME
Work manager name

Note: If the log entry being browsed was written by a version of RRS that does not save work manager names, the Work Manager name field will display `<NOT LOGGED>`.

SYNCPOINT
The type of service the job requested:

1. Commit
2. Backout

RETURN CODE
Return code from commit or backout request

START
The date and time when RRS created the UR, or UNKNOWN if the information is not known. The time format is labeled with LOCAL.

COMPLETE
The date and time when RRS completed processing the UR. The time format is labeled with LOCAL.

EXITFLAGS
Flag byte set for the exit routine most recently driven for the UR. For the possible indicators, see the description of *exit_flags* in "Parameters" on page 101.

LUWID
LU 6.2 logical unit of work identifier for this UR

TID Transaction identifier for this UR

GTID Global transaction identifier for this UR

FORMATID
Format identifier portion (in decimal) of the X/Open ID for this UR

GTRID
Global Transaction Identifier portion of the X/Open ID for this UR

BQUAL
Branch Qualifier portion of the X/Open ID for this UR

If an XID is not present, the `FORMATID=`, `GTRID=`, and `BQUAL=` fields are displayed as blanks.

A **detail archive entry** includes the information in the summary archive entry and the following information about each expression of interest in the UR:

RMNAME
The name of the resource manager associated with this expression of interest in the UR. If there are multiple expressions of interest, there are multiple detail UR entries.

ROLE The role the resource manager has in relation to this UR:

Participant: the resource manager is not the coordinator for this UR.

DSRM: the resource manager is the distributed syncpoint resource manager for this UR.

SDSRM: the resource manager is the server distributed syncpoint resource manager for this UR.

Last Agent: the resource manager is the last agent for this UR.

FLAGS
Indicators for the UR. You might need to supply the contents of this field to IBM service personnel.

PROTOCOL
The syncpoint logging protocol the resource manager has requested for this expression of interest:

1. PresumeNothing
2. PresumeAbort

The information returned also includes a list of the exit routines that RRS might have driven for this expression of interest. Each exit routine shows one of the following:

- If the exit was driven for the UR interest, the most recent overall return code.
- If the exit was not driven, the value **Uncalled**.
- If the exit was driven for the UR but has not yet returned to RRS, the value **Called**.

Figure 24 shows the format of a detail archive entry.

```
SY1      11/15/1996 14:02:41.607677  LOCAL         BLOCKID=0000000000000001
  URID=ADC6419A7ED5400000000000101010000 JOBNAME=MAINASID USERID=*
  PARENT URID=00000000000000000000000000000000  CASCADED
  SURID=020000100000000001000000000000000B5B4AD1F7E87E4B80000000001020000
  WORK MANAGER NAME=ABCDEFGHIJKLMNOPQRSTUVWXYZ012345
  SYNCPOINT=Commit  RETURN CODE=00000000
  START=11/15/1996 18:44:12.510303 COMPLETE=11/15/1996 19:02:41.517837  LOCAL
  EXITFLAGS=00800000
  LUWID=ABC.DEF 00000000000 0002            TID=0028644343397 GTID=
1122334455667788
  FORMATID=1122334455 GTRID=
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
  BQUAL=
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
  RMNAME=UTRICK                          ROLE=Participant
    FLAGS=100E0000 PROTOCOL=PresumeNothing
    StateCheck EXIT RC=Uncalled
    Prepare    EXIT RC=00000000
    DistSp     EXIT RC=Uncalled
    Commit     EXIT RC=00000000
    Backout    EXIT RC=Uncalled
    EndUr      EXIT RC=00000000
    ExitFailed EXIT RC=Uncalled
    Completion EXIT RC=00000000
    OnlyAgent  EXIT RC=Uncalled
```

Figure 24. Detail Archive Report Entry

Resource manager entry

A resource manager entry reports information about a resource manager, obtained from the RRS resource manager log stream. The information in the entry includes:

RESOURCE MANAGER
The name of the resource manager

LOGGING SYSTEM
The name of the MVS system that wrote this log block for this resource manager

The entry also:

- identifies the system on which the resource manager was last active with RRS
- indicates whether there are any restrictions on the system where the resource manager can restart
- includes the name of the resource manager log, if one was provided
- includes the time stamp, labeled as the log instance number, when the Resource Manager last stopped using RRS services with no interests (thus allowing it to restart on any system).

Note: Duplicate entries for the same resource manager might be found in this log stream. For a specific resource manager, the only entry of interest to RRS is the latest, or youngest, log entry.

Figure 25 on page 577 shows the format of a resource manager entry.

```
RRS/MVS LOG STREAM BROWSE DETAIL  REPORT

READING ATRRRS.RM.DATA.LOG         LOG STREAM

SY1      08/23/1996 11:54:19.061552  LOCAL         BLOCKID=0000000000000161
 RESOURCE MANAGER=LOGGING1_RM1A                    LOGGING SYSTEM=SY1
  RESOURCE MANAGER MAY RESTART ON ANY SYSTEM
  RESOURCE MANAGER WAS LAST ACTIVE WITH RRS ON SYSTEM SY1
  A LOG NAME WAS NOT PROVIDED
  LOG INSTANCE NUMBER: 08/23/1996 11:54:19.117732

SY1      08/23/1996 11:54:27.070752  LOCAL         BLOCKID=0000000000000211
 RESOURCE MANAGER=ATR.RESOURCEMANAGER.IBM          LOGGING SYSTEM=SY1
  RESOURCE MANAGER MUST RESTART ON SYSTEM SY1
  RESOURCE MANAGER WAS LAST ACTIVE WITH RRS ON SYSTEM SY1
  LOG NAME IS ATR.AD5C492088350281.IBM
  LOG INSTANCE NUMBER: 08/20/1996 23:20:15.772928
```

Figure 25. Sample Resource Manager Entry

Resource manager meta data entry

A resource manager meta data entry reports information saved by a resource manager, obtained from the RRS resource manager meta data log stream. The information in the entry includes:

RESOURCE MANAGER
The name of the resource manager

LOGGING SYSTEM
The name of the MVS system that wrote this log block for this resource manager

The entry also includes the meta data stored by the resource manager in a hex dump format.

Figure 26 shows the format of a resource manager meta data entry.

```
RRS/MVS LOG STREAM BROWSE SUMMARY REPORT

READING ATR.PLEX1.RM.METADATA    LOG STREAM

SY1     2005/09/23 13:42:40.673986  LOCAL         BLOCKID=0000000000003B45
 RESOURCE MANAGER=LOGGING_RM1                    LOGGING SYSTEM=SY1

  0000-000F 40D4C5E3 C1C4C1E3 C1404040 D9D4F140 * METADATA RM1 *
```

Figure 26. Sample Resource Manager Meta Data Entry

Working with resource manager information

From the resource manager selection panel, shown in Figure 27 on page 578, you can identify a specific resource manager or press Enter to see a scrollable list of resource managers. If you know, for example, the resource manager associated with an application that you need to check on, you can go directly to the resource manager.

You can display any resource managers that:

- Are currently active with RRS within the sysplex where you are using the panels
- Are currently not active with RRS but were last active with a running RRS within the sysplex where you are using the panels

Note: You can specify wildcard characters in the specification of resource manager names, system names, and logging group names to display: * for zero or more characters, and ? for any single character.

```
                        RRS Resource Manager Selection
Command ===> ____________________________________________________________

Optionally provide resource manager name, system name, and logging group
patterns and press Enter:

RM name  . . . . . . _____________________________
System name  . . . . ________
Logging Group  . . . ________

If left blank, system name and logging group will default to the current
system's name and group.
```

Figure 27. Resource Manager Selection (ATRFPRMS)

Once a resource manager has set its exit routines, RRS can display information about the resource manager, and it remains in the list until RRS is cancelled.

After you have made your selection, RRS lists the resource managers you have selected. Figure 28 on page 579 shows the format of the scrollable list. The returned information is sorted alphanumerically with a primary ascending sort key of logging group name and a secondary ascending sort key of system name. Press PF6 to refresh the list and display the new list of resource managers that match the resource manager name string you supply.

```
                        RRS Resource Manager List         ROW 34 TO 50 OF 50
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
    ONEWAYA_WM4NAME_SY1              Run              SY1      PLEX1
    ONEWAYB_RM1NAME_SY1              Run              SY1      PLEX1
    ONEWAYB_RM2NAME_SY1              Run              SY1      PLEX1
    ONEWAYB_RM3NAME_SY1              Run              SY1      PLEX1
    ONEWAYB_RM4NAME_SY1              Run              SY1      PLEX1
    ONEWAYB_WM1NAME_SY1              Set              SY1      PLEX1
    ONEWAYB_WM2NAME_SY1              Set              SY1      PLEX1
    ONEWAYB_WM3NAME_SY1              Run              SY1      PLEX1
    ONEWAYB_WM4NAME_SY1              Run              SY1      PLEX1
    ONEWAYC_RM1NAME_SY1              Run              SY1      PLEX1
    ONEWAYC_RM2NAME_SY1              Run              SY1      PLEX1
    ONEWAYC_RM3NAME_SY1              Run              SY1      PLEX1
    ONEWAYC_RM4NAME_SY1              Run              SY1      PLEX1
    ONEWAYC_WM1NAME_SY1              Run              SY1      PLEX1
    ONEWAYC_WM2NAME_SY1              Set              SY1      PLEX1
    ONEWAYC_WM3NAME_SY1              Set              SY1      PLEX1
```

Figure 28. Resource Manager List (ATRFPRML)

The list shows the name of the resource manager and its state, as known on the local system. To work with a resource manager, select it and choose the action you want to perform. You can:

1. Obtain the resource manager token and the resource manager state. Enter `v` in the selection field to view details. The system will display the resource manager (RM) name, token, and state.
2. Obtain a scrollable list of the URs associated with the selected resource manager. Enter `u` in the selection field to view URs. “Working with UR information” describes how to work with the information you receive.
3. Remove all of a resource manager's interests, possibly in all URs. Enter `r` in the selection field to remove an interest. “Removing a resource manager interest in a UR” on page 592 describes how to continue.
4. Delete all the resource manager information from all RRS systems in the logging group and the RRS Resource Manager logs. The specified resource manager must not be active with RRS or have any interests in the URs.
5. Clean up the resource manager's involvement with RRS. Enter `n` in the selection field to unregister a resource manager. Only resource manager's that have been unregistered with Registration Services but still registered with RRS can be processed. RRS Resource Manager Unregister processing will be invoked to clean up the resource manager's involvement with RRS. The system will display the Unregister RM Confirmation panel to verify that the user wants to perform the unregister action. RRS will not unregister the resource manager when:
   - The resource manager is still registered with Registration Services.
   - The resource manager state is either Reset or Unset. A resource manager in Reset or Unset state is already considered unregistered with RRS.

Working with UR information

When you request UR information from the main selection panel, you see the scrollable panel shown in Figure 29 on page 581.

The scrollable panel allows you to create a profile defining one or more URs you would like to see information about. You can view URs

- in one or more specific states;
- on specific systems or logging groups in a sysplex;
- with interests in specific resource managers or work managers; and,
- based on when they were created, or how long they have been in some state.

Note: You can specify wildcard characters in the specification of system names, logging group names, resource manager names, and work manager names to display: `*` for zero or more characters, and `?` for any single character.
You can exclude URs that

- are not deferred;
- require restart processing;
- have been read from the restart log;
- cascaded URs in a cascaded UR family;
- are in local transaction mode;
- are in global transaction mode (both global transaction mode and hybrid-global transaction mode);
- are managed by RRS; or,
- are managed by some other work manager.

You can sort URs in a variety of ways. They can be presented in ascending or descending order, based on these criteria that you can assign sort priority:

- Work Manager Name;
- UR Identifier;
- Creation Time;
- UR State;
- Logical Unit of Work ID (LUWID);
- Enterprise ID (EID);
- X/Open ID (XID);
- Logging Group Name;
- System Name; and,
- Sysplex UR identifier (SURID).

You can also quickly store or retrieve a specific UR profile. Enter `save` to store the current profile. Enter `get` to retrieve a profile. See "Working with UR selection profiles" on page 583 for more information about working with UR selection profiles.

```
                        RRS Unit of Recovery Selection
  Command ===> ____________________________________________________________

  Commands: save-Save Profile  get-Get Profile  ENTER-Query

  Profile Name  . .  ________ Profile Data Set HLQ . . ________

  UR and Work Identifier Criteria====================
  URID pattern . . ___________________________________

  SURID pattern
  ____________________________________________________________________

  LUWID pattern (netid.luname,instnum,seqnum)
  ___________________________________________

  TID  . . . . ____________  (from 1 to 4294967295 in decimal)
  Low TID  . . ____________  High TID . . ____________

  GTID Pattern
  00-0F  ____________________________________________________________
  10-1F  ____________________________________________________________
  20-27  ____________________________________________________________

Format ID    ____________  (from 1 to 4294967295 in decimal)

GTRID Pattern
00-0F  ____________________________________
10-1F  ____________________________________
20-2F  ____________________________________
30-3F  ____________________________________
40-4F  ____________________________________
50-5F  ____________________________________
60-6F  ____________________________________
70-7F  ____________________________________

BQUAL Pattern
00-0F  ____________________________________
10-1F  ____________________________________
20-2F  ____________________________________
30-3F  ____________________________________
40-4F  ____________________________________
50-5F  ____________________________________
60-6F  ____________________________________
70-7F  ____________________________________

  UR Type/State Criteria=============================
  UR type:              UR State:   ('/' - select one or more)
  __  1.  all           _  all                    _  In_Commit
      2.  unprotected   _  In_Flight              _  In_Backout
      3.  protected     _  In_State_Check         _  In_End
                        _  In_Prepare             _  In_Completion
                        _  In_Doubt               _  In_Forget

  System Criteria==========================
  System name . . . . . . . ________
  Logging Group . . . . . . ________

  If left blank, system name and logging group will default to the current
  system's name and group.

  Resource Manager Criteria==========================
  Resource Manager Name . . ___________________________________
  Work Manager Name . . . . ___________________________________
```

```
  Time-related Criteria==============================
  Time Format . . . . . . . GMT__              (Local|GMT)
  UR created after time  . . ________          (hh:mm:ss)
  UR created after date  . . __________        (yyyy/mm/dd)
  UR created before time . . ________          (hh:mm:ss)
  UR created before date . . __________        (yyyy/mm/dd)
  Min Time in State  . . . . ____________      (hh:mm:ss.ssss)

  Exclusion Criteria=================================
  _  Interests which are not deferred
  _  Restart Required Interests
  _  UR was read from Restart Log
  _  Cascaded URs in a cascaded UR family
  _  UR with Local Transaction Mode
```

Note: The "Time Format" defaults to the value entered on the "RRS Global Panel Options" panel. See Figure 21 on page 570.

Because there might be a large number of URs, use this panel to define the URs you want to see. If you have specific information about the UR, such as its identifier, you can specify it here. Otherwise, leave the UR identifier blank but use UR type and/or UR state choices to define a subset of the possible URs. Checking for URs that are **In_Doubt**, for example, might be an early step in troubleshooting an application that you think might be hung because of a resource recovery problem.

When you have defined your selection, press Enter. The information you requested appears in the format shown in Figure 32 on page 584.

Note: If some of your requests fail, you may receive error message ATR510I. If so, press PF5 (LISTERR) to view the error information on the panel shown in Figure 30.

```
                          RRS ATRQUERY RC Table
Command ===> _____________________________________________ Scroll ===> ____

Commands: s-View Error Message

Press EXIT to return to the previous panel.

S   System   Logging Group QueryRC  QueryRSN   SrvID    SrvRC    SrvRSN
_   ________ ________      ________ ________ ________ ________ ________
```

Figure 30. RRS ATRQUERY RC Table (ATRFPRCL)

Viewing multisystem cascaded UR families

To find all top level cascaded URs:

- Use System Name = *
- Exclude cascaded URs

To view a single cascaded UR family:

- Use the `f` command on the UR list panel (ATRFPURL) against the top (T) or child (C) UR from the family you are interested in.
- Use the SURID of the sysplex cascaded UR family as a UR filter.

To find all the top level cascaded URs associated with a work manager::

- Use Work Manager Name = *wmname* (DB2, for example)
- Use System Name = *
- Exclude cascaded URs

Working with UR selection profiles

RRS ISPF Panels allow you to save and reuse sets of UR selection criteria. From the main selection panel shown in Figure A-2 on page A-4, you can select the "Display/Update RRS UR selection criteria profiles" option to access a subpanel to manage your currently defined UR selection profiles. You can rename, copy, or delete a profile from this panel. You can also select a profile to view or update its settings as in the panel shown in Figure 29 on page 581.

A partitioned dataset (PDS) is allocated, if one does not already exist, to hold the profiles. The data set name is in the pattern: *hlq*.ATR.PROFILE', where *hlq* is the high level qualifier. RRS has an option in the UR selection panel and the profile selection panel to allow you to specify a high level qualifier. The high level qualifier you provide can contain periods, so you may provide multiple qualifiers. If you do not specify a high level qualifier, RRS will use your TSO user PREFIX. If you do not have a TSO user PREFIX, your TSO userid will be used.

Each member in the partitioned data set represents a single profile, with the same name as specified in the Profile name field.

A partitioned data set extended (PDSE) can be used instead of a PDS as long as the PDSE is allocated prior to using the RRS ISPF Panels. When using a PDSE, there could be instances when message "Permanent I/O error" is issued by the RRS panel processing indicating the data set is out of space. Contact your System Programmer to discuss how to correct the error.

Following are some samples of UR panels.

```
                        RRS UR Selection Profiles
Command ===> _____________________________________________ Scroll ===> ____

Options: s-Select r-Rename c-Copy d-Delete


Profile Name . . . . ________  Profile Data Set HLQ . . ________

Option  Profile   Prompt     Description
_       ________  ________   ____________________________________
```

Figure 31. UR Selection Profiles (ATRFPURP)

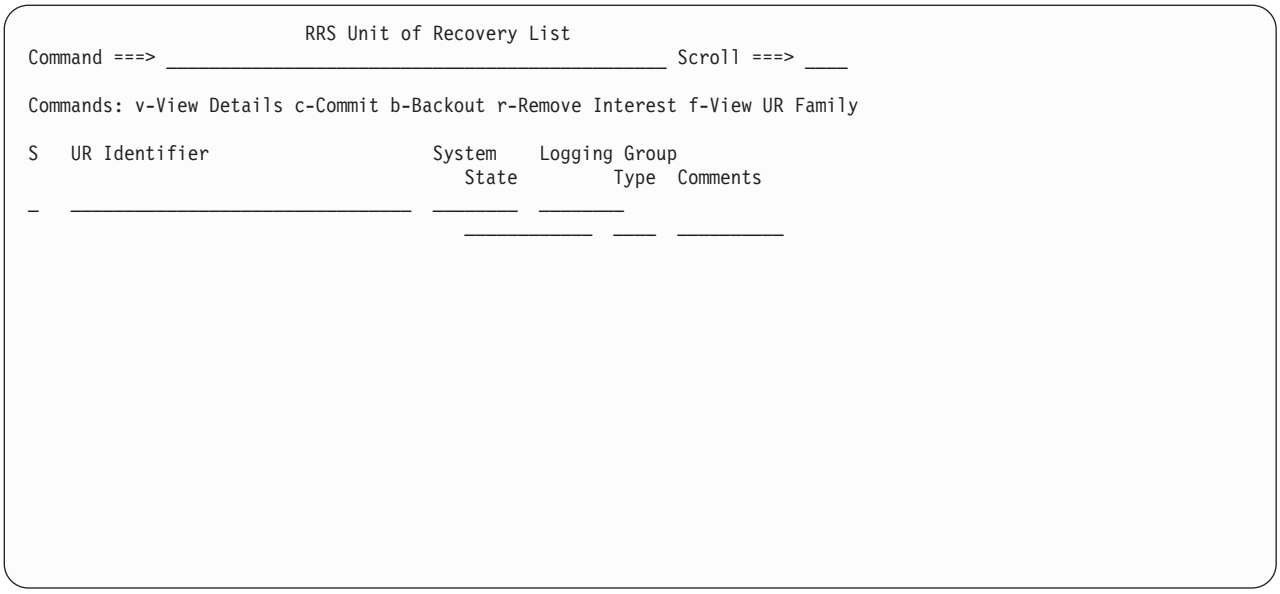

Figure 32. UR List (ATRFPURL)

The UR list panel shows the UR identifier, the system and logging group the UR is on, and the state of each UR, along with its UR type and comments about the UR. If more than one UR met your criteria, the list is scrollable. To work with a specific UR, select it and choose the action you want to perform.

The panels shown in Figure 32 and Figure 33 on page 586 may display the following comments about a UR:

Table 54. UR Comments

UR comment	Meaning
A	The UR is waiting for the child or subordinate application to signal that it is complete (ready for the syncpoint to be driven).
C	The UR is a child UR in a cascaded UR family.
D	The UR is damaged.
E	The UR is waiting for a resource manager to reply to a syncpoint exit.
L	The UR is in local transaction mode.
M	The UR is in a heuristic mixed condition.
P	The coordinator UR is waiting for a response from RRS on one or more remote systems in the sysplex.
R	The UR information came from the RRS Restart log stream.
S	The UR is part of a multisystem cascaded UR family.
T	The UR is the top-level UR of a cascaded UR family.
U	The UR information came from the RRS Main or Delayed log stream. **Note:** This entry usually represents an incomplete child UR in a multisystem cascaded transaction on a subordinate system where either RRS or the system itself has failed and the parent UR is still active on the coordinator system.
X	The UR and its interests are not all in the same state.

Table 54. UR Comments (continued)

UR comment	Meaning
*	A portion of the syncpoint represented by this UR has been marked deferred.
?	The UR contains information that this release of RRS does not understand. For example, the Release 1.3 level of RRS does not understand the XID data present in some release 2.5 log blocks, and the comments field will display a ? to indicate this.

You can:

1. Obtain additional details about the UR including any associated expressions of interest. Enter `v` on the command line to view details. The format of the information you receive appears in Figure 33 on page 586.

   From the panel shown in Figure 33 on page 586, you can enter `v` to obtain information about expressions of interest in the UR. See Figure 34 on page 586. You can also enter `r` to remove an interest. "Removing a resource manager interest in a UR" on page 592 describes how to continue.

   From the panel shown in Figure 33 on page 586, you can also place a non-blank character in the `Display Work IDs` field, to present a separate panel showing the work identifiers, either formatted or unformatted depending on your specification in the `Display IDs formatted` field. If you specify formatted output, the work identifiers are displayed via the panel shown in Figure 36 on page 589. Unformatted output identifiers are displayed via the panel shown in Figure 37 on page 589.
2. Change the state of the UR from **In-Doubt** to **In-Commit**. Enter `c` on the command line. You will receive a confirmation panel so that you can verify the change.

   RRS will reject the request when:

   - The UR is not **In-Doubt**.
   - You are trying to process a UR that contains information that this release of RRS does not understand.

   Note: Using this command to change the UR state bypasses normal protocols. Its use can leave data in an inconsistent state.
3. Change the state of the UR from **In-Doubt** to **In-Backout**. Enter `b` on the command line. You will receive a confirmation panel so that you can verify the change.

   RRS will reject the request when:

   - The UR is not **In-Doubt**.
   - You are trying to process a UR that contains information that this release of RRS does not understand.

   Note: Using this command to change the UR state bypasses normal protocols. Its use can leave data in an inconsistent state.
4. Remove an interest associated with the UR. Enter `r` to remove an interest. "Removing a resource manager interest in a UR" on page 592 describes how to continue.
5. View the UR's UR family details. Enter `f` to view UR family details. Cascaded UR family information is displayed via the panel shown in Figure 35 on page 587

587. The f command can be used to display the entire multisystem cascaded UR family for a given top level UR or child UR.

```
                        RRS Unit of Recovery Details
Command ===> _____________________________________________ Scroll ===> ____

Commands r-Remove Interest v-View URI Details

UR identifier : _____________________________
Create time : _____________________________  GMT        Comments: _______________
UR state : ____________  UR type : ____
System : ________  Logging Group : ________
SURID  : ____________________________________________________________________
Work Manager Name : _____________________________
_  Display Work IDs           _  Display IDs formatted
   Luwid  . : ___________
   Eid  . . : ___________
   Xid  . . : ___________
Expressions of Interest:
S   RM Name                          Type  Role
_   _______________________________  ____  ___________
```

Figure 33. UR Details (ATRFPURD)

Note: The "Create time" format (GMT | Local) defaults to the value entered on the "RRS Global Panel Options" panel. See Figure 21 on page 570.

```
                               RRS URI Details
   Command ===> ____________________________________________________________

UR identifier :
URI token . . :
RM name . . . :
Type  . . . . :              Status . :
Role  . . . . :               State  . :
SURID:

   Display persistent interest data

Exit/State      Status                 Duration
BACKOUT  . . . :
COMPLETION . . :
COMMIT . . . . :
DSE/IN_DOUBT . :
End_UR . . . . :
EXIT_FAILED  . :
ONLY_AGENT . . :
PRE_PREPARE  . :
PREPARE  . . . :
STATE_CHECK  . :
```

Figure 34. UR Interest Details (ATRFPURE)

From the panel shown in Figure 34, you can see the status of a UR interest. This interest has four possible values described in Table 55 on page 587: If you select "**Display Persistent Interest Data**" from the panel shown in Figure 34, you can see the results in Figure 38 on page 590.

Table 55. UR Interest Status

Status description	Meaning
ACTIVE	The interest in the unit of recovery is active and has not reached forgotten state.
COMPLETE	The interest in the unit of recovery is complete and has reached forgotten state.
DEFERRED	The interest in the unit of recovery is active; however, some processing has occurred to defer its execution. A resource manager may have responded ATRX_LATER to an exit, indicating that the resource manager will return to RRS later with the processing results for the current state of the unit of recovery. The unit of recovery is marked deferred if its interest was being processed by an RRS server task, but it was deferred because the syncpoint processing was required to wait for some event. The interest will remain deferred until the event occurs. After the event occurs, the interest will return to ACTIVE status.
RESTREQ	The Resource Manager has failed with RRS, and has not yet restarted this interest. **Note:** RRS could have failed, causing the failure for all Resource Managers.Regardless of the reason, if the Resource Manager has not completed enough of the restart processing to transition this interest to ACTIVE state.
Note: If there is some question about the state of the Resource Manager with RRS, you may wish to go to the main RRS selection panel, shown in Figure 20 on page 568, and opt to have RRS display the current state for this Resource Manager.	

```
                          RRS Cascaded UR Family
Command ===> _____________________________________________ Scroll ===> ____

Commands v-View Details c-Commit b-Backout
Top-level UR identifier : _____________________________
Top-level Work Manager  : _____________________________
Top-level Create Time . : _____________________________  GMT
SURID : ____________________________________________________________________
Comments : ____________________
Family State : ____________
Cascaded UR Family:
S   Level  UR Identifier                    System        Type  Comments
_   _____  ____________________________  ____________  ____  __________
```

Figure 35. Cascaded UR Family (ATRFPCFD)

Note: The "Top-level Create Time" format (GMT | Local) defaults to the value entered on the "RRS Global Panel Options" panel. See Figure 21 on page 570.

The panel shown in Figure 35 on page 587 displays information about a cascaded UR family. The information is displayed in two parts. The first part contains information about the top-level UR of the family, and information pertaining to all the URs in the family. The second part is a scrollable list of all the URs in the family. The list is displayed in order from the top-level UR to the lowest level cascaded child URs.

Only a top-level UR in **in-doubt** state can be committed or backed out via the Cascaded UR Family panel.

Exit duration inconsistencies

System programmers may notice the reported total elapsed time of a syncpoint event that is not equal to the cumulative time spent driving resource manager exits for syncpoint processing. This anomaly is explained by two timing behaviors:

- Uncaptured time can cause an observed exit time accumulation somewhat smaller than the reported total. Uncaptured time occurs when RRS does not capture time it spends in preparation of driving a resource manager exit, but this time appears as part of the total syncpoint duration.
- Multi-programming can cause an observed exit time accumulation exceeding the reported total elapsed time duration. Multi-programming is the driving of multiple exits for the same syncpoint simultaneously, resulting in an overall smaller elapsed time than the accumulation duration of all the exits.

Exit duration and RRS failures

RRS does not harden exit durations (although it does harden Unit of Recovery creation time). Therefore, if RRS fails, the system programmer may notice zero length durations for states previously completed and logged. Because of resource manager restart processing, the system programmer may also see resource manager interests regress in state. For example, if an interest has progressed past the COMMIT state to the IN_END state, upon restart the interest would be returned to the COMMIT state. In these cases, if RRS has not failed across the resource manager restart, all durations prior to the COMMIT state are preserved and are displayable; however, the exit duration for COMMIT state and all exit durations following the COMMIT state are reset. If RRS has failed, then all durations prior to the COMMIT state are reported as having a zero length.

Similar processing is performed for restart interests returned in BACKOUT state. BACKOUT state has the exception, however, that a phase-1 exit (of the two-phase commit protocol) may have failed. If RRS has not failed across the resource manager restart, RRS will report any incomplete exit in phase-1 as having been completed during the restart process. Therefore, durations for the incomplete exits are reported as the difference between the resource manager restart time and the time the exit was originally driven.

```
                    RRS Unit of Recovery Work IDs

UR identifier : ____________________________

Logical Unit of Work Identifier (LUWID):
 ____________________________________

 NetID.LuName : ________________
 TP Instance  : ____________
 SeqNum . . . : ____

Enterprise Identifier (EID)
 TID  : ____________  (decimal)
 GTID : ____________________________________________________________
        ____________________________________________________________
        ____________________________________________________________

X/Open Transactions Identifier (XID)
 Format ID : ____________  (decimal)
             ____________  (hexadecimal)
 GTRID : ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
 BQUAL : ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
         ____________________________________________________________
```

Figure 36. Formatted UR Work IDs (ATRFPIDF)

The panel shown in Figure 36 is scrollable, because the formatted work identifiers will not fit on a standard 24x80 screen.

The GTID, GTRID, and BQUAL are all displayed in standard IPCS hex dump format like:

```
 0- F  00010203 04050607 08090A0B 0C0D0E0F  |................|
10-1F  10111213 14151617 18191A1B 1C1D1E1F  |................|
20-27  20212223 24252627                    |........        |
```

```
                    RRS Unit of Recovery Work IDs

UR identifier : ____________________________

Logical Unit of Work Identifier (LUWID)
 ____________________________________________________

Enterprise Identifier (EID)
 ____________________________________________________________________
 ____________________________________________________________________

X/Open Transactions Identifier (XID)
 ____________________________________________________________________
 ____________________________________________________________________
 ____________________________________________________________________
 ____________________________________________________________________
```

Figure 37. Unformatted UR Work IDs (ATRFPIDU)

Figure 38 is displayed if you click on "**Display Persistent Interest Data**" from the panel shown in Figure 34 on page 586.

```
                  RRS URI Persistent Interest Data

Command ===>

UR identifier : BC94B8A07E25C0000000000001010000
URI token . . : 7E15C000000000000055000155555555

Offset     PI Data
 000-00F   00010203 04050607 08090A0B 0C0D0E0F  |................|
 010-01F   10111213 14151617 18191A1B 1C1D1E1F  |................|
 020-027   20212223 24252627                    |........        |
```

Figure 38. RRS URI Persistent Interest Data (ATRFPPDT)

Mixed unit of recovery states

Certain events and conditions can cause the interests in a UR to be in different states from not only other interests in the UR, but also the overall state or the UR itself. The RRS system management query function will accurately reflect these mixed interest states. The following list of syncpoint events can cause interests in a given UR and the overall UR to be in mixed states:

- **Resource manager failure:** When a resource manager fails while it has an interest in a UR, the syncpoint processing for the UR is allowed to proceed. The UR will progress in state, but the failed expression of interest will not proceed until the resource manager restarts with RRS, and indicates that it is ready to continue processing the syncpoint. Since units of recovery can have numerous expressions of interests from a variety of resource managers depending on the complexity of the transaction, it is clear that multiple failures of resource managers and the asynchronous nature of resource manager restart can cause this mixed state condition.
- **Resource manager optimization:** A resource manager can respond to RRS that it has completed all of the syncpoint processing for a UR. RRS will transition that resource manager's interest to IN_FORGET state immediately, even though the overall UR state has not progressed to IN_FORGET. A resource manager can also respond to RRS that it has completed the critical syncpoint processing for a given UR, and syncpoint processing can continue.
- **Timing:** The RRS system management query function examines the RRS data structures in an unserialized manner to minimize the impact of the query to the component's performance. Since the query function is running unserialized, inconsistencies in state can sometimes appear.

Working with work manager information

From the resource manager selection panel, shown in Figure 39 on page 591, you can identify a specific work manager or press Enter to see a scrollable list of work managers. If you know, for example, the work manager associated with an application that you need to check on, you can go directly to that work manager.

You can display any work managers that:

- Own contexts that have incomplete units of recovery on this system; or,
- Own or owned contexts that have incomplete units of recovery which are currently not active with RRS on any system, but were last active on this system.

Note: If a native context has an incomplete unit of recovery, the work manager name reported by RRS is reported as the following concatenated string:

- SystemName
- Period (.)
- JobName of the native context address space
- Period (.)
- ASID (4 bytes readable hexadecimal)
- Blanks (padded to 32 bytes)

Note: You can specify wildcard characters in the specification of work manager names, system names, and logging group names to display: * for zero or more characters, and ? for any single character.

```
                        RRS Work Manager Selection
Command ===> ____________________________________________________________

Optionally provide a work manager name, system name, and logging group
patterns and press Enter:

WM name  . . . . . . _____________________________
System name  . . . . ________
Logging group  . . . ________

If left blank, system name and logging group will default to the current
system's name and group.
```

Figure 39. Work Manager Selection (ATRFPWMS)

After you have made your selection, RRS lists the work managers you have selected. Figure 40 on page 592 shows the format of the scrollable list. The returned information is sorted alphanumerically with a primary ascending sort key of logging group name and a secondary ascending sort key of system name. Press PF6 to refresh the list and display the new list of work managers that match the resource manager name string you supply.

Note: The work manager panels will not show UR information from the restart log stream for cascaded or multisystem cascaded transactions.

```
                          RRS Work Manager List
Command ===> _____________________________________________ Scroll ===> ____

Commands: u-View URs

S   WM Name                           System    Logging Group
_   ___________________________________  ________  ________
```

Figure 40. Work Manager List (ATRFPWML)

Removing a resource manager interest in a UR

When you indicate that you want to remove an interest that a resource manager has in a UR, you see the panel shown in Figure 41.

Note: Use extreme caution when removing interest in a UR – Only do this action as a last resort.

```
                     RRS Remove Interest Confirmation
Command ===> ____________________________________________________________

You are requesting that some resource manager interests be removed from
one or more URs.
RM name  . . . . . : ____________________________________
UR identifier  . . : ____________________________________
System . . . . . : ____________________________________
Logging Group  . . : ____________________________________
SURID  . . . . . : ____________________________________

Press ENTER to continue, PF3 to cancel
```

Figure 41. Remove Interest Confirmation (ATRFPRIN)

Both the RM name and the UR identifier on the Remove Interest panel might be filled in when you see the panel. You can change either field, and you must be sure that at least one of the fields is filled in. If you supply only a resource manager name, the system removes all the resource manager's interests from all associated URs. If you supply only a UR identifier, the system removes all resource managers' interests associated with the UR.

Specifying both a resource manager name and a UR identifier removes only interests the specified resource manager has in the specified UR.

The panel displays the system, logging group, and SURID associated with the UR, but these fields are only informational. They can help you verify that the correct URs are being processed. You must choose a specific RM name, UR identifier, or both when you want to remove interest.

When you are ready, press Enter. You will receive a confirmation panel so that you can verify your action.

RRS will prevent you from removing an interest in a UR when:

- Any resource manager associated with the request to remove an interest is still active with RRS.
- You are trying to remove a particular resource manager's interest in an **in-doubt** UR when the resource manager is functioning as the distributed syncpoint resource manager (DSRM) or server distributed syncpoint resource manager (SDSRM) for the UR.
- You are trying to remove interest in a UR that contains information that this release of RRS does not understand.

Working with RRS system information

When you request RRS related system information from the main selection panel, you can ask for information on specific systems or logging groups using the panel shown in Figure 42.

Note: You can specify wildcard characters in the specification of system names and logging group names to display: * for zero or more characters, and ? for any single character.

```
                    RRS System Information Selection
Command ===> ____________________________________________________________

Optionally provide a system name and logging group patterns and press Enter:

System name  . . . . ________
Logging group  . . . ________

If left blank, system name and logging group will default to '*'.
```

Figure 42. RRS System Information Selection (ATRFPSIS)

Based on your selections, you will receive data in the scrollable panel shown in Figure 43 on page 594. If you don't select specific or wildcarded system names or logging groups, all the system names and logging groups active with RRS are

displayed.

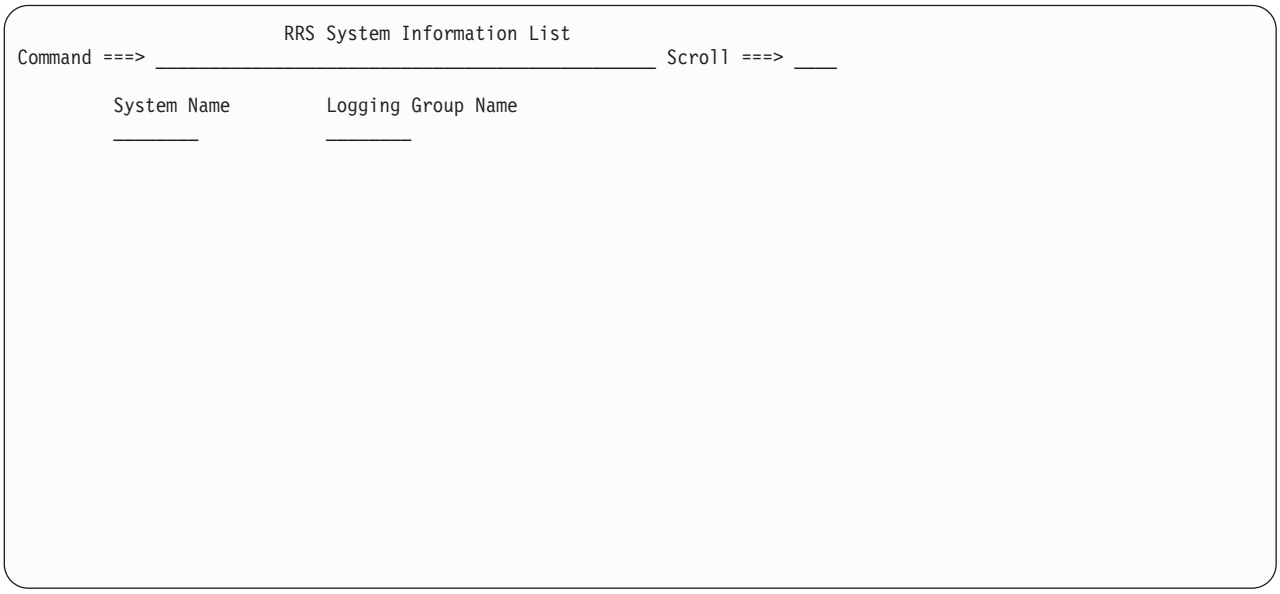

Figure 43. RRS System Information List (ATRFPSIL)

Chapter 11. ATRQUERY — Obtain RRS Information

The ATRQUERY macro allows callers to:

- Retrieve information about resource managers that are involved with RRS processing
- Retrieve information about units of recovery (URs)
- Retrieve information about resource manager metadata.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state, any PSW key, or RACF authority as defined in the following paragraphs
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held
Control parameters:	Control parameters must be in the primary address space.

If your installation uses the RACF component of SecureWay for z/OS to control access to RRS information, ATRQUERY requires READ access to one of two RACF resources in the FACILITY class.

To view URs on other systems in a sysplex, ATRQUERY requires READ access to the MVSADMIN.RRS.COMMANDS.*gname*.*sysname* resource in the FACILITY class, where *gname* is the target logging group name, and *sysname* is the target system name. You may create a RACF profile to permit access to multiple logging groups and systems by including RACF valid generic characters (**, *, and %) in *gname* and *sysname*. See the *z/OS Security Server RACF Security Administrator's Guide* and *z/OS Security Server RACF Command Language Reference* for more information about using these RACF generic characters and defining RACF profiles. By permitting READ access through this profile, you can allow users to view RRS information on any number of systems in the sysplex.

If you are running RRS on a single system, ATRQUERY requires READ access to either the MVSADMIN.RRS.COMMANDS.*gname*.*sysname* resource or the MVSADMIN.RRS.COMMANDS resource in the FACILITY class. The MVSADMIN.RRS.COMMANDS resource only allows access to RRS system management functions on the current system. You cannot use the MVSADMIN.RRS.COMMANDS resource to allow or disallow use of RRS on another system.

ATRQUERY does not check the SYSSTATE global variable or the SPLEVEL global variable.

Programming requirements

ATRQUERY callers must include the following mapping macros: ATRFZQRY, CVT, IHAECVT

To use the ATR_* constants referenced in the keyword descriptions, ATRQUERY callers must include the ATRRASM mapping macro.

Restrictions

None.

Input register information

Before issuing the ATRQUERY macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Contains the reason code
1-13	Unchanged
14	Used as a work register by the system.
15	Contains the return code

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The ATRQUERY macro is written as follows:

Note: Keywords prefaced with a vertical bar (|) are mutually exclusive.

name	*name:* symbol. Begin *name* in column 1.
␣	One or more blanks must precede ATRQUERY.
ATRQUERY	
␣	One or more blanks must follow ATRQUERY.

```
 REQUEST=RMINFO
|REQUEST=WMINFO
```

| REQUEST=SYSINFO
| REQUEST=URINFO

,RMNAME=*rmname* — *rmname:* RS-type address or address in register (2) - (12).

,WMNAME=*wmname* — *wmname:* RS-type address or address in register (2) - (12).

,ASID=*asid* — *asid:* RS-type address or address in register (2) - (12).

,URID=*urid* — *urid:* RS-type address or address in register (2) - (12).

| ,URIDSTR=*uridstr* — *uridstr:* RS-type address or address in register (2) - (12).

,SURID=*surid* — *surid:* RS-type address or address in register (2) - (12).

| ,SURIDSTR=*suridstr* — *suridstr:* RS-type address or address in register (2) - (12).

,URTYPE=*urtype* — *urtype:* symbol or value in register (2) - (12).

,URSTATE=*urstate* — *urstate:* symbol or value in register (2) - (12).

| ,URSTMASK=*urstmask* — *urstmask:* RS-type address or address in register (2) - (12).

,LUWID=*luwid* — *luwid:* RS-type address or address in register (2) - (12).

| ,LUWIDSTR=*luwidstr* — *luwidstr:* RS-type address or address in register (2) - (12).

,TID=*tid* — *tid:* RS-type address or address in register (2) - (12).

| ,TIDLOW=*tidlow* — *tidlow:* RS-type address or address in register (2) - (12).

| ,TIDHIGH=*tidhigh* — *tidhigh:* RS-type address or address in register (2) - (12).

,GTID=*gtid* — *gtid:* RS-type address or address in register (2) - (12).

| ,GTIDSTR=*gtidstr* — *gtidstr:* RS-type address or address in register (2) - (12).

| ,XID=*xid* — *xid:* RS-type address or address in register (2) - (12).

| ,XIDFORMATIDSTR=*xidformatidstr* — *xidformatidstr:* RS-type address or address in register (2) - (12).

| ,XIDGTRIDSTR=*xidgtridstr* — *xidgtridstr:* RS-type address or address in register (2) - (12).

| ,XIDBQUALSTR=*xidbqualstr* — *xidbqualstr:* RS-type address or address in register (2) - (12).

,RMNAME=*rmname* — *rmname:* RS-type address or address in register (2) - (12).

,WMNAME=*wmname* — *wmname:* RS-type address or address in register (2) - (12).
,GNAME=*gname* — *gname:* RS-type address or address in register (2) - (12).
,SYSNAME=*sysname* — *sysname:* RS-type address or address in register (2) - (12).
,METADATA

,TODBEG=*todbeg* — *todbeg:* RS-type address or address in register (2) - (12).

,TODEND=*todend*	*todend:* RS-type address or address in register (2) - (12).
,CURDUR=*curdur*	*curdur:* RS-type address or address in register (2) - (12).
,DEFURONLY=*defuronly*	*defuronly:* RS-type address or address in register (2) - (12).
|,EXCLMASK=*exclmask*	*exclmask:* RS-type address or address in register (2) - (12).
,SORTTAB=*sorttab* ,SORTNUM=*sortnum*	*sorttab:* RS-type address or address in register (2) - (12). *sortnum:* RS-type address or address in register (2) - (12). **Note:** SORTTAB and SORTNUM must be specified together, on the same line, separated by a comma.
,EXINFOMASK=*exinfomask*	*exinfomask:* RS-type address or address in register (2) - (12).
,AREAADDR=*areaaddr*	*areaaddr:* RS-type address or address in register (2) - (12).
,AREALEN=*arealen* ,AREAALET=*areaalet* |,AREAOPT=BYADDR |,AREAOPT=BYOFFSET	*arealen:* RS-type address or address in register (2) - (12). *areaalet:* RS-type address or address in register (2) - (12). Return complete addresses in the control structures Return offsets from the beginning of the storage area in the control structures
,COUNT=*count*	*count:* RS-type address or address in register (2) - (12).
,RCTABLE=*rctable* ,RCNUM=*rcnum*	*rctable:* RS-type address or address in register (2) - (12). *rcnum:* RS-type address or address in register (2) - (12).
,RETCODE=*retcode*	*retcode:* RS-type address or register (2) - (12).
,RSNCODE=*rsncode*	*rsncode:* RS-type address or register (2) - (12).
,PLISTVER=**IMPLIED_VERSION** ,PLISTVER=MAX ,PLISTVER=1 ,PLISTVER=2 ,PLISTVER=3 ,PLISTVER=4 ,PLISTVER=5 ,PLISTVER=6 ,PLISTVER=7	**Default:** PLISTVER=IMPLIED_VERSION
,MF=**S** ,MF=(L,*list addr*) ,MF=(L,*list addr*,*attr*) ,MF=(L,*list addr*,**0D**) ,MF=(E,*list addr*) ,MF=(E,*list addr*,**COMPLETE**)	**Default:** MF=S *list addr:* RS-type address or register (1) - (12).

Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the ATRQUERY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

REQUEST=RMINFO
REQUEST=WMINFO
REQUEST=SYSINFO
REQUEST=URINFO
A required parameter that specifies the requested ATRQUERY function.

You must provide a storage area for the returned information. The AREAADDR, AREALEN, and AREAALET parameters define the storage area. If the storage area is not large enough to hold all the information to be returned, a warning return code is set, along with a reason code to identify the problem. The number of entries is returned in the COUNT field.

REQUEST=RMINFO
Retrieves information about resource managers that are involved with RRS processing. The information includes resource managers that are:

- Currently active with RRS on any systems in the sysplex that match all of the specified filters
- Currently not active with RRS on any system but were logged as last active with RRS on a system and match all of the specified filters

Note: Resource managers that are currently active with RRS/MVS are ones that have exits set with RRS.

When you specify REQUEST=RMINFO, DSECT ATRFZRM in mapping macro ATRFZQRY maps the storage area.

ATRQUERY returns this information sorted alphanumerically with a primary ascending sort key of logging group name and a secondary ascending sort key of system name.

REQUEST=WMINFO
Retrieves information about work managers that are involved with RRS/MVS processing. The information includes work managers that:

- Own contexts that have incomplete units of recovery on this system; or,
- Own or owned contexts that have incomplete units of recovery which are currently not active with RRS on any system, but were last active on this system.

Note: If a native context has an incomplete unit of recovery, the work manager name reported by RRS is reported as the following concatenated string:

- SystemName
- Period (.)
- JobName of the native context address space
- Period (.)
- ASID
- Blanks (padded to 32–bytes)

When you specify REQUEST=WMINFO, DSECT ATRFZWM in mapping macro ATRFZQRY maps the storage area.

REQUEST=SYSINFO
Retrieves information about RRS itself.

When you specify REQUEST=SYSINFO, DSECT ATRFZSI in mapping macro ATRFZQRY maps the storage area.

ATRQUERY returns this information sorted alphanumerically with a primary ascending sort key of logging group name and a secondary ascending sort key of system name.

REQUEST=URINFO
Retrieves information about units of recovery (URs) that are active on systems in the sysplex that match all of the specified filters.

When you specify REQUEST=URINFO, DSECTs ATRFZUR and ATRFZURI in mapping macro ATRFZQRY map the storage area.

,RMNAME=*rmname*
With REQUEST=RMINFO or REQUEST=URINFO, an optional parameter that contains the resource manager name.

Note: The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.

When you specify RMINFO and RMNAME, the system uses the specified value as a filter on the returned information, only returning information about the resource managers matching the pattern you specify for this or any other filters. These resource managers must be:

- Currently active on this system; or,
- Currently inactive, but were active on this system in the past.

When you specify URINFO and RMNAME, the system returns information about the URs associated with the resource manager you name. When you specify URINFO without RMNAME, the system returns information about all currently active resource managers.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-character field that contains a resource manager name. The name can consist of the following printable characters:

- Alphanumeric characters: A-Z and 0-9.
- National characters: $ (X'5B'), # (X'7B'), @ (X'7C').
- The period (.).
- The underscore (_).
- The trailing blank characters needed to fill the 32-byte field.
- The wildcard characters: * and ?.

The name may not start with a blank or contain embedded blanks. Lower case characters are folded to upper case characters.

,WMNAME=*wmname*
With REQUEST=WMINFO or REQUEST=URINFO, an optional parameter that contains the work manager name.

Note: The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.

When you specify WMNAME, the system returns information about the work managers matching the pattern you specify. These work managers must be managing currently incomplete units of recovery that:

- Are currently active on this system; or,
- Were last active on this system.

When you omit WMNAME, the system returns information about all work managers that are managing currently incomplete units of recovery that:

- Are currently active on this system; or,
- Were last active on this system.

When you specify URINFO and WMNAME, the system returns information about the URs associated with the work manager you name. When you specify URINFO without WMNAME, the system returns information about all currently active work managers.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-character field that contains a work manager name. The name can consist of the following printable characters:

- Alphanumeric characters: A-Z and 0-9.
- National characters: $ (X'5B'), # (X'7B'), @ (X'7C').
- The period (.).
- The underscore (_).
- The trailing blank characters needed to fill the 32-byte field.
- The wildcard characters: * and ?.

The name may not start with a blank or contain embedded blanks. Lower case characters are folded to upper case characters.

,GNAME=*gname*

With REQUEST=RMINFO, REQUEST=WMINFO, REQUEST=SYSINFO or REQUEST=URINFO, an optional parameter that contains the logging group name.

Note: The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.

When you specify GNAME, the system uses the specified value as a filter on the returned information, only returning information matching the pattern you specify.

When you omit GNAME, ATRQUERY uses the current system's logging group name as a filter on the information returned. To obtain information for all the systems in a sysplex, specify an asterisk (*) for both GNAME and SYSNAME. To obtain information for all the systems in a specific logging group, specify the logging group name for GNAME and specify an asterisk (*) for SYSNAME.

To code: Specify the RS-type address, or address in register (2)-(12), of a 8-character field that contains a logging group name. The name can consist of the following printable characters:

- Alphanumeric characters: A-Z and 0-9.
- National characters: $ (X'5B'), # (X'7B'), @ (X'7C').
- The period (.).
- The underscore (_).

- The trailing blank characters needed to fill the 8-byte field.
- The wildcard characters: * and ?.

The name may not start with a blank or contain embedded blanks. Lower case characters are folded to upper case characters.

,SYSNAME=*sysname*
With REQUEST=RMINFO, REQUEST=WMINFO, REQUEST=SYSINFO or REQUEST=URINFO, an optional parameter that contains the system name.

Note: The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.

When you specify SYSNAME, the system uses the specified value as a filter on the returned information, only returning information matching the pattern you specify.

When you omit SYSNAME, ATRQUERY uses the current system's system name as a filter on the information returned. To obtain information for all the systems in a sysplex, specify an asterisk (*) for both GNAME and SYSNAME.

To code: Specify the RS-type address, or address in register (2)-(12), of a 8-character field that contains a system name.

,METADATA
With REQUEST=RMINFO, an optional keyword that indicates that if the resource manager has stored Meta Data, it will be returned as part of the resource manager's information. When you omit METADATA, no Meta Data information will be returned.

,ASID=*asid*
When you specify REQUEST=URINFO, an optional input parameter that contains the address space identifier (ASID) of the address space where the application you are interested in is running. The system returns information about URs associated with this address space.

Note: This parameter is left for compatibility with previous releases. IBM recommends the usage of WMNAME instead of this parameter.

To code: Specify the RS-type address, or address in register (2)-(12), of a halfword field.

,URID=*urid*
When you specify REQUEST=URINFO, an optional parameter that contains the identifier of the specific UR for which information is to be returned. (To obtain the UR identifier, call the Express_UR_Interest service.)

Note: URID and URIDSTR are mutually exclusive.

To code: Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

,URIDSTR=*uridstr*
When you specify REQUEST=URINFO, an optional parameter that contains the EBCDIC representation of the 16–character identifier of the specific UR for which information is to be returned.

Note: The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.

Note: URID and URIDSTR are mutually exclusive.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

,SURID=*surid*
When you specify REQUEST=URINFO, an optional parameter that contains the sysplex UR identifier of the specific cascaded UR family for which information is to be returned. You can obtain a sysplex UR identifier from a log entry in the Restart, Main, or Delayed log stream or from a prior ATRQUERY URINFO macro invocation.

Note: SURID and SURIDSTR are mutually exclusive.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

,SURIDSTR=*suridstr*
When you specify REQUEST=URINFO, an optional parameter that contains the EBCDIC representation of the 32–character sysplex UR identifier of the specific cascaded UR family for which information is to be returned.

Note: The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.

Note: SURID and SURIDSTR are mutually exclusive.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64-character field.

,URTYPE=*urtype*
When you specify REQUEST=URINFO, an optional input parameter that contains the type of UR for which information is to be returned.

Use one of the following constants:

ATR_NOFILTER
The UR type will not be used to filter the returned information.

ATR_PROTECTED
The system will return all URs in which at least one resource manager has a protected interest.

ATR_UNPROTECTED
The system will return all URs in which no resource manager has a protected interest.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,URSTATE=*urstate*
When you specify REQUEST=URINFO, an optional parameter that specifies the state that a given UR or interest in a given UR must be in to be returned.

Note: URSTATE and URSTMASK are mutually exclusive keywords.

Use one of the following values:

ATR_NOFILTER
The UR state will not be used to filter the returned information.

ATR_IN_FLIGHT
In-Flight URs only.

ATR_IN_STATE_CHECK
In-State_Check URs only.

ATR_IN_PREPARE
In-Prepare URs only.

ATR_IN_DOUBT
In-Doubt URs only.

ATR_IN_COMMIT
In-Commit URs only.

ATR_IN_BACKOUT
In-Backout URs only.

ATR_IN_END
In-End URs only.

ATR_COMPLETE
Complete URs only.

Note: This parameter is left for compatibility. IBM recommends the usage of the ATR_IN_COMPLETION parameter.

ATR_IN_COMPLETION
In-Completion URs only.

ATR_IN_ONLY_AGENT
In-Only-Agent URs only.

ATR_IN_FORGET
In-Forget URs only.

To code: Specify the value, or register (2)-(12) that contains the value.

,URSTMASK=*urstmask*
When you specify REQUEST=URINFO, an optional parameter that specifies a 32–bit mask identifying one or more states of the URs or interests in the URs to be returned.

Note: URSTATE and URSTMASK are mutually exclusive keywords.

The following bit constants can be used to create the fullword bit mask:

ATR_NOFILTER_MASK (X'00000000')
The UR state will not be used to filter the returned information.

ATR_IN_FLIGHT_MASK (X'80000000')
Include In-Flight URs.

ATR_IN_STATE_CHECK_MASK (X'40000000')
Include In-State_Check URs.

ATR_IN_PREPARE_MASK (X'20000000')
Include In-Prepare URs.

ATR_IN_DOUBT_MASK (X'1000000')
Include In-Doubt URs.

ATR_IN_COMMIT_MASK (X'08000000')
Include In-Commit URs.

ATR_IN_BACKOUT_MASK (X'04000000')
Include In-Backout URs.

ATR_IN_END_MASK (X'02000000')
Include In-End URs.

ATR_IN_COMPLETION_MASK (X'01000000')
Include In-Completion URs.

ATR_IN_ONLY_AGENT_MASK (X'00800000')
Include In-Only-Agent URs.

ATR_IN_FORGET_MASK (X'00400000')
Include In-Forget URs.

To code: Specify the value, or register (2)-(12) that contains the value. For example, if the value passed via this parameter is X'0C000000', both the ATR_IN_COMMIT_MASK and the ATR_IN_BACKOUT_MASK are specified. URs that are in commit or backout are returned. This definition might be specified:

```
CMT_AND_BAK    EQU(ATR_IN_COMMIT_MASK+ATR_IN_BACKOUT_MASK)
```

,LUWID=*luwid*
When you specify REQUEST=URINFO, an optional input parameter that specifies the logical unit of work identifier (LUWID) to be used to filter the returned information.

Note: LUWID and LUWIDSTR are mutually exclusive keywords.

To code: Specify the RS-type address, or address in register (2)-(12), of a 26-character field. A LUWID has the following format:

```
netid.luname.instnum.seqnum
```

The fields are as follows:

netid.luname
1-17 character identifier of the network and LU, preceded by a 1-byte length field

instnum
6-byte TP instance

seqnum
2-byte sequence number

If LUWID and LUWIDSTR are left unspecified, then the LUWID will not be used as a filter. If LUWIDSTR is specified, but the instnum or the seqnum (or both) is left unspecified, then that unspecified portion of the LUWID is not used as a filter.

,LUWIDSTR=*luwidstr*
When you specify REQUEST=URINFO, the address of an optional 35–character EBCDIC representation of the LUWID with commas or blanks used to delimit the `netid.luname`, `instnum`, and `seqnum` fields.

Note: The netid.luname string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character. The instnum and seqnum are optional. However, if the

seqnum is specified without the instnum, the delimiter between instnum and seqnum is still required. For example: `ABC.LUNAME,,12`

Here are some more examples of valid LUWIDSTR pattern specifications:

```
*.*,99,12
ABC.*
ABC.*,12
ABC.*,,12
A?C.LUNA??
```

If LUWID and LUWIDSTR are left unspecified, then the LUWID will not be used as a filter. If LUWIDSTR is specified, but the instnum or the seqnum (or both) is left unspecified, then that unspecified portion of the LUWID is not used as a filter.

Note: LUWID and LUWIDSTR are mutually exclusive keywords.

To code: Specify the RS-type address, or address in register (2)-(12), of a 35-character field.

,TID=*tid*
When you specify REQUEST=URINFO, an optional input parameter that contains the transaction identifier (TID) to be used to filter the returned information. If you specify TID=ATR_NOFILTER, this keyword is ignored.

Note: TID may not be specified if either TIDLOW or TIDHIGH is specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-byte field that contains the TID part of the Enterprise identifier (EID). An EID has the following format:

```
tidgtid
```

,TIDLOW=*tidlow*
When you specify REQUEST=URINFO, an optional input parameter that specifies that URs having transaction identifiers (TID) that are equal to or higher than one specified are returned. The low range indicator is set by default to be a value lower than, or equal to, all transaction identifiers. If TIDLOW and TIDHIGH are both specified, URs that have transaction identifiers that are between the TIDLOW and TIDHIGH values, inclusive, are returned.

Note: TID may not be specified if either TIDLOW or TIDHIGH is specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-byte field that contains the lowest desired TID part of the Enterprise identifier (EID). An EID has the following format:

```
tidgtid
```

,TIDHIGH=*tidhigh*
When you specify REQUEST=URINFO, an optional input parameter that specifies that URs having transaction identifiers (TID) that are equal to or lower than one specified are returned. The high range indicator is set by default to be a value higher than, or equal to, all transaction identifiers. If TIDLOW and TIDHIGH are both specified, URs that have transaction identifiers that are between the TIDLOW and TIDHIGH values, inclusive, are returned.

Note: TID may not be specified if either TIDLOW or TIDHIGH is specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-byte field that contains the highest desired TID part of the Enterprise identifier (EID). An EID has the following format:

```
tidgtid
```

,GTID=*gtid*
When you specify REQUEST=URINFO, an optional input parameter that contains the global transaction identifier (GTID) to be used to filter the returned information. If you specify GTID=ATR_NOFILTER, this keyword is ignored.

Note: GTID and GTIDSTR are mutually exclusive keywords.

To code: Specify the RS-type address, or address in register (2)-(12), of a 40-byte field that contains the GTID part of the Enterprise identifier (EID). An EID has the following format:

```
tidgtid
```

,GTIDSTR=*gtidstr*
When you specify REQUEST=URINFO, the address of an optional 80–character input that contains the EBCDIC representation of the 40–character GTID used to filter the returned information.

Note: The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.

Note: GTID and GTIDSTR are mutually exclusive keywords.

To code: Specify the RS-type address, or address in register (2)-(12), of an 80-byte field that contains the EBCDIC representation of the GTID part of the Enterprise identifier (EID). An EID has the following format:

```
tidgtid
```

,XID=*xid*
When you specify REQUEST=URINFO, the address of an optional 140–character input that contains the X/Open Identifier (XID) used to filter the returned information.

The default is 0. The XID will not be used to filter the returned information.

Note: XID may not be specified if either XIDFORMATID, XIDGTRIDSTR, or XIDBQUALSTR is specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a 140-byte field that contains the X/Open Identifier (XID). An XID has the following format:

```
FormatIDGtrid_lengthBqual_lengthID
```

,XIDFORMATIDSTR=*xidformatidstr*
When you specify REQUEST=URINFO, the address of the EBCDIC representation of an optional 8–character input that contains the 4–character FormatID portion of the X/Open Identifier (XID) used to filter the returned information.

The default is 0. The XIDFORMATIDSTR will not be used to filter the returned information.

Note:

1. The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.
2. XID may not be specified if either XIDFORMATID, XIDGTRIDSTR, or XIDBQUALSTR is specified.

To code: Specify the RS-type address, or address in register (2)-(12), of the EBCDIC representation of a 4-byte field that contains the FormatID portion of the X/Open Identifier (XID).

,XIDGTRIDSTR=*xidgtridstr*

When you specify REQUEST=URINFO, the address of the EBCDIC representation of an optional 128–character input that contains the 64–character GTRID portion of the X/Open Identifier (XID) used to filter the returned information.

The default is 0. The XIDGTRIDSTR will not be used to filter the returned information.

Note:

1. The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.
2. XID may not be specified if either XIDFORMATID, XIDGTRIDSTR, or XIDBQUALSTR is specified.

To code: Specify the RS-type address, or address in register (2)-(12), of the EBCDIC representation of a 64-byte field that contains the GTRID portion of the X/Open Identifier (XID).

,XIDBQUALSTR=*xidbqualstr*

When you specify REQUEST=URINFO, the address of the EBCDIC representation of an optional 128–character input that contains the 64–character BQUAL portion of the X/Open Identifier (XID) used to filter the returned information.

The default is 0. The XIDBQUALSTR will not be used to filter the returned information.

Note:

1. The character string you specify may include wildcard characters. An asterisk (*) represents any string having a length of zero or more characters. A question mark (?) represents a position which may contain any single character.
2. XID may not be specified if either XIDFORMATID, XIDGTRIDSTR, or XIDBQUALSTR is specified.

To code: Specify the RS-type address, or address in register (2)-(12), of the EBCDIC representation of a 64-byte field that contains the BQUAL portion of the X/Open Identifier (XID).

,TODBEG=*todbeg*

When you specify REQUEST=URINFO, an optional 8–byte begin range time of day (UTC) value used to filter the returned information. URs which were created at the same time or later than the time specified by this parameter will be returned. If this parameter is not specified, the starting TOD range is set to the UTC time of the oldest UR.

Note: If TODBEG and TODEND are both specified, URs that have create times that are between the TODBEG and TODEND values, inclusive, are returned.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-byte field.

,TODEND=*todend*
When you specify REQUEST=URINFO, an optional 8–byte end range time of day (UTC) value used to filter the returned information. URs which were created at the same time or earlier than the time specified by this parameter will be returned. If this parameter is not specified, the ending TOD range is set to the current UTC time.

Note: If TODBEG and TODEND are both specified, URs that have create times that are between the TODBEG and TODEND values, inclusive, are returned.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-byte field.

,CURDUR=*curdur*
When you specify REQUEST=URINFO, an optional 13–byte field character string in the format `hh:mm:ss.ssss` containing the minimum duration that a given UR has been in its current state. If this parameter is not specified, URs will be returned regardless of how long they have been in their current state.

To code: Specify the RS-type address, or address in register (2)-(12), of a 13-byte field.

,DEFURONLY=*defuronly*
When you specify REQUEST=URINFO, an optional 3–character string used to determine if only the URs that are marked deferred should be returned. There are two possible strings:

- `NO` — All URs will be returned, including those marked deferred.
- `YES` — Only URs marked deferred will be returned.

The default is `NO` .

Note: This parameter is left for compatibility. IBM recommends the usage of the EXCLMASK parameter. DEFURONLY and EXCLMASK are mutually exclusive.

To code: Specify the RS-type address, or address in register (2)-(12), of a 3–character field.

,EXCLMASK=*exclmask*
When you specify REQUEST=URINFO, an optional parameter that specifies the address of a fullword containing the 32–bit mask identifying one or more conditions that identify URs to be excluded from the returned information.

Note: DEFURONLY and EXCLMASK are mutually exclusive keywords.

The following bit constants can be used to create the fullword bit mask:

ATR_NOEXCLUSION_MASK (X'00000000')
No exclusion conditions will be used to filter the returned information.

ATR_UR_IN_RESTART_MASK (X'80000000')
URs that are read from the Restart log will be excluded from the returned information.

ATR_RESTART_REQUIRED_MASK (X'40000000')
URs that contain interests whose resource managers are required to restart with RRS will be excluded from the returned information.

ATR_NONDEFERRED_UR_MASK (X'20000000')
URs that are not deferred from an RRS server task will be excluded from the returned information. This specification is equivalent to specifying the DEFURONLY keyword.

ATR_CASCADED_UR_MASK (X'10000000')
URs that are included only because they are part of a cascaded UR family will be excluded from the returned information.

ATR_LOCAL_UR_MASK (X'08000000')
URs in local transaction mode will be excluded from the returned information.

ATR_GLOBAL_UR_MASK (X'04000000')
URs in global transaction mode or hybrid-global transaction mode will be excluded from the returned information.

ATR_RRS_MANAGED_UR_MASK (X'02000000')
URs whose current work manager is RRS will be excluded from the returned information. RRS is a work manager for restart URs and for URs associated with contexts delegated to RRS.

ATR_NONRRS_MANAGED_UR_MASK (X'01000000')
URs whose current work manager is not RRS will be excluded from the returned information.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field. For example, if the address passed via this parameter points to a fullword that contains a value of X'C0000000', both the ATR_RESTART_LOG_MASK and the ATR_RESTART_REQUIRED_MASK are specified. Neither URs that are read from the Restart log nor URs that contain interests that have RMs that are required to restart will be returned. This definition might be specified:

```
REST_AND_RQRD    DC  AL4(ATR_UR_IN_RESTART_MASK+ATR_RESTART_REQUIRED_MASK)
```

,EXINFOMASK=*exinfomask*
When you specify REQUEST=URINFO, an optional parameter that specifies the address of a fullword containing the 32–bit mask identifying which types of extended information will be returned for the returned units of recovery.

The following bit constants can be used to create the fullword bit mask:

ATR_NOXINFO_MASK (X'00000000')
No extended information will be returned.

ATR_XINFO_TIMESTAMP_MASK (X'80000000')
Timestamp information for each unit of recovery as well as each interest in each unit of recovery will be added to the returned information.

ATR_XINFO_CASCADED_MASK (X'40000000')
Cascaded family information for each unit of recovery will be added to the returned information. If any UR in a cascaded UR family is returned, all of the URs in the family will be returned, unless EXCLUDEURMASK indicates that they should be excluded.

ATR_XINFO_PID_MASK (X'20000000')
Persistent Interest Data for each unit of recovery interest will be added to the returned information.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field containing the desired mask.

,SORTTAB=*sorttab*
When you specify REQUEST=URINFO, an optional row-major sort table array. Each element in the table is comprised of a fullword sort key selector and a 32–bit sort options field. Each element in the table represents a sort key; therefore, a table with multiple elements specifies a multiple key sort of the returned information. The position of a given sort element in the sort table array specifies its sort priority. The first element in the table specifies the primary sort key. The second element specifies the secondary sort key. The third specifies the tertiary sort key, and so on. A mapping of the element format is provided in the ATRFZQRY mapping macro.

Note: If SORTTAB is specified, SORTNUM must also be specified.

Example table:

Table 56. ATRQUERY — SORTTAB Parameter

Sort order	Sort key	Sort option
Primary sort key	`DC F(ATR_SORT_WM_NAME)`	`DC F(ATR_SORTOPT_ASCENDING)`
Secondary sort key	`DC F(ATR_SORT_CREATE_TIME)`	`DC F(ATR_SORTOPT_DESCENDING)`
Tertiary sort key	`DC F(ATR_SORT_URID)`	`DC F(ATR_SORTOPT_ASCENDING)`

The SORTNUM for this table would be 3.

Sort Key Selector Values:

ATR_SORT_WM_NAME (X'00000001')
Sort returned information based on Work Manager Name.

ATR_SORT_URID (X'00000002')
Sort returned information based on UR Identifier.

ATR_SORT_CREATE_TIME (X'00000003')
Sort returned information based on the create time of the UR.

ATR_SORT_UR_STATE (X'00000004')
Sort returned information based on the UR state.

ATR_SORT_LUWID (X'00000005')
Sort returned information based on the Logical Unit Work Identifier.

Note: ATR_SORT_LUWID is really a three key sort, with the primary key being the luname.netid, the secondary key being the instnum, and the tertiary key being the seqnum. The luname.netid is padded to 17 bytes with blanks on the right before comparing the IDs for sort order.

ATR_SORT_TID (X'00000006')
Sort returned information based on the Transaction Identifier.

ATR_SORT_GTID (X'00000007')
Sort returned information based on the Global Transaction Identifier.

ATR_SORT_XID (X'00000008')
Sort returned information based on the X/Open Identifier.

Note: ATR_SORT_XID is really a three key sort, with the primary key being the FormatID, the secondary key being the Global Transaction ID, and the tertiary key being the BQUAL.

ATR_SORT_GNAME (X'00000009')
Sort returned information based on the logging group name.

ATR_SORT_SYSNAME (X'0000000A')
Sort returned information based on the system name.

ATR_SORT_SURID (X'0000000B')
Sort returned information based on the sysplex UR identifier.

Sort Option Values:

ATR_SORTOPT_ASCENDING (X'0nnnnnnn')
Sort returned information in ascending order using the sort key.

ATR_SORTOPT_DESCENDING (X'8nnnnnnn')
Sort returned information in descending order using the sort key.

Note: Bits 1–31 are reserved, and must contain binary zeros.

To code: Specify the RS-type address, or address in register (2)-(12), of a sort table array.

,SORTNUM=*sortnum*
When you specify REQUEST=URINFO, an optional fullword unsigned integer value containing the number of elements in the sort table passed via the SORTTAB parameter. The number of elements that can be passed in the sort table is limited to the number of possible sort fields.

Note: SORTNUM must be specified if SORTTAB is specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword unsigned integer value.

,AREAADDR=*areaaddr*
A required input parameter that contains the address of the storage area to contain the returned information. Specify the length of the area in AREALEN.

To code: Specify the RS-type address, or address in register (2)-(12), of a pointer field.

,AREALEN=*arealen*
A required input parameter that contains the length of the area to contain the returned information. Specify the address of the area in AREAADDR.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,AREAALET=*areaalet*
An optional input parameter that contains the ALET of an entry on the DU Access List that will be used to address the area to contain the returned information. Specify the address of the area in AREAADDR.

Note: ALETs of access list entries on the PASN access list and the special ALET value of 1 (secondary) are not valid.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,AREAOPT=*areaopt*
An optional input parameter that specifies the type of addressing used in the control structures returned in the area specified by AREAADDR.

To code: Specify either:

AREAOPT=BYADDR
Complete addresses are to be returned in the control structures.

AREAOPT=BYOFFSET
Offsets from the beginning of the storage area are to be returned in the control structures.

The default is BYADDR.

,COUNT=*count*
A required output parameter that, upon return from the service, will contain the number of entries that match your request.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,RCTABLE=*rctable*
An optional 4–byte input parameter that contains the address of a "2304" area to contain the return code and reason code from each system and logging group queried. The area consists of up to 64 36–byte entries. DSECT ATRFZRCA in mapping macro ATRFZQRY maps the storage area. The number of entries will be returned in the RCNUM parameter. The table will only contain valid information if the reason code returned to you is ATRSRV_REMOTE_WARNING or ATRSRV_REMOTE_ERROR.

The default is RCTABLE=0. No return code table is desired.

To code: Specify the RS-type address, or address in register (2)-(12), of a "2304" area.

,RCNUM=*rcnum*
When using RCTABLE, a required output parameter that will contain the number of entries in the RCTABLE area.

To code: Specify the RS-type address of a fullword field, or register (2)-(12).

,RETCODE=*retcode*
An optional output parameter into which the system copies the return code from GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12).

,RSNCODE=*rsncode*
An optional output parameter into which the system copies the reason code from GPR 0.

To code: Specify the RS-type address of a fullword field, or register (2)-(12).

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=1
,PLISTVER=2
,PLISTVER=3
,PLISTVER=4
,PLISTVER=5
,PLISTVER=6

,PLISTVER=7

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **1**, if you use the following parameters:
  - AREAADDR
  - AREALEN
  - ASID
  - COUNT
  - GTID
  - LUWID
  - REQUEST
  - RMNAME
  - TID
  - URID
  - URSTATE
  - URTYPE
- **2**, if you use the parameters from PLISTVER=1 plus the following parameters:
  - DEFURONLY
- **3**, if you use the parameters from PLISTVER=2 plus the following parameters:
  - CURDUR
  - EXCLMASK
  - EXINFOMASK
  - GTIDSTR
  - LUWIDSTR
  - SORTNUM
  - SORTTAB
  - TIDHIGH
  - TIDLOW
  - TODBEG
  - TODEND
  - URIDSTR
  - URSTMASK

- **4**, if you use the parameters from PLISTVER=3 plus the following parameters:
  - AREAALET
  - AREAOPT
  - GNAME
  - RCNUM
  - RCTABLE
  - SYSNAME
- **5**, if you use the parameters from PLISTVER=4 plus the following parameters:
  - XID
  - XIDFORMATIDSTR
  - XIDGTRIDSTR
  - XIDBQUALSTR
- **6**, if you use the parameters from PLISTVER=5 plus the following parameters:
  - SURID
  - SURIDSTR
- **7**, if you use the parameters from PLISTVER=6 plus the following parameters:
  - ATRxinfo
  - PID_Mask
- **8**, if you use the parameters from PLISTVER=7 plus the following parameters:
  - METADATA

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 1
- A decimal value of 2
- A decimal value of 3
- A decimal value of 4
- A decimal value of 5
- A decimal value of 6
- A decimal value of 7

,MF=S
,MF=(L,*list addr***)**
,MF=(L,*list addr***,***attr***)**
,MF=(L,*list addr***,0D)**
,MF=(E,*list addr***)**
,MF=(E,*list addr***,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an in-line parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The

list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,*list addr*
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*
An optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *attr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

Note: Declarations for Bit constants are provided in the ATRFZQRY mapping macro interface.

ABEND codes

None.

Return and reason codes

When the ATRQUERY macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code:

  0 ATRQUERY_SUCCESS — request completed successfully

  4 ATRQUERY_WARNING — request completed with a warning condition

  8 ATRQUERY_FAILURE — request failed

- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code. The return and reason codes are shown in the following table. The ATRFZQRY mapping macro provides the equate symbols for the return and reason codes.

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
4	3	**Equate symbol:** ATRQUERY_AREA_FULL **Meaning:** The return area is too small to hold all the information to be returned. **Action:** Increase the size of the return area and retry the request.

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
4	17	**Equate symbol:** ATRQUERY_REMOTE_WARNING **Meaning:** Errors occurred while querying data on at least one system. Some information may not have been returned. If you have been provided with an RCTABLE, it will contain detailed error information for each system involved in your request. **Action:** Fix the errors noted in the RCTABLE and retry the request.
4	29	**Equate symbol:** ATRQUERY_RMMETADATALOGUNAVAILABLE **Meaning:** The RM Meta Data log is not available. No information was returned. **Action:** Check SYSLOG for messages ATR132I or ATR172E that will further explain why the log is unavailable. Retry the request when the RM Meta Data log is available.
8	1	**Equate symbol:** ATRQUERY_ASID_INVALID **Meaning:** You specified an address space identifier (ASID) that does not exist. **Action:** Specify a valid ASID and retry the request.
8	2	**Equate symbol:** ATRQUERY_RRS_NOT_ACTIVE **Meaning:** The ATRQUERY service is not available; RRS is not active on this system. **Action:** Retry the request when RRS is available.
8	4	**Equate symbol:** ATRQUERY_NOT_SAF_AUTH **Meaning:** The caller does not have RACF authority to issue RRS commands. **Action:** Obtain READ access to the MVSADMIN.RRS.COMMANDS resource in the FACILITY class and retry the request.
8	5	**Equate symbol:** ATRQUERY_URSTMASK_RSVD **Meaning:** Reserved fields in the UR State Mask were nonzero. **Action:** Correct the mask, clearing the reserved fields, and retry the request.

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	6	**Equate symbol:** ATRQUERY_XINFOMASK_RSVD **Meaning:** Reserved fields in the Extended Information Mask were nonzero. ATR_XINFO_PID_MASK is set but the PLISTVER specified is less than 7. When requesting for Persistent Interest Data, PLISTVER must be 7 or higher. **Action:** Correct the mask, PLISTVER or clearing the reserved fields, and retry the request.
8	7	**Equate symbol:** ATRQUERY_EXCLUSIONMASK_RSVD **Meaning:** Reserved fields in the UR Exclusion Condition Mask were nonzero. **Action:** Correct the mask, clearing the reserved fields, and retry the request.
8	8	**Equate symbol:** ATRQUERY_SORTKEY_INVALID **Meaning:** The sort table passed contained an invalid sort key. **Action:** Correct the sort table and retry the request.
8	9	**Equate symbol:** ATRQUERY_SORTOPT_INVALID **Meaning:** The sort table passed contained an invalid option. **Action:** Correct the sort table and retry the request.
8	A	**Equate symbol:** ATRQUERY_SORTNUM_INVALID **Meaning:** The sort element number passed is not valid. **Action:** Correct the number of elements in the sort table and retry the request.
8	B	**Equate symbol:** ATRQUERY_TID_RANGE_INVALID **Meaning:** The value specified for the TIDLOW parameter is higher than the value specified for the TIDHIGH parameter. **Action:** Correct either the TIDLOW value or the TIDHIGH value, and retry the request.
8	C	**Equate symbol:** ATRQUERY_TOD_RANGE_INVALID **Meaning:** The time specified for the TODBEG parameter is after the time specified for the TODEND parameter. **Action:** Correct either the TODBEG value or TODEND value, and retry the request.

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	D	**Equate symbol:** ATRQUERY_LUWIDSTR_INVALID **Meaning:** The value passed via the LUWIDSTR parameter is not valid. **Action:** Correct the LUWIDSTR value and retry the request.
8	E	**Equate symbol:** ATRQUERY_CURDUR_INVALID **Meaning:** The value passed via the CURDUR parameter is not valid. **Action:** Correct the CURDUR value and retry the request.
8	F	**Equate symbol:** ATRQUERY_SORTTABPTR_BAD **Meaning:** The sort table entry count was nonzero, but a valid table pointer was not passed. **Action:** Correct the table pointer value and retry the request.
8	10	**Equate symbol:** ATRQUERY_AREAALET_SECONDARY **Meaning:** The AREAALET specified is the secondary ALET, which is not accessible by the ATRQUERY service routine. **Action:** Correct the AREAALET value and retry the request.
8	11	**Equate symbol:** ATRQUERY_REQUEST_UNKNOWN **Meaning:** A downlevel version of RRS did not understand the request. **Action:** Correct the parameter value and retry the request.
8	12	**Equate symbol:** ATRQUERY_GNAME_INVALID **Meaning:** The value specified for the GNAME parameter is not valid. **Action:** Correct the GNAME value and retry the request.
8	13	**Equate symbol:** ATRQUERY_SYSNAME_INVALID **Meaning:** The value specified for the SYSNAME parameter is not valid. **Action:** Correct the SYSNAME value and retry the request.
8	14	**Equate symbol:** ATRQUERY_RESOURCE_ERROR **Meaning:** No buffer was available to satisfy a remote request. **Action:** Retry the request.

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	15	**Equate symbol:** ATRQUERY_TOO_MANY_ITEMS **Meaning:** There are too many data items to fit in the buffer to be sorted. **Action:** Add more filters and retry the request.
8	16	**Equate symbol:** ATRQUERY_INSTANCE_FAILURE **Meaning:** An error occurred processing a remote request. Your RCTABLE, if one was provided, will contain the failing service identifier, return code, and reason code. Service identifiers: **1** IXCMSGO macro interface **2** IXCMSGI macro interface Any other value is an internal error and should be reported to the IBM Support center. **Action:** Fix the errors noted in the RCTABLE and retry the request.
8	18	**Equate symbol:** ATRQUERY_REMOTE_ERROR **Meaning:** Errors occurred while querying data on all systems. No information was returned. If you received an RCTABLE, it will contain detailed error information for each system involved in your request. **Action:** Fix the errors noted in the RCTABLE and retry the request.
8	19	**Equate symbol:** ATRQUERY_RESP_NOT_RECEIVED **Meaning:** No response was received from the remote system. No information was returned. **Action:** Retry the request.
8	1A	**Equate symbol:** ATRQUERY_REMOTE_NOT_ACTIVE **Meaning:** The remote system is not active or RRS is not active on the remote system. No information was returned. **Action:** Retry the request once the remote system is available.
8	1B	**Equate symbol:** ATRQUERY_AREALEN_INVALID **Meaning:** The value specified for the AREALEN parameter is not valid. No information was returned. **Action:** Correct the AREALEN value and retry the request.

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	1C	**Equate symbol:** ATRQUERY_AREAADDR_INVALID **Meaning:** The value specified for the AREAADDR parameter is not valid. No information was returned. **Action:** Correct the AREAADDR value and retry the request.
8	FFF	**Equate symbol:** ATRQUERY_UNEXPECTED_ERROR **Meaning:** An unexpected system error occurred. **Action:** Contact your system support.

Examples

The following examples show several uses of the ATRQUERY macro.

Example 1

To obtain information about all resource managers known to RRS, issue the following request:

```
         ATRQUERY REQUEST=RMINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
               COUNT=RETCNT
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)              ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                 LENGTH OF RETURN AREA
*
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED RM COUNT
RETAREA  DS     1024C                   RETURN AREA
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZQRY
```

Example 2

To obtain information about the resource manager named UTRICK, issue the following request:

```
         ATRQUERY REQUEST=RMINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
               COUNT=RETCNT,RMNAME=MYRM
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)              ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                 LENGTH OF RETURN AREA
MYRM     DC     CL32'UTRICK'            RESOURCE MANAGER NAME
*
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED RM COUNT
RETAREA  DS     1024C                   RETURN AREA
```

```
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZQRY
```

Example 3

To obtain information about all URs, issue the following request:

```
         ATRQUERY REQUEST=URINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
               COUNT=RETCNT
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)              ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                 LENGTH OF RETURN AREA
*
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED RM COUNT
RETAREA  DS     1024C                   RETURN AREA
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZQRY
```

Example 4

To obtain information about all protected URs, issue the following request.

```
         ATRQUERY REQUEST=URINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
               COUNT=RETCNT,URTYPE=ATR_PROTECTED
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)              ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                 LENGTH OF RETURN AREA
*
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED RM COUNT
RETAREA  DS     1024C                   RETURN AREA
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZQRY
         ATRRASM
```

Example 5

To obtain information about the URs associated with the application running in address space 52, issue the following request:

```
         ATRQUERY REQUEST=URINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
               COUNT=RETCNT,ASID=MYASID
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)              ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                 LENGTH OF RETURN AREA
```

```
MYASID   DC     XL2'52'                 ADDRESS SPACE IDENTIFIER
*
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED RM COUNT
RETAREA  DS     1024C                   RETURN AREA
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZQRY
         ATRRASM
```

Example 6

To return only those URs that are In-Doubt or In-Commit, issue the following request:

```
         L   10,URSTMASK
         ATRQUERY REQUEST=URINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
               COUNT=RETCNT,URSTMASK(10),PLISTVER=MAX                  X
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)              ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                 LENGTH OF RETURN AREA
URSTMASK DC     A(STATES)               UR STATE MASK
STATES   EQU    ATR_IN_DOUBT_MASK+ATR_IN_COMMIT_MASK
*
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED RM COUNT
RETAREA  DS     1024C                   RETURN AREA
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZQRY
         ATRRASM
```

Example 7

To return URs sorted by UR create time and UR state, in ascending order, issue the following request:

```
         ATRQUERY REQUEST=URINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
               COUNT=RETCNT,PLISTVER=MAX,                              X
               SORTTAB=STAB@,SORTNUM=SORTNUM
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)              ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                 LENGTH RETURN AREA
URSTMASK DC     A(STATES)               UR STATE MASK
STATES   EQU    ATR_IN_DOUBT_MASK+ATR_IN_COMMIT_MASK
*
STAB@    DC     A(SORTTAB)              SORT TABLE ADDRESS
SORTNUM  DC     F'2'                    ENTRIES IN SORT TABLE
SORTTAB  DS     0F                      SORT TABLE-WORD BOUNDARY
         DC     A(ATR_SORT_CREATE_TIME)  PRIMARY KEY
         DC     A(ATR_SORTOPT_ASCENDING) PRIMARY KEY OPTION
         DC     A(ATR_SORT_UR_STATE)     SECONDARY KEY
         DC     A(ATR_SORTOPT_ASCENDING) SECONDARY KEY OPTION
*
```

```
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED RM COUNT
RETAREA  DS     1024C                   RETURN AREA
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZQRY
         ATRRASM
```

Example 8

To return URs that have been In-Doubt more than 5 seconds, issue the following request:

```
         ATRQUERY REQUEST=URINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
               COUNT=RETCNT,PLISTVER=MAX,                              X
               SORTTAB=STAB@,SORTNUM=SORTNUM
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)              ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                 LENGTH OF RETURN AREA
URSTMASK DC     A(STATES)               UR STATE MASK
STATES   EQU    ATR_IN_DOUBT_MASK+ATR_IN_COMMIT_MASK
*
STAB@    DC     A(SORTTAB)              SORT TABLE ADDRESS
SORTNUM  DC     F'2'                    ENTRIES IN SORT TABLE
SORTTAB  DS     0F                      SORT TABLE-WORD BOUNDARY
         DC     A(ATR_SORT_CREATE_TIME) PRIMARY KEY
         DC     A(ATR_SORTOPT_ASCENDING) PRIMARY KEY OPTION
         DC     A(ATR_SORT_UR_STATE)    SECONDARY KEY
         DC     A(ATR_SORTOPT_ASCENDING) SECONDARY KEY OPTION
*
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED RM COUNT
RETAREA  DS     1024C                   RETURN AREA
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZQRY
         ATRRASM
```

Example 9

To return all the work manager names in the system, issue the following request:

```
         ATRQUERY REQUEST=WMINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
               COUNT=RETCNT,PLISTVER=MAX                               X
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)              ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                 LENGTH OF RETURN AREA
*
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED RM COUNT
RETAREA  DS     1024C                   RETURN AREA
*
* REQUIRED MAPPINGS
*
```

```
        CVT DSECT=YES
        IHAECVT
        ATRFZQRY
        ATRRASM
```

Example 10

Issue the following request to obtain information about URs that meet these conditions:

- The URs are either In-Doubt or In-Commit.
- RMs with names matching the pattern "UTR*" have interests in the URs.
- The URs are sorted by LUWID from lowest to highest; and, in case of ties, sorted by UR creation time from oldest to newest.

```
        L   10,URSTMASK
        ATRQUERY REQUEST=URINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
              RMNAME=MYRM,COUNT=RETCNT,PLISTVER=MAX,                  X
              URSTMASK=(10),SORTTAB=STABPTR,SORTNUM=SORTNUM
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)               ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                  LENGTH OF RETURN AREA
MYRM     DC     CL32'UTR*'               RM NAME PATTERN
URSTMASK DC     A(STATES)
SORTNUM  DC     F'2'                     ENTRIES IN SORT TABLE
*
SORTTAB  DS     0F                        SORT TABLE-WORD BOUNDARY
SRTKEY1  DC     A(ATR_SORT_LUWID)         PRIMARY KEY
SRTOPT1  DC     A(ATR_SORTOPT_ASCENDING)  PRIMARY KEY OPTION
SRTKEY2  DC     A(ATR_SORT_CREATE_TIME)   SECONDARY KEY
SRTOPT2  DC     A(ATR_SORTOPT_DESCENDING) SECONDARY KEY OPTION
*
STABPTR  DC     A(SORTTAB)
STATES   EQU    ATR_IN_DOUBT_MASK+ATR_IN_COMMIT_MASK
*
* OUTPUT AREAS
*
RETCNT   DS     F                        RETURNED RM COUNT
RETAREA  DS     1024C                    RETURN AREA
*
* REQUIRED MAPPINGS
*
        CVT DSECT=YES
        IHAECVT
        ATRFZQRY
        ATRRASM
```

Example 11

Issue the following request to obtain information about URs that meet these conditions:

- The URs are In-Doubt.
- The URs are in logging group PLEX1 on any system in that logging group.

```
        ATRQUERY REQUEST=URINFO,AREAADDR=AREA@,AREALEN=AREAL,         X
              GNAME=MYGROUP,SYSNAME=ALLSYS,PLISTVER=MAX,              X
              RCTABLE=MYRC,RCNUM=MYRCNUM
*
* INPUT VALUES
*
AREA@    DC     A(RETAREA)               ADDRESS OF RETURN AREA
AREAL    DC     F'1024'                  LENGTH OF RETURN AREA
MYGROUP  DC     CL8'PLEX1'               LOGGING GROUP
```

```
ALLSYS   DC     CL8'*'                  ALL SYSTEMS
*
* OUTPUT AREAS
*
RETCNT   DS     F                       RETURNED COUNT
RETAREA  DS     1024C                   RETURN AREA
MYRC     DC     CL2304                  RCTABLE
MYRCNUM  DC     F                       RCTABLE ENTRIES
*
* REQUIRED MAPPINGS
*
        CVT DSECT=YES
        IHAECVT
        ATRFZQRY
        ATRRASM
```

Chapter 12. ATRSRV — Resolve Units of Recovery

The ATRSRV macro allows authorized callers to:

- Remove a resource manager's interest in a UR
- Resolve a UR state from **In-doubt** to **In-Commit**.
- Resolve a UR state from **In-doubt** to **In-Backout**.
- Remove a Resource Manager's identity
- Unregister a Resource Manager's involvement with RRS
- Perform Forget processing on a UR whose In-Doubt: condition has been resolved.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state and PSW key 0-7 or RACF authority as defined in the following paragraphs
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	Control parameters must be in the primary address space.

If your installation uses the RACF component of SecureWay for z/OS to control access to RRS information, ATRSRV requires ALTER access to one of two RACF resources in the FACILITY class.

To view URs on other systems in a sysplex, ATRSRV requires ALTER access to the MVSADMIN.RRS.COMMANDS.*gname.sysname* resource in the FACILITY class, where *gname* is the target logging group name, and *sysname* is the target system name. You may create a RACF profile to permit access to multiple logging groups and systems by including RACF valid generic characters (**, *, and %) in *gname* and *sysname*. See the *z/OS Security Server RACF Security Administrator's Guide* and *z/OS Security Server RACF Command Language Reference* for more information about using these RACF generic characters and defining RACF profiles. By permitting ALTER access through this profile, you can allow users to alter RRS information on any number of systems in the sysplex.

If you are running RRS on a single system, ATRSRV requires ALTER access to either the MVSADMIN.RRS.COMMANDS.*gname.sysname* resource or the MVSADMIN.RRS.COMMANDS resource in the FACILITY class. The MVSADMIN.RRS.COMMANDS resource only allows access to RRS system management functions on the current system. You cannot use the MVSADMIN.RRS.COMMANDS resource to allow or disallow use of RRS on another system.

ATRSRV does not check the SYSSTATE global variable or the SPLEVEL global variable.

Programming requirements

ATRSRV callers must include the following mapping macros: ATRFZSRV, CVT, IHAECVT

Restrictions

None.

Input register information

Before issuing the ATRSRV macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output register information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Contains the reason code
1-13	Unchanged
14	Used as a work register by the system.
15	Contains the return code

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The ATRSRV macro is written as follows:

name	*name*: symbol. Begin *name* in column 1.
␣	One or more blanks must precede ATRSRV.
ATRSRV	
␣	One or more blanks must follow ATRSRV.

```
REQUEST=REMOVINT
  ,RMNAME=(xrmname | 0)
  ,URID=(xurid | 0)
```

```
REQUEST=COMMIT
  ,URID=(xurid | 0)
REQUEST=BACKOUT
  ,URID=(xurid | 0)
REQUEST=REMOVRM
  ,RMNAME=xrmname
REQUEST=UNREGRM
  ,RMNAME=xrmname
REQUEST=FORGET
  ,URID=xurid

  ,RMNAME=xrmname              xrmname: RS-type address or address in register (2) - (12).

  ,URID=xurid                  xurid: RS-type address or address in register (2) - (12).
                               Note: URID is optional with REQUEST=REMOVINT.

  ,GNAME=(xgname |             xgname: RS-type address or register (2) - (12).
current_gname)
  ,SYSNAME=(xsysname |         xsysname: RS-type address or register (2) - (12).
current_sysname)
  ,RCTABLE=(xrctable | 0)      xrctable: RS-type address or address in register (2) - (12).
   ,RCNUM=xrcnum               xrcnum: RS-type address or address in register (2) - (12).
  ,RETCODE=xretcode            xretcode: RS-type address or register (2) - (12).

  ,RSNCODE=xrsncode            xrsncode: RS-type address or register (2) - (12).

  ,PLISTVER=IMPLIED_VERSION
  ,PLISTVER=MAX                Default: PLISTVER=IMPLIED_VERSION
  ,PLISTVER=1
  ,PLISTVER=2
  ,PLISTVER=3
  ,PLISTVER=4
  ,PLISTVER=5

  ,MF=S                        Default: MF=S
  ,MF=(L,list addr)            list addr: RS-type address or register (1) - (12).
  ,MF=(L,list addr,attr)
  ,MF=(L,list addr,0D)
  ,MF=(E,list addr)
  ,MF=(E,list addr,COMPLETE)
```

Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the ATRSRV macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

REQUEST=REMOVINT
REQUEST=COMMIT
REQUEST=BACKOUT
REQUEST=REMOVRM

REQUEST=UNREGRM
REQUEST=FORGET
A required parameter that specifies the requested function:

REQUEST=REMOVINT
Remove a resource manager's interest in a UR.

Note: The resource manager must not be currently active with RRS. That is, the resource manager cannot have exits set with RRS.

You can provide:

- Only the resource manager name

  In this case, the resource manager's interests will be removed from all URs, except for **in-doubt** URs, that the resource manager has expressed interest in.
- Only the UR identifier

  In this case, all resource manager interests will be removed from the specified UR, except for the top-level **in-doubt** UR.
- Both the resource manager name and the UR identifier

  In this case, the resource manager's interests will be removed from the specified UR.

To affect units of recovery on other systems in a sysplex, you may also provide a logging group name and a system name with the GNAME and SYSNAME keywords.

REQUEST=COMMIT
Resolve an **In-doubt** UR to **In-commit**.

To affect units of recovery on other systems in a sysplex, you may also provide a logging group name and a system name with the GNAME and SYSNAME keywords.

REQUEST=BACKOUT
Resolve an **In-doubt** UR to **In-backout**.

To affect units of recovery on other systems in a sysplex, you may also provide a logging group name and a system name with the GNAME and SYSNAME keywords.

REQUEST=REMOVRM
Remove a resource manager's identity.

Note: The resource manager must not be currently active with RRS or have an interest in any URs. If this request is issued after an RRS failure, it is possible that the request will fail due to outstanding interests. RRS does not force deletion of log records for completed transactions, so the failure of RRS may have left completed log records that RRS would normally return to the restarting RM. The restarting RM would respond COMPLETE to those interests. Should this occur, a REMOVINT request by RM name would be required to remove the completed interests before the REMOVRM request can succeed.

You must provide:

- The resource manager name

To affect units of recovery on other systems in a sysplex, you may also provide a logging group name with the GNAME keyword. A target system

name using the SYSNAME keyword, is not required. However, if the SYSNAME keyword is used, it is not validated and is ignored.

REQUEST=UNREGRM
Unregister RM - is used to clean up the resource managers involvement with RRS. Only resource managers that have been unregistered with Registration Services but still set with RRS can be processed. RRS Resource Manager Unregister processing will be invoked to clean up the resource managers involvement with RRS.

You must provide:

- The resource manager name. In this case, provide the resource manager to be unregistered with RRS.

Optionally you may provide:

- The GNAME and SYSNAME keywords. To affect resource managers on other systems in a sysplex, you may also provide a logging group name and a system name with the GNAME and SYSNAME keywords.

REQUEST=FORGET
Perform Forget processing for a UR (Unit of Recovery) whose In-Doubt condition was previously resolved to In-Commit or In-Backout.

You must provide:

- The URID (Unit of Recovery Identifier). In this case, the SDSRM (Server Distributed Syncpoint Manager) interest of the UR will be forgotten. The URID specified must contain the SDSRM interest (top-level UR). Specifying a URID of a child or subordinate UR is not valid for this request and will result in a reason code of ATRSRV_URID_NOT_FOUND (8).

Optionally you may provide:

- The GNAME and SYSNAME keywords. To affect units of recovery on other systems in a sysplex, you may also provide a logging group name and a system name with the GNAME and SYSNAME keywords.

,RMNAME=*rmname*

An input parameter that contains the resource manager name. It is optional with:

- REQUEST=REMOVINT.

It is required with:

- REQUEST=REMOVRM
- REQUEST=UNREGRM.

If REQUEST=REMOVINT is requested and the resource manager you name is a distributed syncpoint resource manager (DSRM), you can not remove an interest from any UR with a state of **in-doubt** unless you specify URID to remove all interests from a specific UR. If REQUEST=REMOVRM is requested, the resource manager will be deleted from RRS storage and the RRS logs. If REQUEST=UNREGRM is requested, the resource manager will be unregistered with RRS.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

,URID=*urid*
An input parameter that contains the UR identifier. It is required with REQUEST=COMMIT, REQUEST=BACKOUT, or REQUEST=FORGET. It is optional with REQUEST=REMOVINT.

To code: Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

,GNAME=*gname*
An optional input parameter that contains the name of the RRS logging group containing the specified resource manager or URID.

- If you specify GNAME, you must also specify SYSNAME. However, if REQUEST=REMOVRM, the SYSNAME parameter is not required.
- If you do not specify GNAME, it will default to the current system's RRS logging group name.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

,SYSNAME=*sysname*
An optional input parameter that contains the name of the system containing the specified resource manager or URID.

- If you specify SYSNAME, you must also specify GNAME.
- If you do not specify SYSNAME, it will default to the current system's system name.
- If you specify REQUEST=REMOVRM, this parameter is not required. If specified, the value is not validated and is ignored.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

,RCTABLE=*rctable*
An optional 4–byte input parameter that contains the address of a "36" area to contain the return code and reason code from the system and logging group processing the request. The area consists of up to 1 36–byte entry. DSECT ATRFZRCA in mapping macro ATRFZQRY maps the storage area. The number of entries will be returned in the RCNUM parameter. The table will only contain valid information if the reason code returned to you is ATRSRV_REMOTE_WARNING or ATRSRV_REMOTE_ERROR. **Default:** RCTABLE=0 No return code table is desired.

If you specify REQUEST=REMOVRM, this parameter contains the address of a "2304" area to contain the return/reason code from each system in the Logging Group processing the REMOVRM request. The area consists of up to 64 36-byte entries.

To code: Specify the RS-type address, or address in register (2)-(12), of a "36" area, or a "2304" if you specify REQUEST=REMOVRM.

,RCNUM=*rcnum*
When using RCTABLE, a required output parameter that will contain the number of entries in the RCTABLE area (either 0 if no error occurred or 1 for one error).

To code: Specify the RS-type address of a fullword field, or register (2)-(12).

,RETCODE=*retcode*
An optional output parameter into which the system copies the return code from GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12).

,RSNCODE=*rsncode*

An optional output parameter into which the system copies the reason code from GPR 0.

To code: Specify the RS-type address of a fullword field, or register (2)-(12).

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=1
,PLISTVER=2
,PLISTVER=3
,PLISTVER=4
,PLISTVER=5

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **1**, which supports all parameters except those specifically referenced in higher versions.
- **2**, which supports all parameters in version 1 along with the following parameters:
  - GNAME
  - RCNUM
  - RCTABLE
  - SYSNAME
- **3**, which supports additional function along with the following REQUEST parameter:
  - REQUEST=REMOVRM
- **4**, which supports additional function:
  - REQUEST=UNREGRM
- **5**, which supports additional function:
  - REQUEST=FORGET

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 1
- A decimal value of 2
- A decimal value of 3
- A decimal value of 4

- A decimal value of 5

,MF=S
,MF=(L,*list addr***)**
,MF=(L,*list addr***,***attr***)**
,MF=(L,*list addr***,0D)**
,MF=(E,*list addr***)**
,MF=(E,*list addr***,COMPLETE)**
An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,*list addr*
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*
An optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *attr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

,COMPLETE
Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

ABEND codes

None.

Return and reason codes

When the ATRSRV macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code:

  0 ATRSRV_SUCCESS — request completed successfully

  4 ATRSRV_WARNING —request completed with a warning condition

  8 ATRSRV_FAILURE — request failed
- When the return code is not 0, GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code. The return and reason codes are shown in the following table. The ATRFZSRV mapping macro provides the equate symbols for the return and reason codes.

Table 57. ATRSRV — Resolve Units of Recovery Return and Reason Codes

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
4	17	**Equate symbol:** ATRSRV_REMOTE_WARNING **Meaning:** An error occurred while processing a remote request that did not affect the processing of the request. If you have been provided with an RCTABLE, it will contain detailed error information. **Action:** Fix the error noted in the RCTABLE to eliminate the warning message for future requests.
4	20	**Equate symbol:** ATRSRV_DeleteRMObjectNotSupported **Meaning:** The specified Resource Manager was deleted from the Resource Manager Data log but one or more systems in the RRS logging group do not support the Delete RM (REMOVRM) function. If the Resource Manager is on those systems, that Resource Manager will persist. **Action:** None. The Resource Manager cannot be deleted from systems that do not support the Delete RM (REMOVRM) process.
8	1	**Equate symbol:** ATRSRV_UR_NOT_IN_DOUBT **Meaning:** The UR state is not **In-Doubt**. **Action:** Specify the correct UR and retry the request.
8	2	**Equate symbol:** ATRSRV_RM_IS_ACTIVE **Meaning:** You specified REMOVINT or REMOVRM but named a resource manager that is currently active with RRS. **Action:** Verify that you specified the correct resource manager. You might need to stop the resource manager, or make it inactive with RRS/MVS. When the resource manager is inactive, retry the request.
8	3	**Equate symbol:** ATRSRV_RRS_NOT_ACTIVE **Meaning:** The ATRSRV service is not available. RRS/MVS is not active on this system. **Action:** Retry the request when RRS/MVS is active.
8	4	**Equate symbol:** ATRSRV_UR_HAS_DSRM **Meaning:** You specified REMOVINT and named a distributed syncpoint resource manager (DSRM), but the state of the UR you specified is **In-Doubt**. **Action:** Before you can remove the interest for the DSRM, the UR state must be resolved or you must remove all resource manager interests for the **In-Doubt** UR.

Table 57. ATRSRV — Resolve Units of Recovery Return and Reason Codes (continued)

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	5	**Equate symbol:** ATRSRV_BAD_REMOVINT_PARM **Meaning:** You specified REMOVINT but did not specify either RMNAME or URID. **Action:** Specify one or both of the missing keywords and retry the request.
8	6	**Equate symbol:** ATRSRV_URID_NOT_VALID **Meaning:** You specified a UR identifier that is either not valid or associated with a UR that no longer exists. **Action:** Verify that you specified the correct UR identifier. If you did not specify the correct UR identifier, do so and retry the request. If you did specify the correct UR identifier, the UR has completed. Do not retry the request.
8	7	**Equate symbol:** ATRSRV_RID_NOT_SUPPORTED **Meaning:** You specified the UR identifier for an **in-doubt** UR that the system cannot resolve to **in-commit** or **In-backout**. **Action:** You must wait for the DSRM to resolve the In-Doubt condition.
8	8	**Equate symbol:** ATRSRV_URID_NOT_FOUND **Meaning:** You specified a UR identifier that the system is unable to find. The UR might still exist but the system cannot find it. This reason code might also be issued when the request requires a top level URID and a child or subordinate URID is specified. **Action:** Verify that you specified the correct UR identifier. If you did not specify the correct UR identifier, do so and retry the request. If you did specify the correct UR identifier, retry the request later.
8	9	**Equate symbol:** ATRSRV_NO_UR_FOR_RM **Meaning:** You specified URINFO and named a resource manager that does not have any interests in any URs. **Action:** Verify that you provided the correct resource manager. If you did not specify the correct resource manager, do so and retry the request. If you expected URs to be returned, you need to determine why the named resource manager has no interests in any URs. Otherwise, do nothing.

Table 57. ATRSRV — Resolve Units of Recovery Return and Reason Codes (continued)

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	A	**Equate symbol:** ATRSRV_NOT_AUTH **Meaning:** You are not in supervisor state, with system key, and the RACF component of SecureWay for z/OS is not active. **Action:** Either request activation of the RACF component and obtain ALTER access to MVSADMIN.RRS.COMMANDS or recode your application to run in supervisor state with system key.
8	B	**Equate symbol:** ATRSRV_NOT_SAF_AUTH **Meaning:** You do not have ALTER access to MVSADMIN.RRS.COMMANDS. **Action:** Either obtain ALTER access to MVSADMIN.RRS.COMMANDS or recode your application to run in supervisor state with system key.
8	C	**Equate symbol:** ATRSRV_RRS_DOWNLEVEL **Meaning:** The unit of recovery that was specified to process contains information that is not understood by this level of RRS. This level of RRS is downlevel. This reason code may also be issued when an ATRSRV request was processed on a downlevel version of RRS that does not understand the request. **Action:** The attempt to process this unit of recovery must be made from a version of RRS that can process this unit of recovery. Route the ATRSRV request to a RRS system that does support the request.
8	F	**Equate symbol:** ATRSRV_GNAME_INVALID **Meaning:** The value specified for the GNAME parameter is not valid. **Action:** Correct the GNAME value and retry the request.
8	10	**Equate symbol:** ATRSRV_SYSNAME_INVALID **Meaning:** The value specified for the SYSNAME parameter is not valid. **Action:** Correct the SYSNAME value and retry the request.

Table 57. ATRSRV — Resolve Units of Recovery Return and Reason Codes (continued)

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	16	**Equate symbol:** ATRSRV_INSTANCE_FAILURE **Meaning:** An error occurred while processing a remote request. Your RCTABLE, if one was provided, will contain the failing service identifier, return code, and reason code. Service identifiers: **1** IXCMSGO macro interface **2** IXCMSGI macro interface Any other value is an internal error and should be reported to the IBM Support center. **Action:** Fix the errors noted in the RCTABLE and retry the request.
8	18	**Equate symbol:** ATRSRV_REMOTE_ERROR **Meaning:** An error occurred while processing a remote request. The request was not performed. If you received an RCTABLE, it will contain detailed error information. **Action:** Fix the error noted in the RCTABLE and retry the request.
8	19	**Equate symbol:** ATRSRV_RESP_NOT_RECEIVED **Meaning:** No response was received from the remote system. No information was returned. **Action:** Retry the request.
8	1A	**Equate symbol:** ATRSRV_REMOTE_NOT_ACTIVE **Meaning:** The remote system is not active or RRS is not active on the remote system. No information was returned. **Action:** Retry the request once the remote system is available.
8	1B	**Equate symbol:** ATRSRV_UR_HAS_NO_INT **Meaning:** The target UR for a remove interest request has no interests. The request is rejected. **Action:** Specify the correct UR and retry the request.
8	1C	**Equate symbol:** ATRSRV_UR_NOT_TOP **Meaning:** The requested action can only be performed on a top-level UR, but you specified a cascaded UR. The request is rejected. **Action:** Specify the top-level UR associated with the cascaded UR and retry the request.

Table 57. ATRSRV — Resolve Units of Recovery Return and Reason Codes (continued)

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	1D	**Equate symbol:** ATRSRV_RIN_NOT_SUPPORTED **Meaning:** RRS is unable to process your remove interest request because the coordinator system cannot be found. The request is rejected. **Action:** If the URID is in the restart log, retry the remove interest request.
8	1E	**Equate symbol:** ATRSRV_RMSTILLHASINTERESTS **Meaning:** The specified Resource Manager cannot be deleted since it has outstanding interest in one or more URs. **Action:** If the Resource Manager must be deleted, the interests in all URs must be removed prior to deleting the Resource Manager.
8	1F	**Equate symbol:** ATRSRV_RMISNOTKNOWNTORRS **Meaning:** The specified Resource Manager could not be found in the Resource Manager Data log or on any system in the RRS logging group. Either the Resource Manager has already been deleted or the name was entered incorrectly. **Action:** Make sure the Resource Manager's name is spelled correctly. Otherwise the Resource Manager is deleted.
8	20	**Equate symbol:** ATRSRV_DELETERMOBJECTSNOTSUPPORTED **Meaning:** The specified Resource Manager was deleted from the Resource Manager Data log but one or more systems in the RRS logging group do not support the Delete RM (REMOVRM) function. If the Resource Manager is on those systems, that Resource Manager will persist. **Action:** None, the Resource Manager cannot be deleted from systems that do not support Delete RM (REMOVRM) process.
8	21	**Equate symbol:** ATRSRV_ERRORDELETINGRMLOGENTRY **Meaning:** The specified Resource Manager was not deleted due to errors deleting the Resource Manager from the Resource Manager logs. **Action:** Correct the problem and retry the Delete RM (REMOVRM) request.

Table 57. ATRSRV — Resolve Units of Recovery Return and Reason Codes (continued)

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	22	**Equate symbol:** ATRSRV_DELETERMNOTSUPPORTED **Meaning:** The specified Resource Manager is on a system that does not support the Delete RM (REMOVRM) function. **Action:** None, the Resource Manager cannot be deleted.
8	23	**Equate symbol:** ATRSRV_RM_NOT_FOUND **Meaning:** The requested Resource Manager could not be found on the specified system in the RRS logging group. Either the Resource Manager is not currently defined on the specified system or it was entered incorrectly. **Action:** Make sure the Resource Manager's name is spelled correctly. Otherwise, determine where the Resource Manager is currently defined and perform the RM Unregister request on that system.
8	24	**Equate symbol:** ATRSRV_RM_STILL_REGISTERED **Meaning:** The requested Resource Manager is still registered with Registration Services. To unregister a Resource Manager with RRS, it must be unregistered with Registration Services. **Action:** Issue the request again after the resource manager has become unregistered with Registration Services.
8	25	**Equate symbol:** ATRSRV_RM_UNREGISTERED_NOT_ALLOWED **Meaning:** A Resource Manager in the Reset or Unset state is already considered unregistered with RRS so it cannot be unregistered again. **Action:** None, the Resource Manager is already considered unregistered.
8	26	**Equate symbol:** ATRSRV_UR_Not_In_Forget **Meaning:** The UR state is not In-Forget. **Action:** Specify the correct UR and retry the request later.
8	27	**Equate symbol:** ATRSRV_Not_Server_DSRM **Meaning:** The resource manager does not have the server distributed syncpoint resource manager role for the unit of recovery. The system rejects this service request. **Action:** Check the calling program for a probable coding error.

Table 57. ATRSRV — Resolve Units of Recovery Return and Reason Codes (continued)

Return code in: hexadecimal	Reason code in: hexadecimal	Meaning and action
8	38	**Equate symbol:** ATRSRV_UR_Not_In_Forget **Meaning:** The ur state is not In-Forget. **Action:** Specify the correct UR and retry the request.
8	39	**Equate symbol:** ATRSRV_Not_Server_DSRM **Meaning:** The resource manager does not have the server distributed syncpoint resource manager role for the unit of recovery. The system rejects this service request. **Action:** Check the calling program for a probable coding error.
8	FFF	**Equate symbol:** ATRSRV_UNEXPECTED_ERROR **Meaning:** An unexpected system error occurred. **Action:** Contact your system support.

Examples

Example 1

Issue the following request to change the state of a single UR from **In-doubt** to **In-commit**:

```
         ATRSRV  REQUEST=COMMIT,URID=MYURID
*
* INPUT VALUES
*
MYURID   DC     XL16'123456789ABCDEF123456789ABCDEF'  UR IDENTIFIER
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZSRV
```

Example 2

The resource manager named UTRICK is not currently active. Issue the following request to remove its interests from all of the URs that it had expressed interest in:

```
     ATRSRV REQUEST=REMOVINT,RMNAME=RMTWO
*
* INPUT VALUES
*
RMTWO    DC     CL32'UTRICK'              RESOURCE MANAGER NAME
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZSRV
```

Example 3

Issue the following request to remove all resource manager interests from a specific UR:

```
         ATRSRV  REQUEST=REMOVINT,URID=MYURID
*
* INPUT VALUES
*
MYURID   DC     XL16'123456789ABCDEF123456789ABCDEF'  UR IDENTIFIER
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZSRV
```

Example 4

Issue the following request to remove all interests for resource manager UTRICK from a specific UR:

```
         ATRSRV  REQUEST=REMOVINT,URID=MYURID,RMNAME=MYRM
*
* INPUT VALUES
*
MYRM     DC     CL32'UTRICK'                          RESOURCE MANAGER NAME
MYURID   DC     XL16'123456789ABCDEF123456789ABCDEF'  UR IDENTIFIER
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZSRV
```

Example 5

Issue the following request to change the state of a single UR from in-doubt to in-backout on system SY1 in logging group PLEX1:

```
         ATRSRV  REQUEST=BACKOUT,URID=MYURID,SYSNAME=MYSYS,           X
               GNAME=MYGROUP
*
* INPUT VALUES
*
MYSYS    DC     CL8'SY1'                              SYSTEM NAME
MYGROUP  DC     CL8'PLEX1'                            LOGGING GROUP NAME
MYURID   DC     XL16'123456789ABCDEF123456789ABCDEF'  UR IDENTIFIER
*
* REQUIRED MAPPINGS
*
         CVT DSECT=YES
         IHAECVT
         ATRFZSRV
```

Chapter 13. ATRQSRV utility - query and update RRS information

Use the ATRQSRV Utility to Query and Update RRS Information from JCL jobs.

The control statements for this utility program follow standard conventions for JCL statements. The utility accepts 80-byte records with one or more parameters per record. See *z/OS MVS JCL Reference* for additional information about JCL statements.

Utility control statements are contained in columns 1 through 80. Ensure columns 73 through 80 do not include line numbers. A statement that exceeds 80 characters must be continued on one or more additional records. When it is necessary to continue to the next line, use a plus sign as the last character of the line you wish to continue.

Using the ATRQSRV utility

The name of the utility is ATRQSRV. The program resides in SYS1.LINKLIB so no special program authorization is required.

Authorizing use of the utility

For more information see, "Setting up access authorization" on page 565.

Report levels

- Each command report will begin with the command input line(s) as entered by the user.
- Query commands support either a SUMMARY level or a DETAILED level.
  - Requesting the SUMMARY level returns a tabular report summarizing the returned information for the input parameters and is limited to 121 characters in width. For some commands, not all available information is returned.
  - Requesting the DETAILED level returns a non-tabular report displaying all the returned information for the input parameters and is limited to 121 characters in width.

Coding the ATRQSRV utility

For examples of how to set up your JCL and code the control statements for ATRQSRV, see the following figures:

- List all resource managers, see Example 1 in "Examples of using the ATRQSRV utility" on page 645
- List all active units of recovery (URs), see Example 2 in "Examples of using the ATRQSRV utility" on page 645.

The utility control statements are described below:

PGM=ATRQSRV
: Describes the program name (PGM=ATRQSRV) or, if the job control statements reside in a procedure library, the procedure name.

ATRQSRV does not support any PARM= values.

SYSPRINT
Defines a sequential data set for the output. The output will contain the command input line(s) as entered by the user followed by the report. It may also contain appropriate ATRxxxI messages indicating report success or failure. The output can be written to a system output device, a tape volume, or a DASD volume.

The SYSPRINT DD statement is required for each use of ATRQSRV. The block size for the SYSPRINT data set must be at least 121. Any blocking factor can be specified for this record size, but the maximum value for the block size is 32670 bytes.

SYSIN
Defines the control data set. The control data set normally resides in the input stream; however, it can be defined as a member in a partitioned data set or PDSE.

The SYSIN DD statement is required for each use of ATRQSRV. The block size for the SYSIN data set must be a multiple of 80. Any blocking factor can be specified for this block size.

SYSABEND
Defines a sequential data set to contain any SYSABEND dumps. Not required for normal processing.

STATEMENTS

Table 58. Statements

Statement	Use
LOGINFO	Browse the Archive, Main/Delayed UR, Restart and RM Data log streams
URINFO	Query unit of recovery (UR) information
RMINFO	Query resource manager (RM) information
WMINFO	Query work manager (WM) information
SYSINFO	Query RRS sysplex and logging group information
REMOVINT	Remove interest(s) from URs
COMMIT	Force an InDoubt transaction to the COMMIT state
BACKOUT	Force an InDoubt transaction to the BACKOUT state
DELETERM	Delete a resource manager from RRS
UNREGRM	Unregister RM to clean up the resource manager's involvement with RRS

Using Wildcards in the ATRQSRV Utility
RRS allows you to use wildcard characters in the ATRQSRV statements. See “Using wildcards in RRS panels” on page 569 for more information on using wildcards in the ATRQSRV Utility.

ATRQSRV return codes

Table 59. ATRQSRV Return Codes

Return Code In Decimal	Meaning and Action
0	**Meaning:** Service completed successfully. **Action:** No action required.
4	**Meaning:** Errors detected. Error messages will appear in SYSPRINT DD, **Action:** In SYSPRINT, look for ATR messages and refer to their explanation and user response.
8	**Meaning:** Errors writing to SYSPRINT DD. The SYSPRINT DD could not be used. **Action:** In the JCL, make sure the SYSPRINT DD specifies an output type of data set or file.
12	**Meaning:** Errors opening SYSPRINT DD. The SYSPRINT DD was not specified or could not be used. Error messages will appear in the JCL job log. **Action:** In the JCL job log, look for IEC messages and refer to their explanation and user response.
16	**Meaning:** Errors closing SYSPRINT DD. The SYSPRINT DD did not close properly. **Action:** In the JCL job log, look for IEC messages and refer to their explanation and user response.
4095	**Meaning:** This service encountered an unexpected error. **Action:** Contact IBM support.

Examples of using the ATRQSRV utility

Example 1

List all resource managers:

```
//LISTRM      JOB
//STEP1       EXEC PGM=ATRQSRV
//SYSPRINT    DD  SYSOUT=*
//SYSIN       DD  *
  RMINFO
```

Example 2

List all active units of recovery (URs)

```
//LISTUR      JOB
//STEP1       EXEC PGM=ATRQSRV
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD *
  URINFO
```

ATRQSRV statement details and parameters

URINFO

URINFO	[URID(urid-pattern)] [SURID(surid-pattern)] [URTYPE(ALL | PROT | UNPROT)] [URSTATE(ALL | urstate-list)] [RMNAME(rmname-pattern)] [WMNAME(wmname-pattern)] [GNAME(gname-pattern)] [SYSNAME(sysname-pattern)] [LUWID(luwid-pattern)] [EIDTID(tid)] [EIDTIDLOW(tid)] [EIDTIDHIGH(tid)] [EIDGTID(gtid-pattern)] [XIDFORMATID(formatid)] [XIDGTRID(gtrid-pattern)] [XIDBQUAL(bqual-pattern)] [AFTER(yyyy/mm/dd[,hh:mm:ss])] [BEFORE(yyyy/mm/dd[,hh:mm:ss])] [TIMEFORMAT(LOCAL | GMT)] [DURATION(hh:mm:ss.sssss) [EXCLUDE(NONE | exclude-list)] [SORT(NONE | sort-list)] [LEVEL(SUMMARY | DETAILED)]

URID(urid-pattern)
specifies a specific UR identifier or a pattern string containing * and/or ? to filter the returned information. If omitted, the returned information is not filtered by UR identifier.

SURID(surid-pattern)
specifies a specific Sysplex UR identifier or a pattern string containing * and/or ? to filter the returned information. If omitted, the returned information is not filtered by Sysplex UR identifier.

URTYPE(ALL|PROT|UNPROT
specifies the type of URs to be returned. PROT requests only URs with protected interests. UNPROT requests only URs with unprotected interests. If omitted or ALL is specified, the returned information is not filtered by UR type.

URSTATE(ALL|urstate-list)
specifies the state of the URs to be returned. URSTATE-LIST is one or more of the following, separated by commas:

FLT
In-Flight UR state

SCK
In-State_Check UR state

OLA
In-Only-Agent UR state

PRP
In-Prepare UR state

DBT
In-Doubt UR state

CMT
In-Commit UR state

BAK
In-Backout UR state

EUR
In-End UR state

CMP
In-Completion UR state

FGT
In-Forget UR state

If omitted or ALL is specified, the returned information is not filtered by UR state.

RMNAME(rmname-pattern)
specifies a specific resource manager name or a pattern string containing * and/or ? to filter the returned information. If omitted, the returned information is not filtered by resource manager name.

WMNAME(wmname-pattern)
specifies a specific work manager name or a pattern string containing * and/or ? to filter the returned information. If omitted, the returned information is not filtered by work manager name.

GNAME(gname-pattern)
specifies a specific logging group to be queried or a pattern string containing * and/or ? to determine the logging group(s) to be queried. If omitted, this parameter defaults to the logging group name of the current system.

SYSNAME(sysname-pattern)
specifies a specific system name to be queried or a pattern string containing * and/or ? to determine the system(s) to be queried. If omitted, this parameter defaults to the current system's name.

LUWID(luwid-pattern)
specifies a LUWID or a pattern string containing * and/or ? to filter the returned information. luwid-pattern has the following format:

netid-pattern.luname-pattern,instnum-pattern,seqnum-pattern

All patterns must be specified. If omitted, the returned information is not filtered by LUWID.

EIDTID(tid)
specifies a specific TID value to filter the returned information. If omitted, the returned information is not filtered by TID. If specified, EIDTIDLOW and EIDTIDHIGH can not be specified.

EIDTIDLOW(tid)
specifies a specific TID value to filter the returned information such that only URs with TIDs greater than or equal to this value are be returned. If omitted, the returned information is not filtered by TID. If specified, EIDTID can not be specified.

EIDTIDHIGH(tid)
specifies a specific TID value to filter the returned information such that only

URs with TIDs less than or equal to this value are be returned. If omitted, the returned information is not filtered by TID. If specified, EIDTID can not be specified.

EIDGTID(gtid-pattern)
specifies a GTID or a pattern string containing * and/or ? to filter the returned information. If omitted, the returned information is not filtered by GTID.

XIDFORMATID(formatid)
specifies a specific FORMATID to filter the returned information. If omitted, the returned information is not filtered by FORMATID.

XIDGTRID(gtrid-pattern)
specifies a specific GTRID or a pattern string containing * and/or ? to filter the returned information. If omitted, the returned information is not filtered by GTRID.

XIDBQUAL(bqual-pattern)
specifies a specific BQUAL or a pattern string containing * and/or ? to filter the returned information. If omitted, the returned information is not filtered by BQUAL.

AFTER(yyyy/mm/dd[,hh:mm:ss])
specifies the date and time for the oldest UR to be returned. If AFTER is omitted, URs are returned from the oldest UR known. If a time is omitted, records are returned from midnight of the specified date.

BEFORE(yyyy/mm/dd[,hh:mm:ss])
specifies the date and time for the youngest UR to be returned. If BEFORE is omitted, URs are returned up to the youngest UR known. If a time is omitted, records are returned up to midnight of the specified date.

TIMEFORMAT(LOCAL|GMT)
specifies the time format for the AFTER and BEFORE parameters.

DURATION(hh:mm:ss.sssss)
specifies that only URs be returned if they have been in their current state longer than the specified time. If omitted, the returned information is not filtered by current state duration.

EXCLUDE(NONE|exclude-list)
specifies that certain information not be returned. exclude-list is one or more of the following, separated by commas:

NONDEFERRED
Exclude URs that are deferred

FAILEDINTS
Exclude URs with failed interests

RESTARTLOG
Exclude URs from the Restart log stream

CASCADEDURS
Exclude cascaded URs

LOCALTRANS
Exclude Local transaction mode URs

GLOBALTRANS
Exclude Global transaction mode URs

RRSMANAGED
Exclude URs currently managed by RRS

NONRRSMANAGED
Exclude URs currently managed by a non-RRS work manager

If omitted or NONE is specified, no exclusions are performed on the returned information.

SORT(NONE|sort-list)
specifies that the returned information is to be sorted, using the provided criteria. sort-list is one or more of the following, separated by commas:

- key,direction
- where
  - key is one of the following:

    WMNAME URID CREATETIME URSTATE LUWID EID XID GNAME SYSNAME SURID
  - direction is one of the following

    A (ascending) D (descending)

A key can only be specified once.

Each sort key/direction pair is applied from left to right. For example, SORT(SYSNAME,A,URID,A) will cause the UR information to be returned sorted by system name and within each system name, by UR identifier.

If omitted or NONE is specified, no sorting of the returned information is performed.

LEVEL(SUMMARY|DETAILED)
specifies a summary (tabular) output or a detailed output

SUMMARY output

```
URINFO  2006/09/07 17:08:39  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
URINFO  LEVEL(SUMMARY)
        DEFAULTS: GNAME(PLEX1) SYSNAME(SY1) URTYPE(ALL) URSTATE(ALL)
                  TIMEFORMAT(LOCAL) EXCLUDE(NONE) SORT(NONE)
URID              SYSNAME GNAME STATE    TYPE COMMENTS WMNAME
BBA25A0F7E2701010 SY1     PLEX1 InCommit Prot *        SYS1.MAINASID.0024
```

DETAILED output

```
URINFO       2006/09/07 17:08:39  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
URINFO  LEVEL(DETAILED)
        DEFAULTS: GNAME(PLEX1) SYSNAME(SY1) URTYPE(ALL) URSTATE(ALL)
                  TIMEFORMAT(LOCAL)EXCLUDE(NONE) SORT(NONE)
URID = BBA25A0F7E2701010
  Created  = 2004/08/07 19:25:20.924229  LOCAL
  Comments = *
  State    = InCommit
  Type     = Prot
  Sysname  = SY1
  Gname    = PLEX1
  Surid    = N/A
  WMName   = SY1.MAINASID.0024

  LUWID
    NetID.LuName = ABC.DEF
    TPInstance   = 000000000001
    SeqNum       = 0002

  EID
    TID  = 002864434397  (decimal)
    GTID = 00-07 11223344 55667788                  |.......h       |

  XID
    FormatID = 003654931682 (decimal) D9D9D4E2 (hexadecimal)
```

```
  GTRID = 00-0F E2C9D4C4 C5E5F0F0 F1C9C2D4 D7D2D9D9 |SIMDEV001IBMPKRR|
          10-1F E2F1F640 40404040 4040BBA2 5A0F7249 |S16       .s!...|
          20-21 D080                               |}.             |
  BQUAL = 00-0F D9D9D4E2 4BBBA25A 0F7249F3 80E2C9D4 |RRMS..s!...3.SIM|
          10-1F C4C5E5F0 F0F1C9C2 D4D7D2D9 D9E2F1F6 |DEV001IBMPKRRS16|
          20-26 40404040 404040                    |               |

Expressions of Interest: Number of Entries: 1
  URIToken = 7E17000000000000000055000155555555
    RMName       = UTRICK
    Type         = Prot
    Status       = DEFERRED
    Role         = Participant
    State        = InCommit
    BACKOUT      = Uncalled
    COMPLETION   = Uncalled
    COMMIT       = 00000030 >00:02:17.0157
    DSE/IN_DOUBT = Uncalled
    End_UR       = Uncalled
    EXIT_FAILED  = 00000000
    ONLY_AGENT   = Uncalled
    PRE_PREPARE  = 00000000  00:00:05.3571
    PREPARE      = 00000000  00:05:19.1777
    STATE_CHECK  = Uncalled
    PDATA = 000-00F D9D9D4E2 4BBBA25A 0F7249F3 80E2C9D4 |RRMS..s!...3.SIM|
            010-01F C4C5E5F0 F0F1C9C2 D4D7D2D9 D9E2F1F6 |DEV001IBMPKRRS16|
```

RMINFO

RMINFO	[RMNAME(rmname-pattern)] [GNAME(gname-pattern)] [SYSNAME(sysname-pattern)] [METADATA(NO | YES)] [LEVEL(SUMMARY | DETAILED)]

where

RMNAME(rmname-pattern)
specifies a specific resource manager name or a pattern string containing * and/or ? to filter the returned information. If omitted, the returned information is not filtered by resource manager name.

GNAME(gname-pattern)
specifies a specific logging group to be queried or a pattern string containing * and/or ? to determine the logging group(s) to be queried. If omitted, this parameter defaults to the logging group of the current system.

SYSNAME(sysname-pattern)
specifies a specific system name to be queried or a pattern string containing * and/or ? to determine the system(s) to be queried. If omitted, this parameter defaults to the current system.

METADATA(NO|YES)
specifies that MetaData will or will not be processed. If NO, the default value, is specified MetaData will not be returned. If YES is specified and a resource manager has saved MetaData, it will be returned. To display the MetaData, LEVEL(DETAILED) must also be specified.

LEVEL(SUMMARY|DETAILED)
specifies a summary (tabular) output or a detailed output.

SUMMARY output

```
RMINFO       2006/09/07 17:08:39  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
RMINFO  LEVEL(SUMMARY)
        DEFAULTS: GNAME(PLEX1) SYSNAME(SY1) METADATA(NO)
RMNAME            RMTOKEN                           STATE  SYSNAME  GNAME
UTRICK            01000001021D41780000000300000001 Reset  SY1      PLEX1
```

DETAILED output

```
RMINFO       2006/09/07 17:08:39  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
RMINFO  METADATA(YES) LEVEL(DETAILED)
        DEFAULTS: GNAME(PLEX1) SYSNAME(SY1)
RMName = UTRICK
  RMToken  = 01000001021D41780000000300000001
  State    = Reset
  Sysname  = SY1
  Gname    = PLEX1
  MetaData = Not Present
  URID               SYSNAME GNAME STATE    TYPE COMMENTS WMNAME
    BBA25A0F7E2701010 SY1     PLEX1 InCommit Prot *      SYS1.MAINASID.0024
```

WMINFO

WMINFO	[WMNAME(wmname-pattern)] [GNAME(gname-pattern)] [SYSNAME(sysname-pattern)] LEVEL(SUMMARY | DETAILED)]

where

WMNAME(wmname-pattern)
specifies a specific work manager name or a pattern string containing * and/or ? to filter the returned information. If omitted, the returned information is not filtered by work manager name.

GNAME(gname-pattern)
specifies a specific logging group to be queried or a pattern string containing * and/or ? to determine the logging group(s) to be queried. If omitted, this parameter defaults to the logging group of the current system.

SYSNAME(sysname-pattern)
specifies a specific system name to be queried or a pattern string containing * and/or ? to determine the system(s) to be queried. If omitted, this parameter defaults to the current system.

LEVEL(SUMMARY|DETAILED)
specifies a summary (tabular) output or a detailed output.

SUMMARY output

```
WMINFO       2006/09/07 17:19:41  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
WMINFO  LEVEL(SUMMARY)
        DEFAULTS: GNAME(PLEX1) SYSNAME(SY1)
WMNAME               SYSNAME  GNAME
SYS1.MAINASID.0024   SY1      PLEX1
```

DETAILED output

```
WMINFO       2006/09/07 17:19:41  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
WMINFO  LEVEL(DETAILED)
        DEFAULTS: GNAME(PLEX1) SYSNAME(SY1)
WMName = SYS1.MAINASID.0024
  Sysname = SY1
```

```
Gname   = PLEX1
URID                SYSNAME GNAME STATE    TYPE COMMENTS
  BBA25A0F7E2701010 SY1     PLEX1 InCommit Prot *
```

LOGINFO

LOGINFO	LOG(ARCHIVE|UR|RESTART|RMDATA|METADATA) [GNAME(logging-group-name)] [URID(ur-identifier)] [SURID(surid)] [RMNAME(rmname)] [AFTER(yyyy/mm/dd[,hh:mm:ss])] [BEFORE(yyyy/mm/dd[,hh:mm:ss])] [LEVEL(SUMMARY|DETAILED)

where

LOG(ARCHIVE|UR|RESTART|RMDATA|METADATA)
specifies the log stream(s) to be read.

GNAME(logging-group-name)
specifies the logging group whose log streams are to be read. If omitted, this parameter defaults to the logging group of the current system.

URID(ur-identifier)
specifies the UR identifier to filter the returned information when LOG(ARCHIVE), LOG(UR) or LOG(RESTART) is specified. This parameter may contain a specific UR identifier or a pattern string containing * and/or ? to filter the returned information. If omitted, returned records will not be filtered by UR identifier.

SURID(surid)
specifies the Sysplex UR identifier to filter the returned information when LOG(ARCHIVE), LOG(UR) or LOG(RESTART) is specified. This parameter may contain a specific Sysplex UR identifier or a pattern string containing * and/or ? to filter the returned information. If omitted, returned records will not be filtered by SURID.

RMNAME(rmname)
specifies the resource manager name to filter the returned information when LOG(RMDATA) or LOG(METADATA) is specified. This parameter may contain a specific resource manager name or a pattern string containing * and/or ? to filter the returned information. If omitted, returned records will not be filtered by resource manager name.

AFTER(yyyy/mm/dd[,hh:mm:ss])
specifies the date and time for the oldest record to be returned. If AFTER is omitted, records are returned from the oldest record in the log stream. If a time is omitted, records are returned from midnight of the specified date.

BEFORE(yyyy/mm/dd[,hh:mm:ss])
specifies the date and time for the youngest record to be returned. If BEFORE is omitted, records are returned up to the youngest record in the log stream. If a time is omitted, records are returned up to midnight of the specified date.

LEVEL(SUMMARY|DETAILED)
specifies the amount of information returned. See “Checking the log streams” on page 570 for the data returned for each level and log type.

Output

```
LOGINFO      2006/09/07 17:43:04  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
LOGINFO  LOG(RMDATA)
         DEFAULTS: GNAME(PLEX1)
RRS/MVS LOG STREAM BROWSE SUMMARY REPORT
```

Refer to the following sample report entries:

- Figure 23 on page 574 Detail Unit of Recovery Report Entry
- Figure 24 on page 576 Detail Archive Report Entry
- Figure 25 on page 577 Sample Resource Manager Entry
- Figure 26 on page 577 Sample Resource Manager Meta Data Entry

SYSINFO

SYSINFO	[GNAME(gname-pattern)] [SYSNAME(sysname-pattern)] [LEVEL(SUMMARY | DETAILED)]

where

GNAME(gname-pattern)
specifies a specific logging group name or a pattern string containing * and/or ? to filter the returned information. If omitted, information is returned for all logging groups on systems matching the SYSNAME parameter.

SYSNAME(sysname-pattern)
specifies a specific system name or a pattern string containing * and/or ? to filter the returned information. If omitted, information is returned for all systems in the sysplex

LEVEL(SUMMARY|DETAILED)
specifies a summary (tabular) output or a detailed output.

SUMMARY output

```
SYSINFO      2006/09/07 17:19:41  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
SYSINFO  LEVEL(SUMMARY)
         DEFAULTS: GNAME(*) SYSNAME(*)
SYSNAME  GNAME
SY1      PLEX1
SY2      PLEX
```

DETAILED output

```
SYSINFO      2006/09/07 17:19:41  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
SYSINFO  LEVEL(DETAILED)
         DEFAULTS: GNAME(*) SYSNAME(*)
Sysname = SY1
  Gname = PLEX1
Sysname = SY2
  Gname = TEST
```

REMOVINT

REMOVINT	[RMNAME(rmname)] [URID(ur-identifier)] [GNAME(logging-group-name)] [SYSNAME(system-name)]

where

RMNAME(rmname)
specifies the resource manager name whose interests are to be removed. This parameter can not contain any wildcard characters. If RMNAME is not specified, URID must be specified.

URID(ur-identifier)
specifies the UR identifier whose interests are to be removed. This parameter can not contain any wildcard characters. If URID is not specified, RMNAME must be specified

GNAME(logging-group-name)
specifies the logging group that contains the UR identifier. If omitted, this parameter defaults to the logging group of the current system. If GNAME is specified, SYSNAME must also be specified.

SYSNAME(system-name)
specifies the system name where the UR resides. If omitted, this parameter defaults to the current system. If SYSNAME is specified, GNAME must also be specified.

Output

```
REMOVINT      2006/09/07 17:19:41  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
REMOVINT RMNAME(UTRICK)
         DEFAULTS: GNAME(PLEX1) SYSNAME(SY1)
ATR074I Remove Interest processed successfully.
```

COMMIT

COMMIT	URID(ur-identifier) [GNAME(logging-group-name)] [SYSNAME(system-name)]

where

URID(ur-identifier)
specifies the UR identifier for the InDoubt UR to be committed. This parameter is required and can not contain any wildcard characters.

GNAME(logging-group-name)
specifies the logging group that contains the UR identifier. If omitted, this parameter defaults to the logging group of the current system. If GNAME is specified, SYSNAME must also be specified.

SYSNAME(system-name)
specifies the system name where the UR resides. If omitted, this parameter defaults to the current system. If SYSNAME is specified, GNAME must also be specified.

Output

```
COMMIT      2006/09/07 17:19:41  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
COMMIT  URID(BBA25A0F7E2700000000000001010000)
        DEFAULTS: GNAME(PLEX1) SYSNAME(SY1)
ATR075I Commit request was scheduled successfully.
```

BACKOUT

BACKOUT	URID(ur-identifier) [GNAME(logging-group-name)] [SYSNAME(system-name)]

where

URID(ur-identifier)
specifies the UR identifier for the InDoubt UR to be backed out. This parameter is required and can not contain any wildcard characters.

GNAME(logging-group-name)
specifies the logging group that contains the UR identifier. If omitted, this parameter defaults to the logging group of the current system. If GNAME is specified, SYSNAME must also be specified.

SYSNAME(system-name)
specifies the system name where the UR resides. If omitted, this parameter defaults to the current system. If SYSNAME is specified, GNAME must also be specified.

Output

```
BACKOUT      2006/09/07 17:19:41  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
BACKOUT URID(BBA25A0F7E270000000000000001010000)
        DEFAULTS: GNAME(PLEX1) SYSNAME(SY1)
ATR076I Backout request was scheduled successfully.
```

DELETERM

DELETERM	RMNAME(rmname) [GNAME(logging-group-name)]

where

RMNAME(rmname)
specifies the resource manager name to be deleted. This parameter is required and can not contain any wildcard characters.

GNAME(logging-group-name)
specifies the logging group that contains the resource manager. If omitted, this parameter defaults to the logging group of the current system.

Output

```
DELETERM      2006/09/07 17:19:41  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
DELETERM RMNAME(UTRICK)
         DEFAULTS: GNAME(PLEX1)
ATR529I RM UTRICK was deleted successfully.
```

UNREGRM

UNREGRM	RMNAME(rmname) [GNAME(logging-group-name)] (SYSNAME(system-name)]

where

RMNAME(rmname)
specifies the resource manager name to be unregistered. This parameter is required and can not contain any wildcard characters.

GNAME(logging-group-name)
specifies the logging group that contains the UR identifier. If omitted, this parameter defaults to the logging group of the current system. If GNAME is specified, SYSNAME must also be specified.

SYSNAME(system-name)
specifies the system name where the resource manager resides. If omitted, this parameter defaults to the current system. If SYSNAME is specified, GNAME must also be specified.

Output

```
UNREGRM      2006/09/07 17:19:41  LOCAL -- ATRQSRV - HBB7740 - 2006250 --
UNREGRM RMNAME(UTRICK)
         DEFAULTS: GNAME(PLEX1) SYSNAME(SY1)
ATR534I RM UTRICK was unregistered successfully.
```

Appendix. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS V2R2 ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out

punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the

default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**

The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted

for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

This information is intended to help the customer to code macros that are available to all assembler language programs. This information documents intended programming interfaces that allow the customer to write programs to obtain services of z/OS.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available at Copyright and Trademark information (http://www.ibm.com/legal/copytrade.shtml).

Glossary

This glossary defines technical terms and abbreviations used in z/OS recoverable resource management services (RRMS) documentation. If you do not find the term you are looking for, refer to the index of this book, or view the *IBM Glossary of Computing Terms*, located at:

http://www.ibm.com/ibm/terminology

ACID transaction
A transaction involving multiple resource managers using the two-phase commit process to ensure ACID (Atomic, Consistent, Isolated, and Durable) properties:

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, the transaction is called atomic. If the transaction is successful, all the changes will be committed. If any piece of the transaction is not successful, then all of the changes will be backed out. An atomic instant occurs when the syncpoint manager in a two-phase commit process logs a commit record for the transaction.

consistent
Applications involved in an ACID transaction must be written to maintain a consistent view of data. The transaction either makes valid changes to data, or it returns all the data to its state before the transaction was started.

isolated
Databases involved in an ACID transaction isolate the updates to their data so that only the application changing the data knows about the individual update requests until the transaction is complete.

durable
Databases involved in an ACID transaction ensure the data is persistent, both before and after the transaction, regardless of success or failure.

agent A component that controls the two-phase commit for a particular node in a distributed transaction. The agent waits for its coordinator to inform it of the outcome of the distributed two-phase commit. Once informed, the agent continues with the second phase of the two-phase commit.

RRS can act as an agent when a resource manager takes the distributed syncpoint manager (DSRM) or server distributed syncpoint manager (SDSRM) role for a particular unit of recovery (UR).

allocation tree
A multilevel logical tree structure representing a hierarchical relationship among transaction programs and other resource managers in a distributed two-phase commit operation. The root node of the tree is the application that starts the transaction, which may not be the initiator of the commit request when you are using a peer-to-peer communication protocol like LU 6.2. See also *syncpoint tree*.

application
In the context of RRS, a program that accesses one or more resources managed by resource managers that work with RRS to manage transactions.

atomic
See *ACID transaction*.

backout
A request to remove all changes to resources since the last commit or backout or, for the first unit of recovery, since the beginning of the application. Backout is also called *rollback* or *abort*.

cascaded transaction
A type of distributed transaction, or part of a distributed transaction, in which the coordination between the nodes is directly controlled by RRS. See also *multisystem cascaded transaction*.

cascaded UR family
The collection of nodes coordinated by RRS in a cascaded transaction.

child UR
A unit of recovery cascaded from a parent UR in a cascaded transaction.

commit
A request to make all changes to resources since the last commit or backout or, for the first unit of recovery, since the beginning of the application.

commit coordinator
See *syncpoint manager*.

communication resource manager
A resource manager that coordinates a two-phase commit across multiple nodes in a distributed transaction. These nodes may be on the same system or multiple systems.

consistent
See *ACID transaction*.

context
Sometimes called a *work context*, a context is a representation of a work request, or part of a work request, in an application. A context may have a series of units of recovery associated with it. See also *native context* and *privately-managed context*.

context interest token
A token that represents a resource manager's interest in a context.

context services
The z/OS system component that provides services used to track a work request and allow a resource manager to express interest in the work request.

context token
A token that represents a work request's context.

conversation
An APPC communication between two applications.

coordinator
The syncpoint manager that controls the two-phase commit process for a node in a distributed transaction. A coordinator that is not the *global commit coordinator* may also be an *agent*.

The system where the top-level UR of a particular *multisystem cascaded transaction* resides is also called the *coordinator* of that multisystem cascaded transaction.

cross-memory resource owning task
The task in an address space that owns its cross memory related resources. Generally, it is the first job step task in an address space after the initiator task.

data resource manager
A resource manager that allows you to retrieve or update protected data and have the changes coordinated via a syncpoint coordinator.

disjoint
A type of distributed transaction in which some of the nodes are connected via different distributed transaction protocols. See also *nondisjoint*.

distributed syncpoint resource manager (DSRM)
A resource manager that extends protection to resources across multiple nodes using a peer-to-peer protocol. See also *server distributed syncpoint resource manager (SDSRM)*.

distributed transaction
A transaction that affects data on multiple nodes. The nodes may be on one system, or across multiple systems. When all the nodes are on one system, the transaction is sometimes called *pseudo distributed*.

durable
See *ACID transaction*.

enterprise identifier (EID)
One possible identifier for a distributed transaction. The format is a concatenation of the TID and GTID used by a communications resource manager that uses the distributed transaction protocols of the Encina Toolkit.

exit manager
A system component, such as context services and resource recovery services, that invokes exit routines.

external coordinator
A coordinator that controls the two-phase commit for a unit of recovery owned by another subsystem or component.

forced update
An update to a log that must be written to nonvolatile storage before processing can proceed. To keep work synchronized

in case of a failure, most RRS logging operations are forced.

forgotten
See *unit of recovery (UR) states in RRS*.

global commit coordinator
The syncpoint manager that controls the overall result of the two-phase commit in a distributed transaction. See also *initiator*.

global transaction
A type of transaction in which a syncpoint coordinator, such as RRS, is available and coordinating changes to resources that could involve multiple resource managers.

heuristic commit (HC)
A decision to commit some, but not all, of the protected resources in an ACID transaction.

heuristic decision
The decision to force an abnormal commit or backout of some but not all of the resources in an ACID transaction. A heuristic decision usually occurs when installation personnel commit or back out some of the resources in an ACID transaction.

heuristic mixed
The condition that occurs when resources in an ACID transaction are in an inconsistent state. Some resources were committed and some resources were backed out.

heuristic reset
The decision to back out some, but not all, of the protected resources in an ACID transaction.

in-backout
See *unit of recovery (UR) states in RRS*.

in-commit
See *unit of recovery (UR) states in RRS*.

in-completion
See *unit of recovery (UR) states in RRS*.

in-doubt
See *unit of recovery (UR) states in RRS*.

in-end See *unit of recovery (UR) states in RRS*.

in-flight
See *unit of recovery (UR) states in RRS*.

in-forget
See *unit of recovery (UR) states in RRS*.

in-only-agent
See *unit of recovery (UR) states in RRS*.

in-prepare
See *unit of recovery (UR) states in RRS*.

in-reset
See *unit of recovery (UR) states in RRS*.

in-state-check
See *unit of recovery (UR) states in RRS*.

initiator
The node of the original issuer of the commit or backout request in a distributed two-phase commit. The initiator is the root of a syncpoint tree. The syncpoint manager of initiator node is known as the *global commit coordinator*.

isolated
See *ACID transaction*.

local transaction
A type of transaction in which each resource manager involved is seperately coordinating its own changes, and only its changes, rather than having a syncpoint coordinator, such as RRS, coordinate them.

logical unit of work identifier (LUWID)
One possible identifier for a distributed transaction used by a communications resource manager that uses LU 6.2 syncpoint protocols.

multisystem cascaded transaction
A *cascaded transaction* that has URs on multiple systems in a sysplex in which the cross system coordination is being provided by RRS. A multisystem cascaded transaction is also known as a *sysplex cascaded transaction* or a *sysplex cascade*.

native context
The automatically occurring context of a work request. A native context is associated with a single task. This context always exists.

node A set of changes to protected resources made by a single work request in a single execution environment. In RRS, a unit of recovery is associated with a work context to form a single transaction node. Multiple nodes may be connected through distributed transaction protocols or be part of a cascaded UR family.

nondisjoint
A type of distributed transaction in which all of the nodes are connected via the same distributed transaction protocol. See also *disjoint*.

parent UR
A unit of recovery in a cascaded transaction with one or more child URs cascaded from it.

privately-managed context
A context created and owned by a resource manager. The resource manager can switch a privately-managed context from one task to another.
Privately-managed contexts are usually used by a resource manager that is also a work manager, like IMS/TM. This sort of work manager can accept and manage transactions, or other kinds of work, from outside the system.

protected conversation
A conversation that connects two nodes in an ACID transaction. Distributed programs use protected conversations to extend protection to resources on multiple systems.

protected resource
A local or distributed resource that can be changed in a synchronized manner during processing coordinated by a syncpoint manager, such as RRS. Databases, conversations between two communications managers, or product-specific resources can all be protected resources. A protected resource is also often called a *recoverable resource*.

recoverable resource management services (RRMS)
The set of three system components that provide resource recovery services in z/OS: resource recovery services (RRS), context services, and registration services.

registered
See *resource manager (RM) states in RRS*.

registration
In RRS, the definition of a resource manager to the system.

registration services
The z/OS system component that enables a resource manager to register itself with the system and identify the exit routines it provides for resource recovery.

reset See *resource manager (RM) states in RRS*.

resource
A database, a conversation between two systems, or a product-specific item. A resource can be local (residing on the current system) or distributed (residing on another system).

A resource is protected when it can be changed in a synchronized manner.

resource manager (RM)
A subsystem or component, such as CICS, IMS, or DB2, that manages resources that can be involved in transactions. There are three types of resource managers: work managers, data resource managers, and communication resource managers.

resource manager (RM) states in RRS
There are five possible resource manager states in RRS:

reset A resource manager in reset state has failed or unregistered itself with the system. It cannot take part in processing units of recovery. Note that a resource manager that is not yet registered with RRS is also in reset state, but it does not appear in the RRS panels.

registered
A resource manager in registered state has registered itself with z/OS registration services, and is ready to set its exit routines with exit managers.

set A resource manager in set state has identified its exit routines with exit managers, such as context services and resource recovery services, and is ready to enter the restart state.

restart A resource manager in restart state has registered and set its exits. While in restart state, it is retrieving and processing any units of recovery that were incomplete the last time it was running. Once the resource manager has restarted, it can proceed to the run state.

run A resource manager in run state is ready to take part in processing units of recovery.

resource recovery services (RRS)
The z/OS system component that provides the services that a resource manager calls to protect resources. RRS is the z/OS system level syncpoint manager.

restart See *resource manager (RM) states in RRS*.

restart anywhere
RRS support that allows resource managers to restart on any system in the same logging group that supports *multisystem cascaded transactions*.

RRMS
See *recoverable resource management services (RRMS)*.

RRS See *resource recovery services (RRS)*.

run See *resource manager (RM) states in RRS*.

server distributed syncpoint resource manager (SDSRM)
A resource manager that extends protection to resources across multiple nodes using a client-server protocol. See also *distributed syncpoint resource manager (DSRM)*.

set See *resource manager (RM) states in RRS*.

state The status of a resource manager (RM) or a unit of recovery (UR). See *resource manager (RM) states in RRS* and *unit of recovery (UR) states in RRS*.

subordinate
A system where a child UR of a particular *multisystem cascaded transaction* resides.

syncpoint
The beginning or ending of a unit of recovery when all resources are consistent.

Syncpoint is also defined as the *process* of doing two-phase commit to make atomic updates to protected resources.

syncpoint manager
A function that coordinates the two-phase commit process for protected resources, so that all changes to data are either committed or backed out. In z/OS, RRS can act as the system level syncpoint manager. A syncpoint manager is also known as a *transaction manager*, *syncpoint coordinator*, or a *commit coordinator*.

syncpoint tree
A multilevel tree structure representing a hierarchical relationship among transaction programs and other resource managers in a distributed two-phase commit operation. The root node of the tree is the initiator, the original issuer of the commit or backout request. The global commit coordinator is the syncpoint manager at the initiator node.

A syncpoint tree is similar to an *allocation tree*. They are identical, except when using a peer-to-peer protocol, such as LU 6.2, for distributed communications. When using peer-to-peer protocols, the commit may occur somewhere other than at the root node of the allocation tree.

sysplex cascaded transaction
See *multisystem cascaded transaction*.

transaction manager
See *syncpoint manager*.

transaction monitor
A work manager that also acts as a syncpoint manager. IMS/TM is a transaction monitor.

two-phase commit
The process used by syncpoint managers and resource managers to coordinate changes in an ACID transaction.

In the first phase of the process, resource managers prepare a set of coordinated changes, but the changes are uncommitted pending the agreement of all the resource managers involved in the transaction. In the second phase, those changes are all committed if the resource managers all agreed to them; or, the changes are all backed out if any of the resource managers failed or disagreed.

Using the two-phase commit process, multiple changes across multiple resource managers can be treated as a single ACID transaction.

unit of recovery (UR)
A set of changes on one node that is committed or backed out as part of an ACID transaction.

A UR is implicitly started the first time a resource manager touches a protected

resource on a node. A UR ends when the two-phase commit process for the ACID transaction changing it completes.

unit of recovery (UR) states in RRS
There are twelve possible UR states in RRS:

in-reset
The UR state before an application program has used any protected resources.

in-flight
The UR state when an application accesses protected resources. The resource managers express interest in the unit of recovery.

in-state-check
The UR state when the application has issued a commit request and the resource managers check if their resources are in the correct state.

in-prepare
The UR state when the application has issued a commit request and the syncpoint manager tells each resource manager to prepare its resources for commit or backout.

in-doubt
For a distributed request, the state of the UR is **in-doubt** on the originating system from the end of the prepare phase of the two-phase commit until the DSRM returns a commit or backout request.

in-only-agent
The UR state when only one resource manager has expressed an interest in the UR. RRS invokes the ONLY_AGENT exit routine to tell the resource manager to process the commit immediately.

in-backout
The UR state when one or more resource managers reply negatively to a commit request. The syncpoint manager tells each resource manager to back out the changes. The resources are returned to the values they had before the UR was processed. When all the resource managers have backed out the changes, the syncpoint manager notifies the application.

in-commit
The UR state when all resource managers reply positively to a commit request. The syncpoint manager tells each resource manager to make its changes permanent. When all resource managers have made the changes, the syncpoint manager notifies the application.

in-end The UR state when the resource managers have responded to the syncpoint manager that commit or backout is complete. The unit of recovery is logically complete.

in-completion
The UR state when any enabled completion exit routines run. Once this phase completes, RRS passes a return code to the application indicating that the changes have been committed or backed out.

in-forget
A UR state for a distributed request. The UR has completed, but RRS is waiting for the SDSRM to indicate how to process the log records for the UR.

forgotten
The UR state that occurs when the UR has completed and RRS has deleted its log records.

work context
See *context*.

work manager
A resource manager that controls the execution of application programs.

work request
A piece of work, such as a request for service, a batch job, an APPC/MVS, CICS, or IMS transaction, a TSO LOGON, or a TSO command. In z/OS, a work request, or part of a work request, is represented by a *context*.

X/Open identifier (XID)
One possible identifier for a distributed transaction used by a communications resource manager that uses the X/Open distributed transaction processing model.

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Product Number: 5650-ZOS

Printed in USA

SA23-1395-02